



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
CAMPUS ITUMBIARA
BACHARELADO EM ENGENHARIA ELÉTRICA

GABRIEL VIRÍSSIMO QUEIROZ LOPES

DESENVOLVIMENTO DE UMA ESTAÇÃO SOLARIMÉTRICA DE BAIXO CUSTO
UTILIZANDO INTERNET DAS COISAS – IOT

ITUMBIARA

2022



INSTITUTO FEDERAL
Goiás

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
SISTEMA INTEGRADO DE BIBLIOTECAS

**TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO
NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG**

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

- | | |
|--|---|
| ☐ Tese | ☐ Artigo Científico |
| ☐ Dissertação | ☐ Capítulo de Livro |
| ☐ Monografia – Especialização | ☐ Livro |
| ☒ TCC - Graduação | ☐ Trabalho Apresentado em Evento |
| ☐ Produto Técnico e Educacional - Tipo: _____ | |

Nome Completo do Autor: Gabriel Viríssimo Queiroz Lopes

Matrícula: 20141040070231

Título do Trabalho: Desenvolvimento de uma estação solarimétrica de baixo custo utilizando internet das coisas - IoT

Autorização - Marque uma das opções

1. ☒ Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. ☐ Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ____/____/____ (Embargo);
3. ☐ Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2** ou **3**, marque a justificativa:

- ☐ O documento está sujeito a registro de patente.
☐ O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
☐ Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

- i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
- ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
- iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

Itumbiara-GO, 11/07/22.
Local Data

Gabriel Viríssimo Queiroz Lopes
Assinatura do Autor e/ou Detentor dos Direitos Autorais

GABRIEL VIRÍSSIMO QUEIROZ LOPES

**DESENVOLVIMENTO DE UMA ESTAÇÃO SOLARIMÉTRICA DE BAIXO CUSTO
UTILIZANDO INTERNET DAS COISAS – IOT**

Trabalho de conclusão de curso apresentado ao Instituto Federal De Educação, Ciência e Tecnologia de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Ghunter Paulo Viajante

ITUMBIARA

2022

Dados Internacionais de Catalogação na Publicação (CIP)

L864d Lopes, Gabriel Viríssimo Queiroz
Desenvolvimento de uma estação solarimétrica de baixo custo
utilizando internet das coisas IOT. / Gabriel Viríssimo Queiroz
Lopes. -- Itumbiara, 2022.
138 p. : il. color.

Trabalho de conclusão de curso (Bacharelado em Engenharia
Elétrica) - Instituto Federal de Educação, Ciência e Tecnologia
de Goiás, Câmpus Itumbiara, 2022.

Orientador: Prof. Dr.º Ghunter Paulo Viajante.

1. Engenharia elétrica 2. Radiação solar 3. Temperaturas. 4.
Sistema solar. I. Título. II. Viajante, Ghunter Paulo. III.
Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

CDD 523.2

Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Biblioteca Maria Gabriela Pacheco Pardey / Câmpus Itumbiara
Bibliotecário: Rosiane Gonçalves de Lima CRB1/1684

GABRIEL VIRÍSSIMO QUEIROZ LOPES

DESENVOLVIMENTO DE UMA ESTAÇÃO SOLARIMÉTRICA DE BAIXO CUSTO UTILIZANDO INTERNET DAS COISAS – IOT

Trabalho de Conclusão de Curso apresentado à banca avaliadora, parte dos requisitos para a obtenção do título de bacharel em Engenharia Elétrica junto ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG), Câmpus Itumbiara.

Aprovado em 11 de julho de 2022.



Prof. Dr. GHUNTER PAULO VIAJANTE
IFG – Campus Itumbiara
Orientador



Prof. Dr. ERIC NERY CHAVES
IFG – Campus Itumbiara
Avaliador



Prof. Dr. MARCELO ESCOBAR DE OLIVEIRA
IFG – Campus Itumbiara
Avaliador

DEDICATÓRIA

Dedico este trabalho a todos aqueles comprometidos com a busca incessante por conhecimento.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais **Crislei Queiroz Lopes dos Santos** e **Fábio Lopes dos Santos** por todo apoio, amor e carinho durante esta jornada.

Agradeço ao professor orientador **Ghunter Paulo Viajante** por toda a ajuda e materiais concedidos para o desenvolvimento deste projeto.

Agradeço ao meu amigo **Renan Moreira Soares** pela ajuda durante o processo de montagem do projeto no instituto federal.

“O maior inimigo do conhecimento não é a ignorância, é a ilusão do conhecimento”.

Stephen Hawking.

RESUMO

Este trabalho apresenta o desenvolvimento e construção de uma estação solarimétrica para medição e armazenamento de valores da temperatura, radiação solar e velocidade do vento utilizando componentes de baixo custo. A elaboração da estação solarimétrica é atrelada ao conceito de *internet* das coisas. Os sensores permanecerão conectados a um microcontrolador para o envio contínuo dos dados de medição por conexão *Wi-Fi* e via protocolo MQTT até uma unidade *Raspberry Pi* responsável pela execução de um gerenciador de dados. Os valores medidos serão exibidos por meio de uma interface gráfica desenvolvida na aplicação *Node-RED* em execução no *Raspberry Pi*. A aplicação manterá o armazenamento local de qualquer novo dado obtido para consultas também através de uma interface gráfica.

Palavras-chaves: internet das coisas, estação solarimétrica, dispositivos, dados, armazenamento, temperatura, velocidade do vento, radiação solar, protocolo mqtt, node-red.

ABSTRACT

This work presents the development and construction of a solarimetric station for measuring and storing temperature, solar radiation and wind speed values using low-cost components. The elaboration of the solarimetric station is linked to the concept of the internet of things. The sensors will remain connected to a microcontroller for the continuous sending of measurement data via Wi-Fi connection and via MQTT protocol to a Raspberry Pi unit responsible for executing a data manager. The measured values will be displayed through a graphical interface developed in the Node-RED application running on the Raspberry Pi. The application will keep the local storage of any new data obtained for queries also through a graphical interface.

Key-words: internet of things, solarimetric station, devices, data, storage, temperature, wind speed, solar radiation, mqtt protocol, node-red.

LISTA DE FIGURAS

Figura 1 - Anemômetro de copos	25
Figura 2 - Anemômetro de ventoinhas	25
Figura 3 - Anemômetro de esfera de arrasto	26
Figura 4 – Termo anemômetro de fio quente	27
Figura 5 - Anemômetro ultrassônico	28
Figura 6 - Anemômetro a laser <i>doppler</i>	29
Figura 7 - Termômetro de expansão a liquido.....	30
Figura 8 - Termômetro bimetálico.....	31
Figura 9 - Termômetro manométrico	32
Figura 10 - Termômetro metálico RTD.....	33
Figura 11 - Termistor.....	33
Figura 12 - Termopar.....	34
Figura 13 - Termômetro infravermelho.....	35
Figura 14- Piranômetro de termopilha	36
Figura 15 - Piranômetro baseado em fotodiodo	37
Figura 16 – Piranômetro fotovoltaico.....	38
Figura 17 - Microcontroladores.....	38
Figura 18 - Microcontrolador ESP32-DevKitM-1	39
Figura 19 - <i>Raspberry Pi</i>	40
Figura 20 – Caminho IOT	41
Figura 21 – Esquemático para o protocolo MQTT	43
Figura 22 - Anemômetro de copos	46
Figura 23 - Sensor de temperatura encapsulado DS18B20	47
Figura 24 - Sensor de irradiância solar Spektron 210	48
Figura 25 - Microcontrolador ESP32	50
Figura 26- <i>Raspberry Pi</i> 3 Modelo B V1.2	51
Figura 27 - Amplificador operacional LM324N	52
Figura 28 - Logo <i>Visual Studio Code</i>	53
Figura 29 - Logo <i>PlatformIO</i>	53
Figura 30 - Logo <i>Mosquitto</i>	54
Figura 31 - Logo <i>Node-RED</i>	55
Figura 32 - Sensores utilizados no projeto	56

Figura 33 – Conexão do ESP32 ao <i>broker</i> via <i>Wi-Fi</i>	56
Figura 34 – <i>Broker Mosquitto</i> gerenciando dados.....	57
Figura 35 – <i>Node-RED</i> armazenando localmente	57
Figura 36 - <i>Visual Studio Code</i> e <i>PlatformIO</i> no microcontrolador ESP32.....	58
Figura 37 – Exemplo da interface final	59
Figura 38 - Pesquisando <i>PlatformIO</i>	61
Figura 39 - Instalando <i>PlatformIO</i>	61
Figura 40 - Extensão <i>PlatformIO</i>	61
Figura 41 - Página inicial da <i>PlatformIO</i>	62
Figura 42 - Criando um projeto	62
Figura 43 - Atribuindo dados do projeto	63
Figura 44 - Local para desenvolvimento do código-fonte.....	63
Figura 45 - Menu de depuração.....	64
Figura 46 - Anemômetro de copos	64
Figura 47 - <i>Pull-Down</i> externo.....	67
Figura 48 - <i>Pull-Up</i> externo.....	67
Figura 49 - Esquema da ligação do anemômetro	68
Figura 50 – Fluxograma para o anemômetro.....	69
Figura 51 - Sensor de temperatura DS18B20.....	70
Figura 52 - Diagrama de blocos do sensor DS18B20	71
Figura 53 - Configuração dos pinos do sensor DS18B20	71
Figura 54 - DS18B20 em modo de alimentação parasita	72
Figura 55 - DS18B20 em modo de alimentação tradicional	72
Figura 56 - Bibliotecas no <i>PlatformIO</i>	73
Figura 57 - Pesquisando uma biblioteca no <i>PlatformIO</i>	74
Figura 58 – Adicionando uma biblioteca no projeto	74
Figura 59 - Fluxograma para o sensor DS18B20	75
Figura 60 - Sensor de irradiância Spektron 210	76
Figura 61 - Circuito condicionador de sinal	77
Figura 62- Esquema de ligação para o sensor Spektron 210.....	78
Figura 63 - Fluxograma para o sensor Spektron 210	78
Figura 64 - Modo ponto de acesso do ESP32.....	80
Figura 65 - Modo estação ESP32	80
Figura 66 - Fluxograma da conexão <i>Wi-Fi</i>	81

Figura 67 – Microcontrolador ESP32 como publicador.....	82
Figura 68 - Pesquisa da biblioteca.....	83
Figura 69 - Adicionando a biblioteca ao projeto	83
Figura 70 - Fluxograma para conexão com <i>broker</i>	84
Figura 71 – <i>Raspberry Pi</i> executando o <i>broker Mosquitto</i>	85
Figura 72 - Terminal de comando do <i>Raspberry Pi</i>	86
Figura 73 - Atualizando aplicativos do <i>Raspberry Pi</i>	86
Figura 74 - Instalando o <i>broker Mosquitto</i>	86
Figura 75 - <i>Broker</i> iniciando com o sistema.....	87
Figura 76 - Configurações do <i>broker Mosquitto</i>	88
Figura 77 - Janela <i>mosquitto.conf</i>	88
Figura 78 - Permitindo conexão remota e com autorização no <i>broker</i>	89
Figura 79 - Usuário para acessar o <i>broker</i>	89
Figura 80 - Senha para acessar o <i>broker</i>	90
Figura 81 - Comando para interfaces de rede.....	91
Figura 82 - Configurações do comando <i>ifconfig</i>	91
Figura 83 - Comando <i>netstat -nr</i>	92
Figura 84 - Número IP do roteador	92
Figura 85 - Comando para <i>dhcpcd.conf</i>	92
Figura 86 - Janela <i>dhcpcd.conf</i>	93
Figura 87 - Configuração para IP estático	94
Figura 88 - <i>Node-RED</i> como assinante	95
Figura 89 - Exemplo de fluxos do <i>Node-RED</i>	95
Figura 90 - Comando para instalação do <i>Node-RED</i> no <i>Raspberry Pi</i>	96
Figura 91 - Instalação concluída do <i>Node-RED</i>	96
Figura 92 - Inicialização do <i>Node-RED</i> pelo menu.....	97
Figura 93 - Início automático do <i>Node-RED</i>	97
Figura 94 - Endereço para acessar a área de trabalho do <i>Node-RED</i>	98
Figura 95 - Área de trabalho do <i>Node-RED</i>	98
Figura 96 - Gerenciador de arquivos	99
Figura 97 - Caminho para arquivos do <i>Node-RED</i>	99
Figura 98 – Pastas ocultas no <i>Raspberry Pi</i>	100
Figura 99 - Local para inserção de usuário e senha do <i>Node-RED</i>	100
Figura 100 - Administrador remoto do <i>Node-RED</i>	101

Figura 101 - Gerando a <i>hash</i> para a senha.....	101
Figura 102 – Novas credenciais para um usuário.....	101
Figura 103 – Acessando o <i>Node-RED</i> por credencias.....	102
Figura 104 - Gerenciador de paletas do <i>Node-RED</i>	102
Figura 105 - Instalação de <i>nodes</i> pelo terminal do <i>Raspberry Pi</i>	103
Figura 106 - Módulo <i>node-red-dashboard</i>	103
Figura 107 - Instalação do <i>dashboard</i> pelo terminal.....	104
Figura 108 - Paleta de <i>nodes</i> para <i>dashboard</i>	104
Figura 109 – Instalação do <i>node</i> para mídia.....	104
Figura 110 - <i>Node</i> media	105
Figura 111 - Instalação do <i>node</i> para arquivos.....	105
Figura 112 - <i>Node fs-file-lister</i>	105
Figura 113 - Configurações do <i>dashboard</i>	106
Figura 114 - Visualizar o <i>dashboard</i>	106
Figura 115 – Interfaces no <i>dashboard</i>	107
Figura 116 – Exemplo de 2 grupos em uma interface.....	107
Figura 117 – <i>Node</i> de conexão MQTT	108
Figura 118 – Propriedades do <i>node mqtt in</i>	109
Figura 119 - Parâmetros do servidor <i>broker</i>	110
Figura 120 - Menu de conexão com o <i>broker</i>	111
Figura 121 - Menu de segurança com <i>broker</i>	111
Figura 122 - Botão <i>Deploy</i>	112
Figura 123 - Estado de conexão do <i>node mqtt in</i>	112
Figura 124 - <i>Node mqtt in</i> em operação	113
Figura 125 - <i>Nodes</i> gráficos.....	113
Figura 126 - Exemplo dos gráficos <i>gauge</i> e <i>chart</i>	114
Figura 127 - Ligação entre o <i>node mqtt in</i> e os gráficos	114
Figura 128 – Criação de 2 interfaces gráficas	114
Figura 129 - Configurações da interface	115
Figura 130 - Configurações do grupo Temperatura	115
Figura 131 - Edição do <i>node gauge</i>	116
Figura 132 – Edição do <i>node chart</i>	116
Figura 133 - <i>Nodes</i> para a Radiação Solar	117
Figura 134 - <i>Nodes</i> para a Velocidade do Vento.....	117

Figura 135 - Interface de dados atuais da temperatura e radiação solar	117
Figura 136 - Interface de dados atuais da radiação e velocidade do vento	118
Figura 137 - Gerenciamento dos dados	119
Figura 138 - Preparando os dados	119
Figura 139 - Gerando nome do arquivo	120
Figura 140 - Listando o arquivo	120
Figura 141 - Condições lógicas para o arquivo	120
Figura 142 - Registrando os dados	121
Figura 143 - Fluxo completo para o armazenamento	121
Figura 144 - Atualizando arquivos	121
Figura 145 - Formatando nome do arquivo	122
Figura 146 - Nome enviado para a tabela.....	122
Figura 147 - Enviando configurações para o gráfico tabela.....	122
Figura 148 - Formatando o nome do caminho do arquivo	123
Figura 149 - Enviando nome para a barra de rolagem	123
Figura 150 - Salvando o nome do arquivo selecionado.....	123
Figura 151 - Fluxo completo para listagem de arquivos	123
Figura 152 - Botão para exibir o gráfico	124
Figura 153 - Leitura do arquivo selecionado.....	124
Figura 154 - Leitura do arquivo em csv.....	124
Figura 155 - Formatando dados para o gráfico	125
Figura 156 - Enviando dados para o gráfico.....	125
Figura 157 - Fluxo completo para a visualização em um gráfico	125
Figura 158 - Interface para o histórico	125
Figura 159 - <i>Node</i> para o caminho do arquivo	126
Figura 160 - Pegar o nome do arquivo selecionado	126
Figura 161 - Leitura do arquivo selecionado.....	126
Figura 162 - Preparação do arquivo para <i>download</i>	127
Figura 163 - Página de requisição para o <i>download</i>	127
Figura 164 - Fluxo completo para o <i>download</i> pela interface.....	127
Figura 165 - Interface com históricos da temperatura e radiação solar.....	127
Figura 166 - Interface com históricos da radiação solar e velocidade do tempo.....	128
Figura 167 – Circuito elaborada para o projeto	129
Figura 168 - Caixa fechada com a tampa aberta	130

Figura 169 - Caixa com o microcontrolador ESP32 posicionada	130
Figura 170 - Sensor de temperatura DS18B20 posicionado	131
Figura 171 - Sensor de irradiância Spektron 210 posicionado.....	131
Figura 172 - Anemômetro de copos posicionado.....	132
Figura 173 - Local de instalação do projeto e sensores.....	132
Figura 174 - Local de instalação do circuito do projeto	133
Figura 175 - Cabeçalho da Interface de dados atuais	134
Figura 176 - Medidor da temperatura atual e gráfico das últimos 24 horas.....	134
Figura 177 - Medidor da radiação solar atual e gráfico das últimas 24 horas	134
Figura 178 - Medidor da velocidade de vento atual e gráfico das últimas 24 horas	135
Figura 179 - Interface de dados atuais completa	135
Figura 180 – Cabeçalho da interface de dados históricos	136
Figura 181 - Histórico da temperatura.....	136
Figura 182 - Histórico da radiação solar	137
Figura 183 - Histórico da velocidade do tempo	137
Figura 184 - Interface de dados históricos completa.....	138
Figura 185 - Monitor para exibição das interfaces no laboratório	139
Figura 186 – Exibição do monitor para apresentação das interfaces	139
Figura 187 - Estação solarimétrica no padrão EPE da Instituição	140
Figura 188 - Sensor EE060.....	141
Figura 189 - Abrigo meteorológico	141
Figura 190 - Piranômetro SR15-D2A2.....	143
Figura 191 - Anemômetro S-WSB-M003	148

LISTA DE GRÁFICOS

Gráfico 1 - Temperatura registrada pelo sensor DS18B20.....	142
Gráfico 2 - Temperatura registrada pelo sensor EE060	142
Gráfico 3 - Comparações entre os sensores DS18B20 e EE060	143
Gráfico 4 - Radiação solar registrada pelo sensor Spektron 210.....	144
Gráfico 5 - Radiação solar registrada pelo sensor SR15-D2A2	144
Gráfico 6 - Comparações entre os sensores Spektron 210 e SR15-D2A2	145
Gráfico 7 - Sensor Spektron 210 das 06:00 às 18:00 horas.....	145
Gráfico 8 - Sensor SR15-D2A2 das 06:00 às 18:00 horas	146
Gráfico 9 - Comparação dos sensores entre as 06:00 e 18:00 horas	146
Gráfico 10 - Medições Spektron 210 no plano inclinado em 18° graus.....	147
Gráfico 11 - Medições Spektron 210 no plano inclinado intervalo de 06:00 às 18:00 horas.	147
Gráfico 12 - Velocidade do vento registrada pelo anemômetro de copos WRFComercial ...	149
Gráfico 13 - Velocidade do vento registrada pelo anemômetro S-WSB-M003.....	149
Gráfico 14 - Comparações entre os anemômetros da WRFComercial e S-WSB-M003.....	150

LISTA DE TABELAS

Tabela 1 - Orçamento da estação solarimétrica padrão EPE.....	151
Tabela 2 - Orçamento da estação solarimétrica de baixo custo.....	152

LISTA DE ABREVIATURAS E SIGLAS

IoT	Internet of things
RTD	Resistance Temperature Detectors
PTC	Positive Temperature Coefficient
NTC	Negative Temperature Coefficient
CPU	Central Processing Unit
PROM	Programmable Read Only Memory
RAM	Random Access Memory
Wi-Fi	Wireless Fidelity
USB	Universal Serial Bus
MQTT	Message Queue Telemetry Transport
TCP/IP	Transmission Control Protocol/Internet Protocol
QoS	Quality of Service
ESP-IDF	Espressif IoT Development Framework
IDE	Integrated Development Environment
GND	Ground
ID	Identity
DQ	Data In/Out
VDD	Power Supply Voltage
VDC	Voltage Direct Current
ADC1	Analog to digital converter 1
ADC2	Analog to digital converter 2
Mbps	Megabits per second
AP	Access Point
STA	Station
APSTA	Access Point and Station
IP	Internet Protocol
CTRL	Control
eth0	Ethernet interface
lo	Virtual loopback interface
wlan0	wireless LAN
CIDR	Classes Inter-Domain Routing
DNS	Domain Name System

TCP/UDP	Transmission Control Protocol/ User Datagram Protocol
SD	Secure Digital
HDD	Hard Disk Drive
SDD	Solid State Drive
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
EPE	Empresa de Pesquisa Energética
ISE	Ingenia Solar Energy
BR/EDR	Basic Rate/Enhanced Data Rate
BLE	Bluetooth Low Energy
GPIO	General Purpose Input/Output
CRESESB	Centro de Referência para as Energias Solar e Eólica Sérgio de Salvo Brito

LISTA DE SÍMBOLOS

°	Grau
nm	Nanômetro
mm	Milímetro
m	Metros
km	Quilômetro
cm	Centímetro
Kg	Quilograma
ms	Milissegundos
mV	Milivolts
kB	Kilobyte
GHz	Gigahertz
GB	Gigabyte
kΩ	Kiloohm
°C	Grau Celsius
watts/m²	Potência por metro quadrado
V	Volts
A	Ampere
W	Watts

SUMÁRIO

1	CAPÍTULO I – INTRODUÇÃO	22
2	CAPÍTULO II – COMPONENTES FÍSICOS	24
2.1	SENSORES	24
2.2	ANEMOMETRIA	24
2.2.1	Anemômetro de copos e de ventoinha	25
2.2.2	Anemômetro de cilindro ou esfera de arrasto	26
2.2.3	Anemômetro a fio quente.....	27
2.2.4	Anemômetro ultrassônico	28
2.2.5	Anemômetro a laser	28
2.3	TERMOMETRIA.....	29
2.3.1	Termômetro mecânico	30
2.3.1.1	Termômetros de expansão.....	30
2.3.1.2	Termômetros bimetálicos	31
2.3.1.3	Termômetros manométricos.....	31
2.3.2	Termômetro de resistência elétrica.....	32
2.3.2.1	Termômetros metálicos – RTD	32
2.3.2.2	Termistores	33
2.3.3	Termopares	34
2.3.4	Termômetros de radiação.....	34
2.4	IRRADIAÇÃO TÉRMICA.....	35
2.4.1	Piranômetro de termopilha	36
2.4.2	Piranômetro baseado em fotodiodo	36
2.4.3	Piranômetro fotovoltaico	37
2.5	MICROCONTROLADORES	38
2.5.1	Microcontrolador ESP32	39
2.6	RASPBERRY PI.....	40
3	CAPÍTULO III – COMPONENTES VIRTUAIS	41
3.1	INTERNET DAS COISAS	41
3.2	LINGUAGEM DE PROGRAMAÇÃO E FRAMEWORK	42
3.3	EDITORES DE CÓDIGO FONTE	42
3.4	PROTOCOLO MQTT.....	43
3.5	APLICATIVO - NODE-RED.....	44

4	CAPÍTULO IV – MATERIAIS E MÉTODOS	46
4.1	MATERIAIS	46
4.1.1	Anemômetro de copos.....	46
4.1.2	Sensor de temperatura DS18B20	47
4.1.3	Sensor de irradiância solar Spektron 210	48
4.1.4	Microcontrolador ESP32	49
4.1.5	<i>Raspberry Pi 3B</i>	51
4.1.6	Condicionador de sinais.....	52
4.2	SOFTWARES.....	53
4.2.1	Editor de código fonte – <i>Visual Studio Code</i>	53
4.2.2	Extensão – <i>PlatformIO</i>	53
4.2.3	Gerenciador de dados – <i>Broker Mosquitto</i>	54
4.2.4	Aplicativo - <i>Node-RED</i>	55
4.3	METODOLOGIA	55
5	CAPÍTULO V – DESENVOLVIMENTO	60
5.1	AMBIENTE DE DESENVOLVIMENTO PARA O ESP32	60
5.2	CONFIGURAÇÃO DO ANEMÔMETRO DE COPOS NO ESP32.....	64
5.3	CONFIGURAÇÃO DO SENSOR DS18B20 NO ESP32.....	70
5.4	CONFIGURAÇÃO DO SENSOR SPEKTRON 210 NO ESP32.....	75
5.5	CONFIGURAÇÕES DA CONEXÃO <i>WI-FI</i> PARA O ESP32.....	79
5.6	CONFIGURAÇÕES DO <i>BROKER</i>	82
5.6.1	<i>Broker</i> no microcontrolador ESP32.....	82
5.6.2	<i>Broker</i> no <i>Raspberry Pi 3B</i>	84
5.6.3	Instalando o <i>broker</i> no <i>Raspberry Pi 3B</i>	85
5.6.4	Configurações de segurança para o <i>broker</i>	87
5.6.5	Configurações de rede para o <i>broker</i>	90
5.7	CONFIGURAÇÃO DO <i>NODE-RED</i>	94
5.7.1	<i>Node-RED</i> no <i>Raspberry Pi 3B</i>	94
5.7.2	Instalando o <i>Node-RED</i> no <i>Raspberry Pi</i>	96
5.7.3	Configurações de segurança no <i>Node-RED</i>	98
5.7.4	Instalação de novos <i>nodes</i> para o <i>Node-RED</i>	102
5.7.5	Manipulações e opções do <i>dashboard</i>	106
5.7.6	<i>Node</i> de conexão com o <i>broker</i>	108
5.7.7	<i>Nodes</i> para gráficos	113

5.7.8	Armazenamento e exibição de dados históricos	118
5.7.8.1	Armazenamento de dados	119
5.7.8.2	Exibição de dados – Listagem na interface	121
5.7.8.3	Exibição de dados - Visualização em um gráfico	124
5.7.8.4	Exibição de dados - <i>Download</i> pela interface	126
6	CAPÍTULO VI – RESULTADOS E COMPARAÇÕES.....	129
6.1	PARTE FÍSICA.....	129
6.2	PARTE VIRTUAL.....	133
6.2.1	Interface de dados atuais	133
6.2.2	Interface de dados históricos.....	136
6.3	ANÁLISE GRÁFICA	140
6.3.1	Resultados e comparações	140
6.3.1.1	Resultados das medições de temperatura	140
6.3.1.2	Resultados das medições de radiação solar	143
6.3.1.3	Resultados das medições da velocidade do vento	148
6.3.2	Perda de dados.....	150
6.4	ANÁLISE ECONÔMICA.....	151
7	CAPÍTULO VII – CONCLUSÃO	153

1 CAPÍTULO I – INTRODUÇÃO

A aquisição de dados é uma forma de poder coletar, armazenar e visualizar informações essenciais de eventos ocorrendo em tempo real.

Os dados documentados podem auxiliar no desenvolvimento ou acompanhamento de projetos, assim como serem fontes para estudos, análises ou avaliações acadêmicas.

Os projetos de desenvolvimento de sistemas de geração de energia elétrica baseado em fontes sustentáveis são exemplos da importância da coleta periódica de dados, uma vez que suas projeções dependem de valores de grandezas físicas que influenciam diretamente em seu princípio de funcionamento e, portanto, no resultado final da energia elétrica gerada.

Para o caso da energia renovável, o histórico da radiação solar e velocidade do vento em determinada localidade são informações utilizadas respectivamente para o dimensionamento de sistemas de geração fotovoltaico e eólica no local em específico.

Logo, seja para estudos ou para a elaboração de projetos como os de energia renovável, a coleta de informações, é de grande importância.

Contudo, nem sempre estes dados podem estar disponíveis para consulta, ou caso estejam, é possível que exista limitações no que foi disponibilizado.

Entre as limitações é possível encontrar apenas uma média de valores de um dia, semana ou mês, conforme concedido por algumas fontes, como o CRESESB, ao invés de uma lista histórica de dados coletados sistematicamente com pequenos intervalos entre cada medição.

Com mais valores de medição seria possível acompanhar as variações e oscilações ocorridas ao decorrer do dia, tendo maior contribuição para estudos ou projetos devido a existência de mais dados.

A possível indisponibilidade de dados de eventos específicos não é o único problema enfrentado, também é conveniente que os valores registrados sejam confiáveis e fidedignos possuindo exatidão e precisão para melhor aproximação da realidade.

Os erros de medições podem estar relacionados a qualidade dos equipamentos ou métodos utilizados para as medições.

Considerando a importância da aquisição de dados e as possíveis adversidades apresentadas, uma solução prática é a possibilidade de construir um sistema que seja capaz de coletar e armazenar dados em um local de escolha própria com a liberdade de estabelecer configurações da maneira mais conveniente possível.

Para tanto, existe um conceito que vem sendo amplamente adotado conforme a tecnologia se torna mais presente em objetos do cotidiano ao ponto de ser possível conectá-los entre si ou estabelecer uma conexão direta com a *internet*. Este conceito é conhecido como *internet das coisas* ou *internet of things*.

Ao passo que a tecnologia evolui, ela passa a ser mais presente não somente no cotidiano como também em dispositivos aprimorados e aperfeiçoados ao longo do tempo, trazendo maiores possibilidades e recursos.

Com a disseminação da tecnologia muitos dispositivos passam a poder estabelecer uma conexão com a *internet* e com outros dispositivos, o que permite uma troca contínua de informação com a menor intervenção humana possível, expandindo o leque de possibilidades para a elaboração de projetos que utilizem estes componentes.

A *internet das coisas* tem como conceito utilizar as possibilidades de conexão entre dispositivos para uma interação mútua entre vários equipamentos, seja para troca ou registro de informações.

Portanto neste trabalho por meio do conceito da *internet das coisas* será desenvolvida uma estação solarimétrica para a coleta e armazenamento dos valores de medição da temperatura, radiação solar e velocidade do vento onde, para este objetivo, uma série de dispositivos de baixo custo serão interconectados para uma operação mútua e contínua ao longo de cada dia.

2 **CAPÍTULO II – COMPONENTES FÍSICOS**

Neste capítulo é introduzido o conceito acerca dos componentes físicos utilizados neste projeto, bem como os modelos alternativos destes mesmos componentes.

2.1 **SENSORES**

Para alguns projetos ou procedimentos é necessário avaliar as características físicas do ambiente ao redor, e são os sensores os dispositivos utilizados para este objetivo.

O sensor é um elemento sensível a algum tipo de energia originada do ambiente como exemplo a luminosidade, energia térmica, energia solar, sendo capaz de relacionar essas variáveis a uma grandeza física conhecida, como intensidade luminosa, temperatura, velocidade, massa entre outras.

2.2 **ANEMOMETRIA**

A anemometria em termos gerais é o estudo da dinâmica do fluxo do ar que tem por objetivo quantificar a natureza do vento através da análise de sua velocidade e intensidade combinadas a sua direção.

O instrumento utilizado para a captura dos dados relacionados ao vento é conhecido como anemômetro, sua função é medir alguns ou todos os componentes do vetor velocidade vento.

O anemômetro ideal seria aquele sensível a pequenos fluxos de ar como os de uma brisa e resistente o suficiente para suportar fluxos mais intensos semelhantes à de um furacão, ser capaz de entregar uma resposta rápida à medida que as flutuações turbulentas ocorrem, além de ter uma saída linear e características de desempenho dinâmico simples [1].

De acordo com as características citadas é possível encontrar vários modelos de anemômetros que tem como objetivo atender a algumas dessas condições.

2.2.1 Anemômetro de copos e de ventoinha

O anemômetro de copos, possui copos conectados a um eixo que rotaciona em proporção à incidência do vento.

Seu eixo gira na presença de um fluxo de ar devido ao coeficiente de arrasto ser maior no interior dos copos em relação a sua superfície lisa e curva da parte traseira, o arrasto é a fricção entre um corpo e um fluido.

Na Figura 1 é demonstrado um anemômetro de copos.

Figura 1 - Anemômetro de copos



Fonte: WRFComercial [7]

Na incidência de um fluxo de ar o anemômetro de ventoinha, semelhantemente ao de copos, rotaciona em proporção a velocidade do vento devido ao arrasto nas pás de sua hélice.

Na Figura 2 é demonstrado um anemômetro de ventoinhas.

Figura 2 - Anemômetro de ventoinhas



Fonte: Minipa [8]

Para que estes modelos de anemômetro possam girar, a força do vento deve superar a força de atrito estático presente internamente no eixo e o momento de inércia do sistema rotacional.

Este valor mínimo de força a ser superado para que o sistema entre em movimento é definido como limiar de velocidade.

Com o valor do limiar de velocidade superado o sistema estará sujeito ao atrito cinético, podendo rotacionar até uma velocidade máxima suportada.

O método de transmissão da informação dependerá do modelo, como exemplo, poderá ser por meio de um sinal elétrico que varia em proporção a rotação do eixo em consequência da velocidade do vento ou através de um ponto de contato que ocorre a cada revolução.

Uma desvantagem na utilização desse sensor é a presença de partes mecânicas móveis, que se desgastam provocando o aumento do atrito com o tempo, sendo necessário manutenção periódica para evitar a degradação dos dados obtidos.

A informação coletada está sujeita ao erro *overspeeding*, onde a velocidade indicada é maior que a velocidade real [1].

2.2.2 Anemômetro de cilindro ou esfera de arrasto

Este anemômetro de arrasto pode ser constituído por um cilindro ou esfera que mede a velocidade do vento através da força de arrasto exercida em sua superfície, o cilíndrico é utilizado para medições bidimensionais da velocidade do vento enquanto que o esférico para medições tridimensionais.

Por natureza este anemômetro possui movimento limitado e não se faz necessário o uso de parte rotacionais.

Na Figura 3 é demonstrado um anemômetro de arrasto composto por uma esfera.

Figura 3 - Anemômetro de esfera de arrasto



Fonte: Carl von Ossietzky Universität Oldenburg [9]

A resposta dinâmica é determinada pelo torque da mola que segura o cilindro ou esfera em uma posição, além de possuir uma resposta rápida, mas a barra utilizada para detectar o

deslocamento pode ser sensível a temperatura e exige altos ganhos dos amplificadores para gerar um sinal de tensão razoável, além de que o cilindro ou esfera pode ser afetado por neve ou gelo mudando sua aerodinâmica [1].

2.2.3 Anemômetro a fio quente

A velocidade do vento é medida por meio da dissipação de calor de um arame ou filme aquecido, que se relaciona com a passagem de fluxo de massa de ar entre o elemento sensível.

Na Figura 4 é demonstrado um termo anemômetro de fio quente.

Figura 4 – Termo anemômetro de fio quente



Fonte: Akrom [10]

A informação é transmitida através de um sinal de corrente elétrica que passa pelo condutor aquecido com uma temperatura constante.

Conforme o elemento condutivo aquece ou esfria em relação a incidência de fluxo de ar sua propriedade de condução elétrica é alterada, o que resultando na variação do valor da corrente elétrica.

Embora este método de medição tenha uma resposta rápida os sensores estão sujeitos a contaminações presente na atmosfera e a alteração dos dados coletados durante chuvas.

2.2.4 Anemômetro ultrassônico

O anemômetro ultrassônico é um instrumento que emite um sinal acústico em um caminho fixo e relaciona seu tempo de percurso neste caminho com a velocidade do vento.

Na Figura 5 é demonstrado um anemômetro ultrassônico.

Figura 5 - Anemômetro ultrassônico



Fonte: Nauticexpo [11]

A partir do tempo necessário para a onda sonora se deslocar entre o ponto de emissão e recepção é possível calcular a velocidade do vento, uma vez que as ondas emitidas sofrem uma alteração em sua velocidade a depender da velocidade do vento.

O anemômetro sônico geralmente é composto por dois ou mais transmissores e receptores.

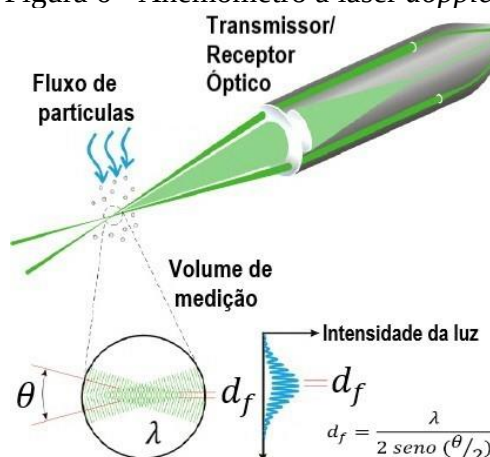
Embora este sensor apresente a vantagem de não possuir partes móveis são normalmente mais caros que os demais.

Os dados fornecidos também podem sofrer alteração devido as chuvas pesadas e neve.

2.2.5 Anemômetro a laser

O anemômetro a laser tem seu funcionamento com base na variação da frequência de um feixe de luz, com propriedades conhecidas, que ocorre devido a difração provocada quando uma partícula em movimento cruza este feixe de luz em específico, causando então esta oscilação em sua frequência [2].

Na Figura 6 é demonstrado um exemplo de funcionamento de um anemômetro a laser.

Figura 6 - Anemômetro a laser *doppler*

Fonte: Adaptado de Dantec Dynamics [12]

Em suma, o anemômetro laser é um sistema óptico que gera feixes de luz e capta a luz difratada quando uma partícula presente no vento cruza o feixe luminoso.

Este comportamento apresenta uma variação de amplitude ao longo do tempo, sendo então a frequência buscada e também conhecida como a frequência *Doppler* [2], por fim através de conversões é relacionado a oscilação desta frequência com a velocidade do vento.

2.3 TERMOMETRIA

A temperatura é uma grandeza física de caráter intensivo e que está relacionada a agitação molecular, isto é, a um nível atômico os átomos que compõem uma molécula estão em constante movimento conhecido como frequência de vibração, logo o aumento desta vibração implica no aumento da temperatura do material que é percebido como a sensação de quente pelo tato, ou frio, conforme esta vibração diminui [3].

O monitoramento do valor da temperatura possibilita, como exemplo, a atuação de projetos de sistemas de controle que tem como objetivo manter um valor de temperatura dentro de uma faixa de operação determinada.

Para que seja possível extrair a informação da temperatura em determinado ambiente geralmente são utilizados materiais sensíveis a agitação molecular, esta sensibilidade altera alguma de suas propriedades físicas que por sua vez são relacionadas a variação da temperatura.

Contudo, a propriedade física que será avaliada irá depender do material utilizado, devido a comportamentos físicos diferentes.

Portanto, um sensor utilizado para termometria relacionará o valor de temperatura ambiente através de um comportamento físico distinto, o qual é possível citar: a expansão do material, que provoca a alteração do comprimento, volume ou pressão; alteração da resistência elétrica; alteração do potencial elétrico de metais diferentes; alteração da potência radiante e alteração da intensidade de carga elétrica em um fotodiodo [3].

2.3.1 Termômetro mecânico

Os medidores de temperatura mecânicos são aqueles que possuem um elemento que altera com facilidade o estado de seu corpo físico devido a variação da temperatura, como comprimento, volume e pressão.

2.3.1.1 Termômetros de expansão

Estes medidores possuem um material que se expande em proporção ao aumento da temperatura, alguns exemplos bem conhecidos são os termômetros de mercúrio ou de álcool.

O funcionamento desses tipos de termômetros baseia-se no coeficiente de dilatação térmica, logo com o aumento da temperatura, o líquido que está contido em um bulbo se expande passando por um canal capilar contido em um tubo de vidro [4].

O líquido que se expande em relação a temperatura indica o valor através de uma escala contida no tubo conforme percorre o canal capilar.

Na Figura 7 é demonstrado um termômetro de expansão a líquido.

Figura 7 - Termômetro de expansão a líquido



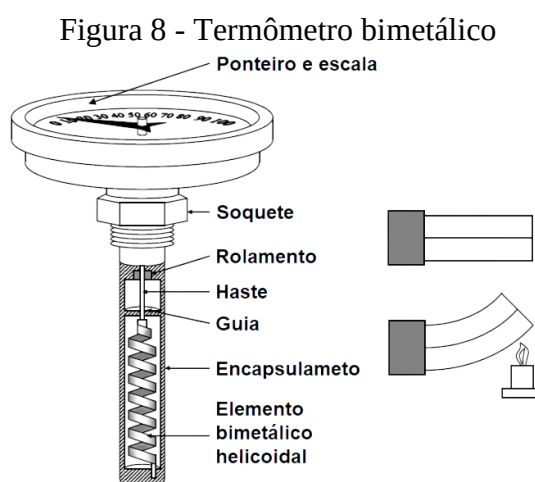
Fonte: Sika [13]

2.3.1.2 Termômetros bimetálicos

Esses indicadores de temperatura são compostos por dois metais firmemente ligados que possuem coeficientes de dilatação diferentes, conforme a temperatura aumenta um destes metais dilata mais que o outro fazendo com que o conjunto enverga.

Essa deformação pode ser utilizada para acionar uma chave em um circuito ou em uma escala calibrada que indica a temperatura conforme o componente bimetálico dilata [4].

Na Figura 8 é demonstrado um esquema de um termômetro bimetálico.



Fonte: Adaptada de Santhosh [14]

2.3.1.3 Termômetros manométricos

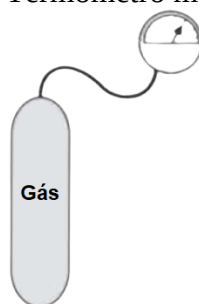
Os termômetros manométricos utilizam algum gás ou vapor que se expande em proporção ao aumento da temperatura.

Geralmente esse elemento está contido em uma câmara que se expande ou retrai por consequência da variação de pressão que surge devido a dilatação do gás em seu interior.

Logo, essa câmara pode ser utilizada para acionar algum elemento em um circuito ou se relacionar a uma escala calibrada que indica a temperatura em proporção a sua deformação.

Na Figura 9 é demonstrado um esquema de um termômetro manométrico.

Figura 9 - Termômetro manométrico



Fonte: Adaptada de Balbinot [4]

2.3.2 Termômetro de resistência elétrica

Os termômetros de resistência elétrica são aqueles que utilizam materiais com propriedades elétricas sensíveis a alteração da temperatura.

2.3.2.1 Termômetros metálicos – RTD

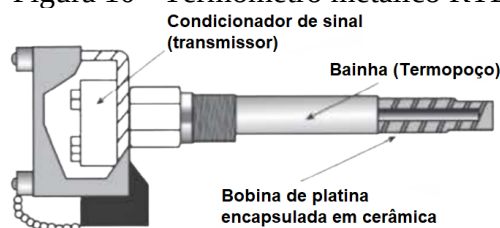
Os sensores RTDs - *Resistance Temperature Detectors* possuem o princípio de funcionamento baseado na variação da resistência elétrica em relação a variação da temperatura, uma vez que a resistência elétrica de metais aumenta conforme sua elevação de temperatura.

Normalmente a platina é o material mais utilizado devido a sua ampla faixa de temperatura, exatidão e estabilidade.

Para relacionar o valor da temperatura com a variação da resistência elétrica do material é necessária a passagem de corrente por esse metal permitindo a análise da variação de tensão sobre ele [4].

Na Figura 10 é demonstrado um esquema de um termômetro metálico do tipo RTD.

Figura 10 - Termômetro metálico RTD



Fonte: Adaptada de Balbinot [4]

2.3.2.2 Termistores

Os termistores são semelhantes aos RTDs uma vez que também associam o valor de temperatura em relação a alteração da resistência elétrica do material utilizado.

No entanto, são formados pela mistura de óxidos metálicos prensados e sintetizados em diversas formas constituindo um semicondutor ao invés de um único elemento metálico como a platina, além de também possuírem um coeficiente de variação térmica geralmente maior que os RTDs [4].

Os termistores usualmente são separados em PTC - *Positive Temperature Coefficient* ao apresentarem um coeficiente de temperatura positivo e NTC - *Negative Temperature Coefficient* ao apresentarem um coeficiente de temperatura negativo.

Os PTCs aumentam sua resistência elétrica conforme a temperatura aumenta enquanto os NTCs diminuem sua resistência elétrica conforme a temperatura aumenta.

Para a utilização desses termistores um circuito elétrico pode ser montado onde a corrente elétrica nesse circuito seria uma função não linear da temperatura em função do termistor [4], mas que pode ser calibrada.

Na Figura 11 é demonstrado um modelo de termistor.

Figura 11 - Termistor



Fonte: Vishay [15]

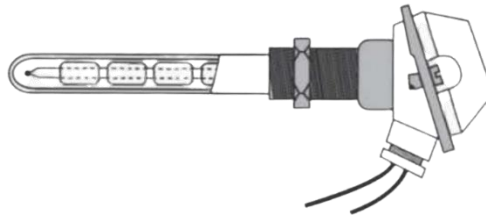
2.3.3 Termopares

Os termopares são sensores ativos que são capazes de gerar seu próprio sinal em estímulo ao valor de temperatura sem a necessidade de alimentação externa.

O termopar é constituído por dois metais de propriedades diferentes unidos em circuito fechado, que passam a ter uma circulação de corrente quando suas junções são expostas a valores diferentes de temperatura.

Na Figura 12 é demonstrado um esquema de um termopar.

Figura 12 - Termopar



Fonte: Adaptada de Balbinot [4]

2.3.4 Termômetros de radiação

Os termômetros de radiação são sensores que detectam a temperatura de um corpo que emana energia térmica através da radiação térmica.

Esses termômetros de radiação utilizam meios ópticos para focarem a energia emitida por um corpo em um detector que converte essa energia em um sinal elétrico proporcional à energia irradiada pelo corpo menos a energia absorvida pelo detector [4].

Na Figura 13 é demonstrado um termômetro infravermelho.

Figura 13 - Termômetro infravermelho



Fonte: Minipa [16]

2.4 IRRADIAÇÃO TÉRMICA

A radiação solar é uma energia emitida radialmente pelo Sol através de ondas eletromagnéticas que incidem no planeta Terra.

Esta forma de energia aquece o planeta permitindo o sustento de várias formas de vida como a humana, é utilizada pela vegetação para a realização da fotossíntese assim como para projetos de energia renováveis e agricultura, além de também se relacionar a variações e mudanças climáticas.

Por ser um elemento de fundamental importância e estar relacionada a vários processos do cotidiano, a radiação solar é um interessante objeto de estudos.

Além da radiação solar também existe outra nomenclatura conhecida como irradiação solar.

Enquanto a radiação solar é a energia transmitida pelo sol sem a necessidade de um meio para propagação, a irradiação solar é definida pela quantidade de radiação solar exposta em determinada área por um determinado intervalo de tempo, outra nomenclatura semelhante é a irradiância solar que representa o fluxo de energia instantâneo em determinada área.

A mensuração da radiação pode ser feita através de um piranômetro, equipamento que possui componentes com propriedades físicas sensíveis à incidência da radiação solar e podem ser encontrados em diferentes modelos.

2.4.1 Piranômetro de termopilha

O piranômetro de termopilha é capaz de fazer medições da radiação em um plano horizontal assim como a densidade de fluxo de radiação solar em um campo com ângulo de visão de 180°.

Neste piranômetro é utilizado uma termopilha interna que converte a energia térmica em energia elétrica, a junção quente da termopilha fica localizada abaixo de um revestimento escuro que permanece em exposição ao ambiente, a junção quente então é aquecida pela radiação absorvida pelo revestimento, já a junção fria fica protegida de qualquer contato com a radiação ou contato térmico.

O revestimento escuro é protegido por uma cúpula de vidro que limita a passagem do comprimento do espectro de onda entre 300 nm a 2800 nm além de proteger o revestimento das condições climáticas como o vento e a chuva por exemplo.

Em modelos de alto custo existe uma segunda cúpula de vidro para reforço da proteção.

Portanto a irradiância solar é relacionada de acordo com a variação da energia elétrica gerada pela termopilha, na Figura 14 é demonstrado um modelo de piranômetro do tipo termopilha.

Figura 14- Piranômetro de termopilha



Fonte: Hukseflux [17]

2.4.2 Piranômetro baseado em fotodiodo

O piranômetro baseado em fotodiodo ou simplesmente piranômetro de silício é composto por um fotodiodo alojado em uma cúpula.

O fotodiodo é um dispositivo semicondutor geralmente feito de silício que pode atuar como um dispositivo fotocondutivo ou fotovoltaico.

Os elementos fotocondutivos são aqueles que alteram o valor da sua resistência elétrica em relação a incidência de luz, que pode ser medida com a passagem de uma corrente elétrica fornecida por uma fonte de alimentação externa, assim a corrente elétrica varia conforme a oscilação da resistência elétrica devido a presença da luz.

Os elementos fotovoltaicos são representados pelos dispositivos capazes de gerar energia elétrica na presença de uma fonte luminosa onde o valor dessa geração irá depender da intensidade luminosa, geralmente estes elementos também são feitos de silício.

Estes semicondutores são formados por uma banda de valência e uma de condução separados por uma lacuna conhecida como *gap*, caso os fótons de uma fonte luminosa possuam energia suficiente, então os elétrons livres na banda de valência, ao receberem esta energia, conseguirão saltar pelo *gap* para a banda de condução gerando a corrente elétrica.

Portanto o fotodiodo pode ser tanto um elemento fotocondutivo quando aproveita a variação da resistência ou fotovoltaico quando aproveita a tensão gerada, em suma o piranômetro de silício relaciona a radiação solar com as oscilações causadas em seu fotodiodo no momento da incidência da luz.

Na Figura 15 é demonstrado um modelo de piranômetro baseado em fotodiodo.

Figura 15 - Piranômetro baseado em fotodiodo



Fonte: Eko [18]

2.4.3 Piranômetro fotovoltaico

O piranômetro fotovoltaico é uma evolução do piranômetro baseado em fotodiodo que surgiu da necessidade do uso de apenas uma célula fotovoltaica para a referência de medições da capacidade de irradiância solar de uma região, sendo também mais rápido em resposta quando comparado ao piranômetro de termopilha.

A medição da irradiância solar neste sensor é realizada relacionando o valor de saída, representado pela diferença de potencial gerado pela única célula fotovoltaica de silício, de acordo com a incidência da radiação solar ao longo do dia.

Este medidor também é conhecido como sensor de irradiância, célula solar de referência, solarímetro ou sensor solar.

Na Figura 16 é demonstrado um sensor de irradiância do tipo fotovoltaico.

Figura 16 – Piranômetro fotovoltaico



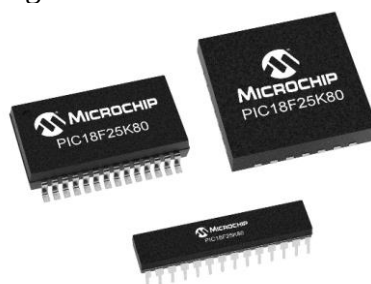
Fonte: Tritec [19]

2.5 MICROCONTROLADORES

Com a evolução dos circuitos digitais e o aumento de sua complexidade passaram a surgir microcontroladores conforme os demonstrados na Figura 17.

Os microcontroladores são componentes eletrônicos que tem a operação determinada por meio de códigos que podem ser gravados em sua memória, bastando uma fonte de alimentação externa para seu funcionamento.

Figura 17 - Microcontroladores



Fonte: Microchip [20]

Em sua construção é possível encontrar uma unidade central de processamento CPU, memória PROM, memória RAM, terminais *input/output* dentre outros periféricos úteis para o funcionamento do microcontrolador.

Contudo, a quantidade de outros recursos que podem ser integrados ao microcontrolador pode variar de fabricante para fabricante assim como seu tamanho e formato. Por reunir vários componentes eletrônicos que trabalham juntos dentro de um único encapsulamento sua utilização se torna mais fácil e barata [5].

Devido a evolução dos microcontroladores entre seus diversos fabricantes é possível encontrar vários modelos pelo mercado que podem atender a várias funcionalidades distintas e trazer recursos que talvez outros modelos não disponham.

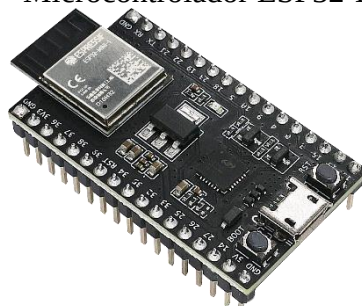
2.5.1 Microcontrolador ESP32

O ESP32 é um microcontrolador desenvolvido pela empresa *Espressif* projetado para trabalhar em dispositivos móveis, aplicações IoT entre outras utilidades.

Este microcontrolador conta com o diferencial de já possuir a tecnologia *Wi-Fi* integrada ao seu *chip*, evitando a necessidade da compra de um adaptador além de possuir a versatilidade de poder ser alimentado por uma conexão via USB dispensando o uso de fontes eletrônicas.

O ESP32 também conta com o recurso *Bluetooth* além de outras especificações técnicas. Na Figura 18 é demonstrado um modelo do microcontrolador ESP32.

Figura 18 - Microcontrolador ESP32-DevKitM-1



Fonte: Espressif [21]

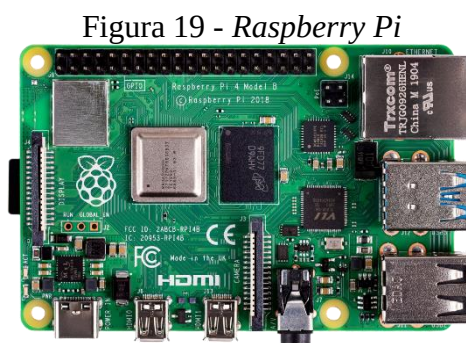
2.6 RASPBERRY PI

O *Raspberry Pi* teve sua origem através de um projeto de Eben Upton que tinha como ponto principal despertar o interesse de jovens em ciências da computação e programação.

A ideia era desenvolver um componente que fosse capaz de apresentar o básico sobre a programação e que introduzisse o conceito geral da computação, mas que, contudo, não tivesse um alto custo, assim viabilizando a disponibilidade desse componente tanto para os jovens como para instituições de ensino.

Logo foi desenvolvido um projeto eletrônico que era capaz de ter um funcionamento semelhante ao de um computador e de baixo custo, suficiente para alcançar o objetivo inicial que foi nomeado de *Raspberry Pi*.

Na Figura 19 é demonstrado um modelo do *Raspberry Pi*.



Fonte: Raspberry Pi [22]

Com o sucesso do *Raspberry Pi* novos modelos foram surgindo após a disponibilização do projeto inicial, trazendo mais recursos e melhorias e seguindo a proposta de se manter a um baixo custo.

3 CAPÍTULO III – COMPONENTES VIRTUAIS

Neste capítulo é introduzido o conceito acerca dos componentes virtuais utilizados para o desenvolvimento deste projeto.

3.1 INTERNET DAS COISAS

O acrônimo IoT - *Internet of Things* é um conceito que trata da conexão digital de vários dispositivos por meio da *internet*.

Conforme a *internet* vem evoluindo, ela passou a ser mais acessível, possibilitando a criação de vários dispositivos e equipamentos acompanhados com recursos como o *Wi-Fi*, que permite uma conexão direta com a *internet* ou entre outros dispositivos acompanhados desta mesma tecnologia.

O recurso para a conexão com a *internet* passa a abranger dispositivos e equipamentos mais simples como microcontroladores, motores elétricos, tomadas, ventiladores, entre outros, criando o cenário ideal para a evolução da *internet* das coisas, onde uma rede pode permanecer com vários dispositivos conectados e trocando informações entre si continuamente.

Na Figura 20 é ilustrado alguns passos tomados pela comunicação entre dispositivos IoT.

Figura 20 – Caminho IOT



Fonte: Elaborada pelo autor

Como exemplo, uma instituição que deseja automatizar alguns processos do cotidiano e coletar dados automaticamente.

Esta instituição pode, por exemplo, fazer a instalação de um sensor para a detecção de movimento em sala de aula enviando um sinal para um microcontrolador responsável por ligar

as luzes e o ar-condicionado desta sala, mantendo-os ligados até que nenhuma presença seja detectada.

O microcontrolador também pode ser conectado a outros sensores como o de temperatura, e através de sua programação enviar periodicamente pela *internet* um registro do momento de acionamento dos equipamentos da sala de aula além do acompanhamento das medições de temperatura.

No fim, ainda seria possível adicionar novas programações para o controle remoto destes equipamentos.

Em resumo um sistema capaz de realizar operações e registros com ou sem a intervenção humana, utilizando o princípio da *internet* das coisas.

3.2 LINGUAGEM DE PROGRAMAÇÃO E *FRAMEWORK*

De modo sucinto, a linguagem de programação é uma sequência de instruções lógicas que especifica o que um hardware deve fazer, enquanto o *framework* é uma estrutura que tem como base uma linguagem de programação específica onde desenvolvedores elaboram funções úteis para o desenvolvimento de projetos a partir deste *framework*.

Uma vez que o código-fonte foi escrito, ele passa por um compilador responsável por converter a forma textual em linguagem de máquina representado pelo binário, dessa forma permitindo que o dispositivo consiga executar cada instrução anteriormente criada.

Para o uso de um microcontrolador é necessária a elaboração de um código fonte que irá instruir o modo como este deve comportar, no entanto a depender do microcontrolador será possível encontrar *frameworks* já desenvolvidas pelos próprios fabricantes ou por grupos de desenvolvimento que facilitam a elaboração de projetos.

3.3 EDITORES DE CÓDIGO FONTE

Os editores de código fonte são ferramentas úteis que auxiliam no desenvolvimento de uma rotina sendo possível encontrar diversas opções pela *internet*.

O *Visual Studio Code*, por exemplo, é um editor de código fonte desenvolvido pela Microsoft e de código aberto, possui o recurso de depuração para a identificação de erros no código, reconhecimento inteligente de código que facilita a escrita, oferece suporte a extensões e várias linguagens de programação além de outras ferramentas úteis para a elaboração de um código.

3.4 PROTOCOLO MQTT

O protocolo MQTT - *Message Queue Telemetry Transport* é um protocolo utilizado para a comunicação entre máquinas sendo amplamente adotado em projetos IoT.

Através dele é possível transmitir pacotes de dados entre dispositivos que utilizam o MQTT através do TCP/IP – *Transmission Control Protocol/Internet Protocol*.

Este protocolo funciona através do conceito de *publish* e *subscribe* que pode ser compreendido como publicar e inscrever, respectivamente.

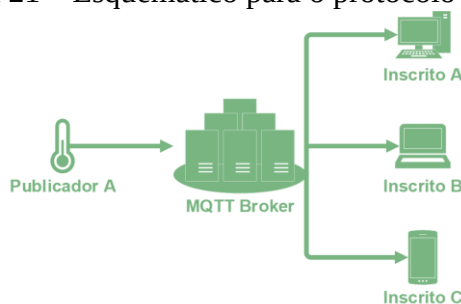
Tanto a publicação como a inscrição são realizadas em tópicos que podem ser interpretados como identificadores de mensagem.

Então, basicamente, o *publish* tem a função de publicar um conteúdo em um tópico específico enquanto o *subscribe* tem por função receber as mensagens que foram enviadas no tópico em que a inscrição foi feita.

Os envios e recebimentos de mensagens através do protocolo MQTT são realizadas por meio de um *broker* que funciona como um intermediário entre os dispositivos, gerenciando as informações providas dos publicadores e as disponibilizando para seus respectivos inscritos.

Na Figura 21 é demonstrado um esquemático do funcionamento do protocolo MQTT.

Figura 21 – Esquemático para o protocolo MQTT



Fonte: Elaborada pelo autor

O protocolo MQTT também possui uma ferramenta chamada QoS – *Quality of Service* o que traduzindo do inglês é conhecido como Qualidade de Serviço, que permite gerenciar a confiabilidade na entrega das mensagens possuindo até três níveis de configurações que são classificadas como 0 – *Almost Once*, 1 – *Atleast Once* e 2 – *Exactly Once*.

Para a configuração 0 – *Almost Once* a mensagem é enviada sem a garantia de recebimento e, caso recebida, não existe nenhuma confirmação de recebimento por parte do destinatário, além de a mensagem não ser armazenada para retransmissão.

Na configuração 1 – *Atleast Once* a mensagem enviada tem a garantia de ser entregue pelo menos uma vez ficando armazenada para retransmissão até a confirmação de recebimento pelo destinatário, mas existindo a possibilidade de uma mensagem duplicada, isto é, receber mais de uma vez a mesma informação.

Na configuração 2 – *Exactly Once* é garantido que a mensagem enviada seja recebida e sem duplicações, contudo esta modalidade deixa o processo mais lento em relação aos métodos anteriores, mas, no entanto, também é o maior nível de confiabilidade disponível.

O gerenciador de dados ou *broker* também possui uma configuração denominada *Keep Alive* representando o tempo máximo que pode decorrer entre o momento que um cliente termina de transmitir um pacote de controle e quando ele começa a enviar o próximo pacote com o intuito de indicar que este cliente ainda está conectado.

O gerenciador de dados multiplica o valor do *Keep Alive* por 1,5, o resultado é o tempo máximo que um cliente pode ficar sem enviar mensagens.

Caso nenhum pacote seja enviado dentro deste intervalo de tempo a conexão entre o cliente e o *broker* é encerrada. Como padrão é utilizado 60 segundos, portanto, depois de 90 segundos se o cliente não enviar nenhuma mensagem, a conexão com o *broker* será encerrada.

3.5 APLICATIVO - NODE-RED

O *Node-RED* é uma ferramenta baseada em navegador que permite uma programação por meio de fluxos construídos através de *nodes* ou nós, estes *nodes* possuem funções definidas que interagem com dados recebidos enviando uma resposta ao próximo *node* conectado no fluxo.

Esta ferramenta permite criar uma representação visual de um problema que pode ser dividido em etapas, onde cada etapa pode ser representada por um *node*, evitando assim o uso de linhas de código ou a necessidade de compreender o código fonte presente em cada *node*.

Caso o usuário possua conhecimento na linguagem de programação textual em JavaScript, também é possível fazer a edição ou criação de *nodes* próprios que atendam a um problema em específico.

Inicialmente esta ferramenta foi desenvolvida como um *proof-of-concept* ou prova de conceito para visualização e mapeamento entre tópico MQTT [6], mas que rapidamente foi se expandido e passando a ser utilizadas em vários projetos, como IoT por exemplo.

4 CAPÍTULO IV – MATERIAIS E MÉTODOS

Neste capítulo é apresentado os componentes físicos e virtuais utilizados para a elaboração do projeto, assim como a metodologia para a sua construção.

4.1 MATERIAIS

Para a construção de uma estação solarimétrica de baixo custo capaz de obter dados da velocidade do vento, temperatura e radiação solar além do armazenamento e disponibilização destas informações, os seguintes materiais e recursos listados a seguir foram utilizados.

4.1.1 Anemômetro de copos

Para o registro da velocidade do vento é utilizado um anemômetro do tipo de copos conforme Figura 22.

Figura 22 - Anemômetro de copos



Fonte: WRFComercial [7]

Especificações técnicas:

- 3 copos de alumínio com 50 mm de diâmetro;
- Suporte de alumínio com 200 mm de comprimento;
- Eixo com rolamento com diâmetro de 210 mm;
- Fio duplo de 9 metros;

- Limiar de velocidade: 0,7 km/h;
- Máxima velocidade suportada 135 km/h.
- Resistente a intempéries;
- Peso de 0,198 kg.

Entre as possibilidades de sensores para a medição de velocidade do vento o anemômetro de copos foi escolhido por ser mais econômico e com possibilidade de ligação direta ao microcontrolador para a leitura de dados.

O modelo escolhido também possui boas características físicas, entre elas, extrema sensibilidade para pequenas variações na velocidade do vento, resistência as intempéries, e ausenta a necessidade de manutenção periódica, como a lubrificação, sendo assim, um modelo ideal para permanecer ao ar livre fornecendo medições ao longo do dia e da noite.

4.1.2 Sensor de temperatura DS18B20

Para as oscilações da temperatura é utilizado o termômetro DS18B20 em sua versão encapsulado que o protege da água, também será necessário o uso de um resistor no valor de 4,7 k Ω para as medições deste sensor.

O DS18B20 é um sensor de temperatura do tipo termistor que conta com um circuito interno para a conversão do sinal analógico em sinal digital. Este sensor é demonstrado na Figura 23.

Figura 23 - Sensor de temperatura encapsulado DS18B20



Fonte: Maxim Integrated [26]

Especificações técnicas:

- 3,0 VDC a 5,5 VDC;

- Precisão $\pm 0,5$ °C de -10 °C a +85 °C;
- Leitura de temperaturas -55 °C a +125 °C;
- Resolução 9 ou 12 bits;
- Interface 1 fio (*One Wire*);
- ID único de 64 bits;
- Alarme de limite de temperatura;
- Tempo de atualização menor que 750 ms;
- Encapsulamento de aço com 6 mm de diâmetro e 35 mm de comprimento;
- Cabo com 4 mm de diâmetro e 90 cm de comprimento.

O sensor de temperatura DS18B20 é utilizado por ser de baixo custo e possuir uma construção que permite ser utilizado em ambientes mais úmidos, possibilitando sua permanência ao ar livre, e por ser um sensor com sinal de saída digital a sua conexão ao microcontrolador se torna mais fácil e prática.

O sensor DS18B20 além de possuir uma boa precisão em suas leituras também conta com o recurso para comunicação por 1 fio, permitindo que vários outros sensores deste mesmo modelo possam ser conectados em uma única porta do microcontrolador, com identificação através de seu ID - *Identity* único.

4.1.3 Sensor de irradiância solar Spektron 210

Para a captura da variação da irradiância solar ao longo do dia é utilizado um piranômetro fotovoltaico, também conhecido como sensor de irradiância solar, modelo Spektron 210, conforme demonstrado na Figura 24.

Figura 24 - Sensor de irradiância solar Spektron 210



Fonte: Tritec [19]

Especificações técnicas:

- Modelo Spektron 210;
- Sensor do tipo célula monocristalina de 33 mm de comprimento e 13 mm de largura;
- Faixa de medição de 0 a 1500 W/m²;
- Precisão de ± 5 % (média anual);
- Saída de aproximadamente 75 mV para 1000 W/m²;
- Proteção IP65;
- Dimensões de 118 x 50 x 44 mm;
- Peso de 0,250 kg.

O sensor Spektron 210 foi escolhido por ser de construção robusta, permitindo o seu uso em diversas condições climáticas e em variados ângulos de inclinação, além de possuir fácil conexão ao microcontrolador.

O sensor Spektron 210 por ser do tipo fotovoltaico e construído com uma célula monocristalina, não será capaz de abranger todos os espectros da luz solar, como seria possível em um piranômetro a termopilha, contudo, o sensor Spektron 210 já é capaz de captar todo o espectro da luz solar que um módulo de geração fotovoltaica obteria, o que já atende aos objetivos do projeto, além de ser uma opção de menor custo se comparada ao piranômetro de termopilha.

4.1.4 Microcontrolador ESP32

O dispositivo responsável por gerenciar os dados originados de cada sensor será o microcontrolador ESP32 já anexado a um módulo de desenvolvimento conforme demonstrado na Figura 25.

Figura 25 - Microcontrolador ESP32



Fonte: Espressif [21]

Especificações técnicas:

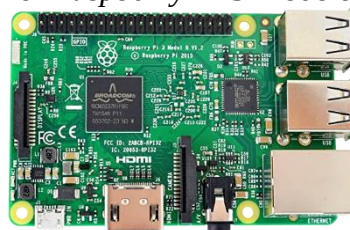
- Chip ESP32-D0WDQ6;
- Processador Xtensa 32-Bit LX6 Dual Core;
- Clock de 80 à 240 MHz;
- Memória ROM 448 kB;
- Memória SRAM 520 kB;
- Tensão de alimentação 4,5 a 12,0 VDC;
- Tensão de nível lógico 3,3 VDC;
- *Wi-Fi* 802.11 b/g/n 2,4 a 2,5 GHz;
- *Bluetooth* 4.2 BR/EDR e BLE;
- Sensor integrado de temperatura e efeito *Hall*;
- Temperatura de trabalho de -40 °C a +85 °C;
- Compatibilidade IDE do Arduino;
- Dimensões 51 x 27,5 x 7 mm.

O microcontrolador ESP32 é utilizado por possuir o recurso de conexão *Wi-Fi* já integrada ao seu *chip* ausentando a necessidade de dispositivos extras, esta conexão é essencial para a comunicação sem fio entre os dispositivos utilizados, além de ser um dispositivo de baixo custo, contando com mais recursos como conexão *Bluetooth* e melhor processamento quando comparada a outros microcontroladores.

4.1.5 Raspberry Pi 3B

Para a execução de um *broker* baseado no protocolo MQTT, aplicações e outras funcionalidades características de um computador é utilizado um *Raspberry Pi 3* Modelo B V1.2. conforme demonstrado na Figura 26.

Figura 26- *Raspberry Pi 3* Modelo B V1.2



Fonte: Raspberry Pi [22]

Especificações técnicas:

- Quad Core 1,2 GHz Broadcom BCM2837 64-bit CPU;
- 1 GB RAM;
- BCM43438 *wireless* LAN e *Bluetooth Low Energy* (BLE);
- 100 Base Ethernet;
- 40 Pinos GPIO;
- 4 portas USB;
- Saída estéreo de 4 polos e porta de vídeo composto;
- HDMI em tamanho real;
- Porta para câmera CSI;
- Porta para display DSI;
- Porta Micro SD para carregamento do sistema operacional e armazenamento de dados;
- Fonte de alimentação micro USB de 2,5 A.

O *Raspberry Pi* modelo 3B é utilizado por ser uma opção de menor custo semelhante a um computador para a execução do *broker* e aplicações, além de ser possível armazenar localmente em seu Micro SD as medições realizadas pelos sensores.

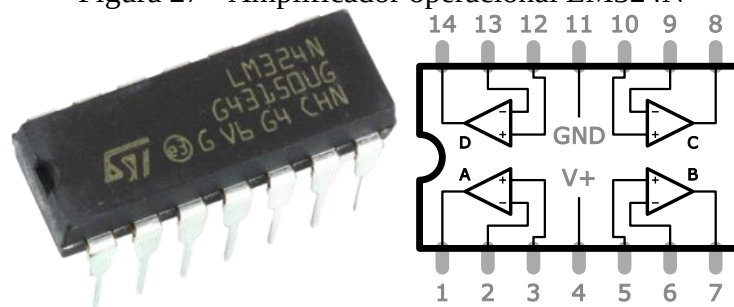
O armazenamento interno do *Raspberry Pi* é utilizado para evitar o uso de serviços online com armazenamento limitado ou o pagamento para obtenção de mais espaço de armazenamento, que por sua vez, fugiria ao objetivo de um projeto de baixo custo.

4.1.6 Condicionador de sinais

O microcontrolador ESP32 possui uma limitação em seu conversor analógico digital que o impede de fazer a leitura correta dos sinais enviados pelo sensor Spektron 210, sendo necessária a elaboração de um circuito amplificador de sinal.

Para tanto, será utilizado o amplificador operacional LM324N que possui em seu encapsulamento 4 amplificadores conforme demonstrado na Figura 27.

Figura 27 - Amplificador operacional LM324N



Fonte: ST [27]

O circuito, a ser demonstrado adiante, também irá utilizar 2 resistores no valor de 10 k Ω , 1 resistor de 100 k Ω e 1 potenciômetro linear de 100k Ω .

4.2 SOFTWARES

4.2.1 Editor de código fonte – *Visual Studio Code*

Para as edições dos códigos a serem desenvolvidos para o microcontrolador ESP32 é utilizado o editor de texto *Visual Studio Code*, cuja a logo é demonstrada na Figura 28.

Figura 28 - Logo *Visual Studio Code*



Fonte: Visual Studio Code [23]

O *Visual Studio Code* é utilizado por ser um editor de texto de código aberto, disponibilizar várias linguagens de programação, por possuir várias funcionalidades e atalhos que facilitam as edições de códigos e oferecer suporte para a instalação de diversas extensões que auxiliam no desenvolvimento dos códigos.

4.2.2 Extensão – *PlatformIO*

O *PlatformIO* é uma extensão disponibilizada pelo editor de código fonte *Visual Studio Code*, utilizada como forma de acesso aos *frameworks* existentes para uso no microcontrolador ESP32, cuja a logo é demonstrada na Figura 29.

Figura 29 - Logo *PlatformIO*



Fonte: PlatformIO [24]

O *PlatformIO* até o momento disponibiliza 25 *frameworks* diferentes, possui compatibilidade com 1.390 modelos de placas e conta com 12.708 bibliotecas, esta extensão também é voltada ao conceito de *internet* das coisas.

O uso desta extensão no projeto se deve ao seu suporte ao microcontrolador ESP32 contado com os *frameworks* Arduino IDE – *Integrated Development Environment* e ESP-IDF – *Espressif IoT Development Framework*, que irá contribuir para o desenvolvimento da programação para o microcontrolador.

O *framework* utilizado será o Arduino IDE, contudo, a opção ESP-IDF estará sempre à disposição para uso em outros projetos.

4.2.3 Gerenciador de dados – *Broker Mosquitto*

Para uso do protocolo MQTT é utilizado o *broker Mosquitto* que permanecerá responsável pelo gerenciamento da troca de dados entre os dispositivos, cuja a logo é demonstrada na Figura 30.

Figura 30 - Logo *Mosquitto*



Fonte: Mosquitto [25]

O *Mosquitto* é utilizado por ser um *broker* baseado em protocolo MQTT e de código aberto, voltado para o conceito de *internet* das coisas e utilizado para troca de mensagens entre dispositivos de baixa potência, além de possuir um código leve e de fácil programação.

4.2.4 Aplicativo - *Node-RED*

A aplicação *Node-RED* é utilizada como forma de criar uma interface gráfica para visualização e gerenciamento das informações coletadas pelos sensores para que assim possa ser facilmente acessada por outros usuários, cuja a logo é demonstrada na Figura 31.

Figura 31 - Logo *Node-RED*



Fonte: Node-RED [6]

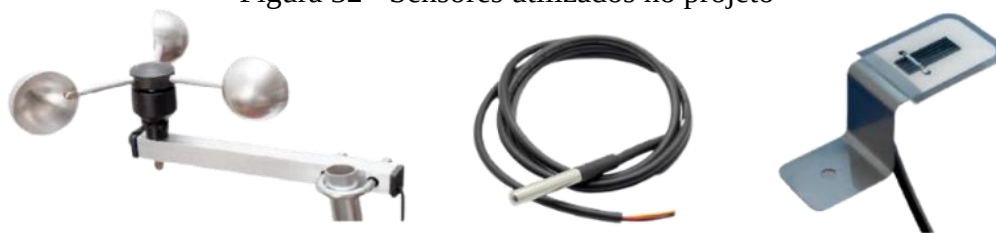
O *Node-RED* é utilizado por ser um aplicativo de código aberto e voltado para dispositivos IoT, possuindo uma ampla comunidade que contribui para um grande acervo de projetos disponibilizados na *internet* que se tornam úteis para consultas e estudos, a aplicação também concede liberdade ao usuário para a edição ou construção de novos *nodes* que atendam um objetivo específico.

A aplicação *Node-RED* também é utilizada por ser adaptada para diversos tipos de telas devido a sua programação baseada em navegador, sendo acessível através de dispositivos móveis e computadores, além de oferecer diversos recursos para a construção e personalização de uma interface gráfica.

4.3 METODOLOGIA

Para a elaboração do projeto são utilizados três tipos de sensores, o anemômetro de copos para a coleta da velocidade do vento, o sensor de temperatura DS18B20 para as oscilações de temperatura e o sensor de irradiância solar Spektron 210 para a variação da radiação solar, apresentados respectivamente na Figura 32.

Figura 32 - Sensores utilizados no projeto



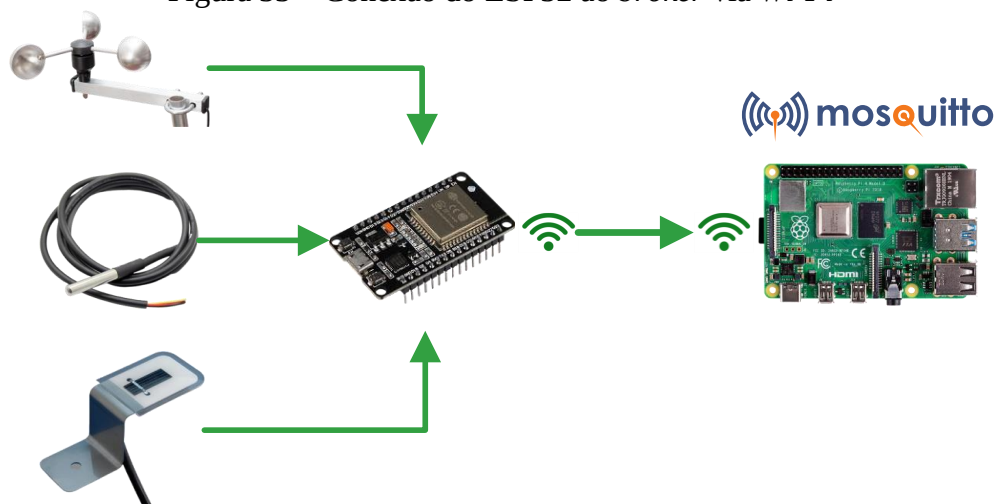
Fonte: Elaborada pelo autor

As informações transmitidas pelos sensores serão direcionadas ao microcontrolador ESP32, responsável por gerenciar os dados recebidos convertendo-os em informação legível.

Em seguida o ESP32 irá transmitir estas informações por meio de sua conexão *Wi-Fi* até o *broker Mosquitto* sendo executado no *Raspberry Pi 3 Modelo B V1.2*.

O *broker Mosquitto* permanecerá gerenciando o recebimento e transmissão de informações entre diversos dispositivos conectados a ele. O esquemático para esta etapa pode ser visualizado na Figura 33.

Figura 33 – Conexão do ESP32 ao *broker* via *Wi-Fi*



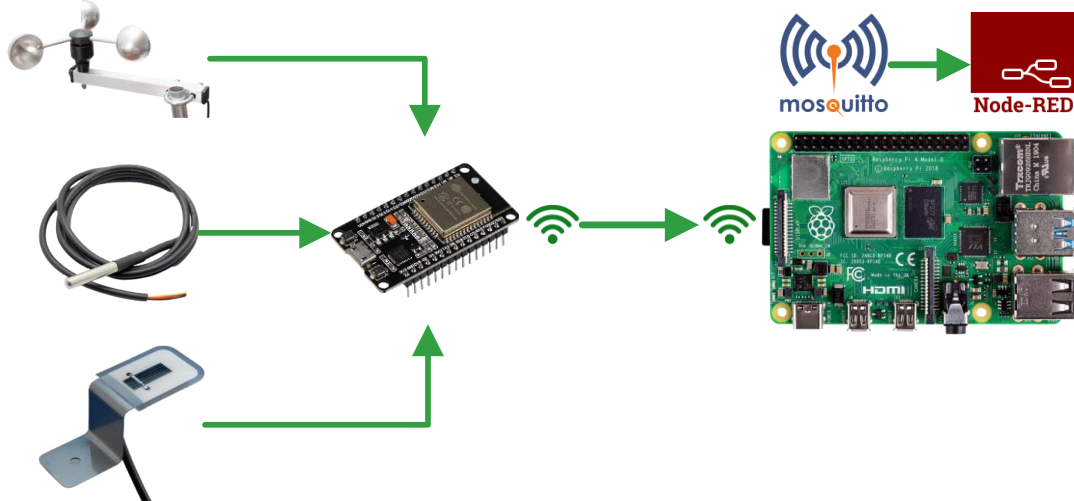
Fonte: Elaborada pelo autor

Com o *broker Mosquitto* atuando como um gerenciador de dados, qualquer outro dispositivo conectado a ele poderá publicar novos dados assim como faz o microcontrolador ESP32 ou então coletar os dados publicados de outros dispositivos.

Como é de interesse a divulgação dos valores obtidos por meio dos sensores a próxima etapa consiste em conectar a aplicação *Node-RED* como um inscrito/assinante do *broker Mosquitto*, que também estará em execução no *Raspberry Pi*.

O *Node-RED* como um assinante atuará coletando qualquer dado nos tópicos de publicação do microcontrolador ESP32 e divulgando em uma interface gráfica a ser desenvolvida no próprio *Node-RED*, conforme esquemático apresentado na Figura 34.

Figura 34 – *Broker Mosquitto* gerenciando dados

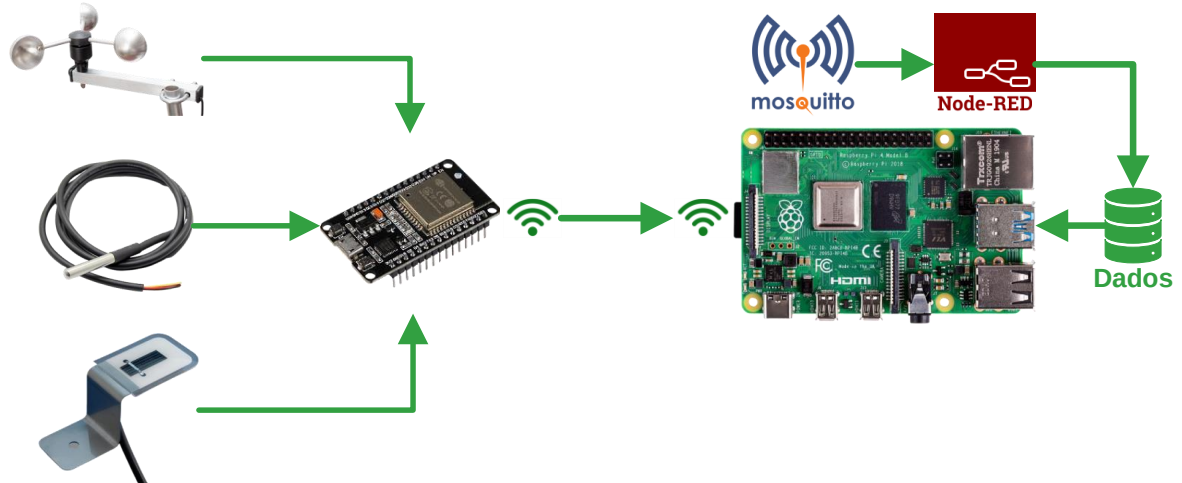


Fonte: Elaborada pelo autor

O *Raspberry Pi* além de executar o *broker Mosquitto* e a aplicação *Node-RED* também atuará como um armazenador de dados local, mantendo a salvo em seu armazenamento um histórico de dados anteriormente recebidos para acesso futuro.

O processo de gerenciamento e armazenamento de dados no *Raspberry Pi* será feito pelo próprio *Node-RED*, conforme esquemático apresentado na Figura 35.

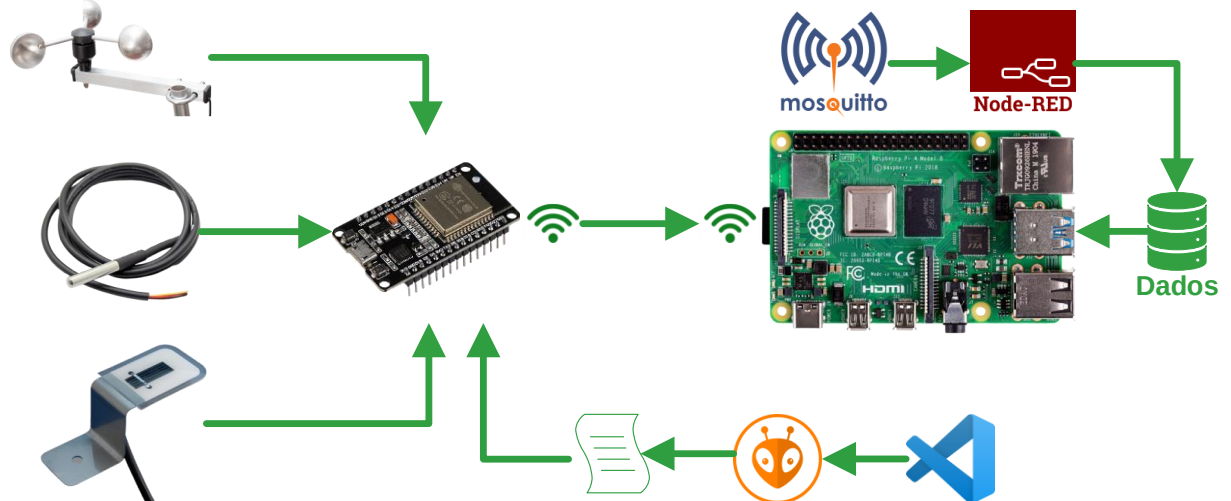
Figura 35 – *Node-RED* armazenando localmente



Fonte: Elaborada pelo autor

Para a programação a ser elaborada e gravada no microcontrolador ESP32 a opção utilizada é o editor de código-fonte *Visual Studio Code* juntamente a extensão *PlatformIO* que é disponibilizada por este mesmo editor, que fornecerá recursos úteis para a programação, conforme apresentado na Figura 36.

Figura 36 - *Visual Studio Code* e *PlatformIO* no microcontrolador ESP32



Fonte: Elaborada pelo autor

Por fim na aplicação *Node-RED* será desenvolvido o fluxo de *nodes* para a coleta e armazenamento de dados, além da elaboração de uma interface gráfica a ser acessada pelo usuário final para a interação com os dados registrados e armazenados anteriormente, incluindo os registrados em tempo real.

Um exemplo da interface a ser desenvolvida é demonstrado na Figura 37.

Figura 37 – Exemplo da interface final



Fonte: Elaborada pelo autor

5 CAPÍTULO V – DESENVOLVIMENTO

Neste capítulo é apresentado em detalhes cada passo tomado para o desenvolvimento e construção da estação solarimétrica.

5.1 AMBIENTE DE DESENVOLVIMENTO PARA O ESP32

Primeiramente é necessário preparar um ambiente de desenvolvimento onde o código-fonte do microcontrolador ESP32 será escrito, utilizando o *Visual Studio Code*. A instalação pode ser feita a partir do *download* do instalador fornecido no próprio site da *Visual Studio Code* podendo ser executado nos sistemas operacionais Windows, macOS e Linux.

O ESP32 possui um *framework* oficial desenvolvido por sua fabricante conhecido como ESP-IDF onde é possível trabalhar diretamente no desenvolvimento de códigos-fonte para serem gravados posteriormente em sua memória sem a necessidade de ferramentas de terceiros.

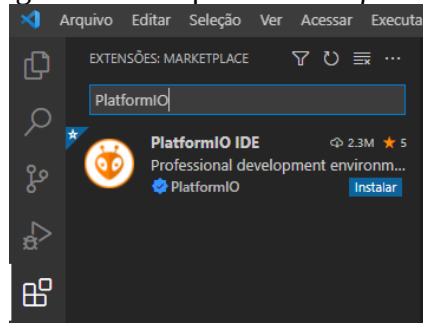
Contudo, ao utilizar o editor *Visual Studio Code* citado anteriormente, é possível fazer a instalação da extensão deste mesmo *framework*.

Embora o ESP-IDF seja o *framework* nativo e oficial para uso no microcontrolador ESP32, esta opção durante a elaboração do projeto, ainda não possuía grande abrangência de material didático disseminado para iniciantes ou grande recurso de bibliotecas disponíveis, sacrificando um pouco da versatilidade e economia de tempo que outros *frameworks* proporcionam.

Por conveniência o ESP32 oferece suporte ao *framework* do Arduino, que facilita o processo de programação para iniciantes, isto por ter uma quantidade de conteúdo didático considerável disponível para consulta além de dar suporte a várias bibliotecas.

Através da extensão *PlatformIO* é possível utilizar qualquer um destes *frameworks* para o microcontrolador ESP32, a instalação é feita por meio do ícone extensão presente no menu ao lado esquerdo da tela inicial do *Visual Studio Code* conforme apresentado na Figura 38.

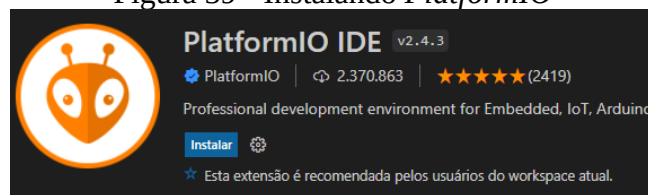
Figura 38 - Pesquisando *PlatformIO*



Fonte: Visual Studio Code [23] e PlatformIO [24]

Quando o menu para instalação de extensões for selecionado, basta prosseguir com o nome da extensão que se deseja instalar, neste caso a busca deve ser feita por *PlatformIO IDE*, em seguida será apenas necessário clicar no botão instalar que fica disponível ao lado da ferramenta pesquisada conforme apresentado na Figura 39.

Figura 39 - Instalando *PlatformIO*



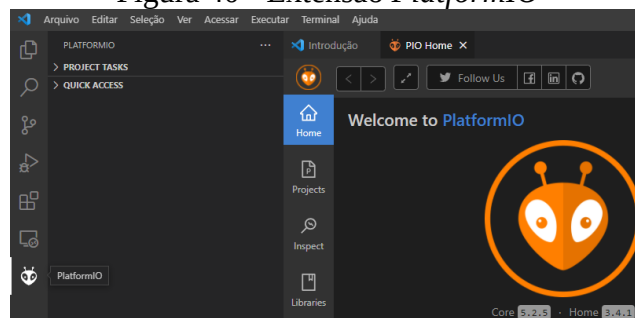
Fonte: Visual Studio Code [23] e PlatformIO [24]

Com a conclusão dos passos anteriores a extensão *PlatformIO* estará pronta para uso.

Por ser uma extensão voltada a IoT, ela oferece suporte a várias plataformas, *frameworks* e placas, facilitando o trabalho para projetos que utilizam diversos dispositivos.

A nova extensão recém instalada pode ser acessada através de seu ícone que ficará disponível no menu ao lado esquerdo do *Visual Studio Code*, precisamente abaixo do ícone utilizado para instalação de extensões conforme exibido na Figura 40.

Figura 40 - Extensão *PlatformIO*

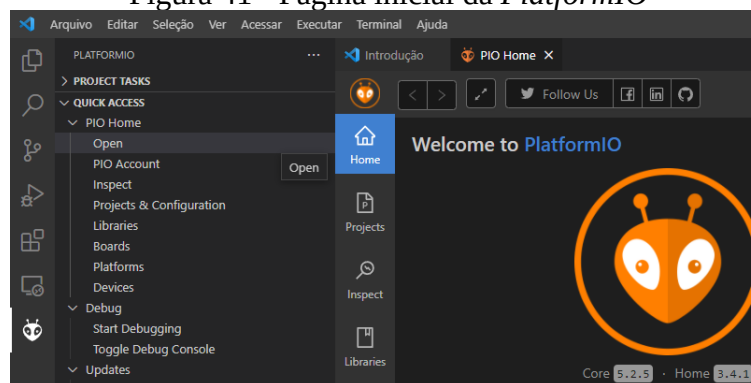


Fonte: Visual Studio Code [23] e PlatformIO [24]

Selecionando o ícone da extensão, será exibido um menu de acesso rápido ao lado onde é possível acessar a página inicial da extensão *PlatformIO*.

A página inicial é aberta através da opção *Open* contida no submenu *PIO Home* conforme demonstrado na Figura 41.

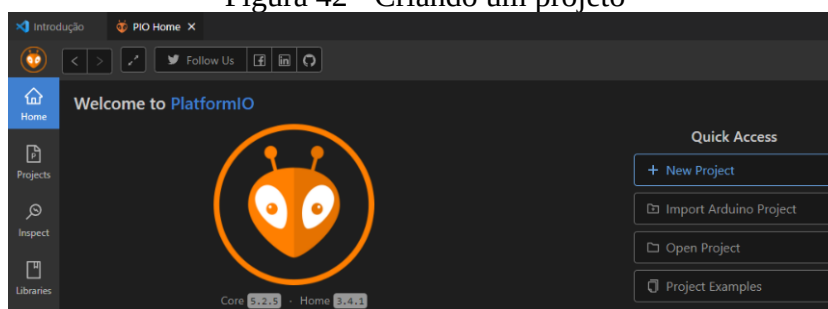
Figura 41 - Página inicial da *PlatformIO*



Fonte: Visual Studio Code [23] e PlatformIO [24]

Um novo projeto pode ser iniciado através do acesso rápido na página inicial da extensão *PlatformIO* selecionando o botão *New Project*, conforme apresentado na Figura 42.

Figura 42 - Criando um projeto

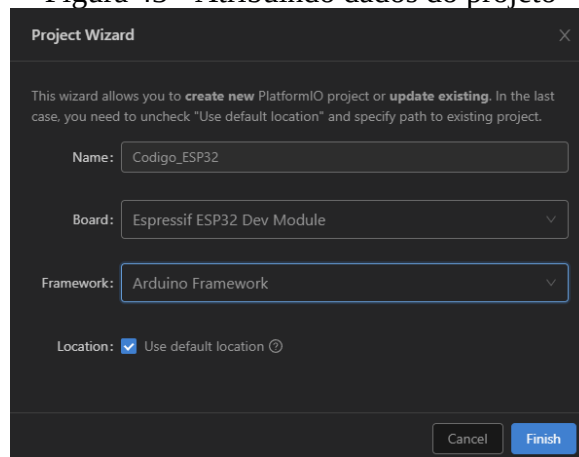


Fonte: Visual Studio Code [23] e PlatformIO [24]

Primeiramente é preciso atribuir um nome ao projeto, especificar o nome da placa de desenvolvimento e o *framework* escolhido.

Para este projeto a placa utilizada é a *Espressif ESP32 Dev Module* com o *framework* Arduino, as especificações podem ser visualizadas na Figura 43.

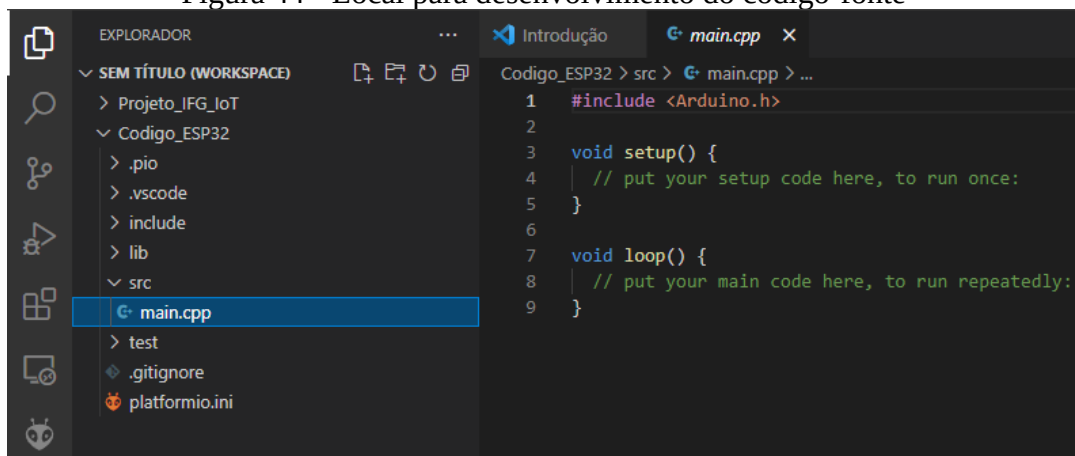
Figura 43 - Atribuindo dados do projeto



Fonte: Visual Studio Code [23] e PlatformIO [24]

Concluída a criação de um novo projeto o arquivo para acesso a área de desenvolvimento do código-fonte estará localizada em *src* com o nome de *main.cpp* no submenu ao lado esquerdo que possui o nome dado na criação do projeto na etapa anterior, conforme demonstrado na Figura 44.

Figura 44 - Local para desenvolvimento do código-fonte



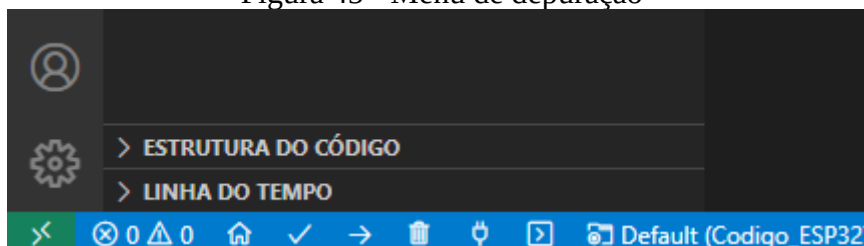
Fonte: Visual Studio Code [23] e PlatformIO [24]

Chegando nesta última parte o ambiente para o desenvolvimento do código-fonte do microcontrolador estará preparado, bastando apenas conectar o ESP32 ao dispositivo com este *software* via conexão USB para que o código-fonte desenvolvido possa ser gravado.

O código-fonte desenvolvido para o microcontrolador ESP32 é detalhado em tópicos adiante.

Em adicional o menu com as ferramentas de depuração, *upload* entre outras opções estão localizadas na parte inferior esquerda do *Visual Studio Code*, conforme demonstrado na Figura 45.

Figura 45 - Menu de depuração



Fonte: Visual Studio Code [23] e PlatformIO [24]

5.2 CONFIGURAÇÃO DO ANEMÔMETRO DE COPOS NO ESP32

O anemômetro utilizado para as medições da velocidade do vento é do tipo de copos, o vento passa pelas concavidades dos copos fazendo-os girar em torno do eixo ao qual estão fixados, além disso internamente deste copo existem dois pontos de contato, sendo esta a forma não direta desse anemômetro indicar a velocidade do vento.

Deste modo sempre que o eixo do anemômetro completa uma revolução estes dois pontos se encontram estabelecendo um contato, entretanto este sensor necessita de um meio secundário que possa ser identificado o momento em que estes contatos ocorrem e a frequência em que ocorrem, para que, então, seja possível mensurar e indicar numericamente a velocidade do vento.

O anemômetro utilizado é demonstrado na Figura 46.

Figura 46 - Anemômetro de copos



Fonte: WRFComercial [7]

Os dois pontos de contato se estendem por meio de fios que podem ser conectados a um dispositivo.

Utilizando o microcontrolador ESP32 é possível identificar os momentos em que cada contato ocorre e utilizar esta informação para desenvolver uma rotina responsável por calcular e disponibilizar um resultado quantificado da velocidade do vento.

Para os cálculos é possível utilizar a relação entre as equações da velocidade angular e a velocidade linear escalar, descrita a seguir.

$$v = \omega * r$$

Em que:

v = Velocidade linear escalar em metros por segundo;

ω = Velocidade angular em radianos por segundo;

r = Raio em metros.

A equação anterior pode ser reescrita da seguinte forma:

$$v = 2 * \pi * frequência * r$$

Portanto para que o valor da velocidade linear escalar possa ser encontrado, basta conhecer o raio entre o centro do eixo rotacional e o centro dos copos no anemômetro, seguido do valor da frequência e utilizando o valor de $\pi = 3,14159265359$.

Para obter a frequência de rotação do anemômetro através do ESP32 é possível utilizar a função de interrupções externas *attachInterrupt*.

A função citada é responsável por averiguar um sinal de interrupção em um terminal escolhido no microcontrolador.

Diferente de uma rotina que segue a ordem de execução no padrão da esquerda para direita e de cima para baixo, a função *attachInterrupt* é executada no momento em que o sinal de interrupção ocorre, sendo uma opção mais fiel para as medições que serão realizadas.

A ideia central para reconhecer cada revolução do anemômetro é definir sinais distintos em dois terminais do microcontrolador para a conexão dos fios do anemômetro individualmente em cada um destes terminais e definir a variação do sinal no momento do contato como uma interrupção.

A estrutura da função *attachInterrupt* necessita de três parâmetros, o primeiro representa o terminal do ESP32 escolhido para avaliar o sinal a ser considerado como uma interrupção, o

segundo é a chamada de uma função criada pelo próprio autor que será executada assim que o sinal considerado como interrupção ocorrer e, por fim, o terceiro é a especificação do sinal que deverá acontecer para que o microcontrolador possa considerar a ocorrência de uma interrupção.

Para a terceira especificação existem cinco condições possíveis, são elas: *LOW*, *CHANGE*, *RISING*, *FALLING*, *HIGH*.

A condição *LOW* aciona a interrupção quando o estado lógico do terminal for baixo.

A condição *CHANGE* aciona a interrupção quando ocorre uma variação no estado lógico do terminal.

A condição *RISING* aciona a interrupção quando o estado lógico do terminal varia de baixo para alto.

A condição *FALLING* aciona a interrupção quando o estado lógico do terminal varia de alto para baixo

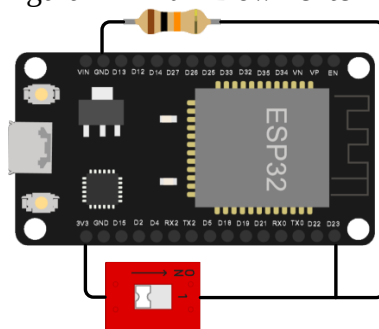
A condição *HIGH* aciona a interrupção quando o estado lógico do terminal for alto.

No ESP32 qualquer terminal digital pode ser utilizado para receber o sinal à ser considerado como interrupção, já a alteração do estado lógico do terminal pode ser feita internamente ou externamente ao microcontrolador.

Esta alteração é necessária para projetos que precisam utilizar terminais com o sinal definido em um estado lógico específico, isto porque quando um microcontrolador está energizado seus terminais permanecem em um estado flutuante, isto é, assumindo aleatoriamente um nível alto ou um nível baixo em seus terminais digitais.

No modo externo, para puxar o estado lógico para baixo também conhecido como *pull-down* é necessário conectar um resistor de 10 kOhms no terminal escolhido e outro terminal no GND – *Ground* do ESP32.

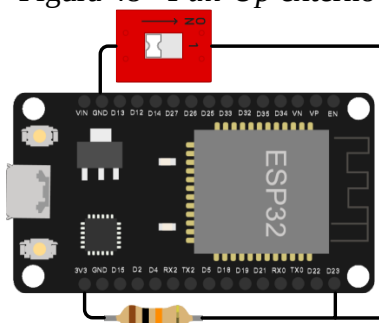
Este esquema irá forçar o terminal a sair do estado flutuante e assumir um estado lógico baixo. Um esquema para o *pull-down* no ESP32 é demonstrado na Figura 47.

Figura 47 - *Pull-Down* externo

Fonte: Elaborada pelo autor

Para puxar o estado lógico para cima também conhecido como *pull-up* é necessário conectar um resistor de 10 kOhms no terminal escolhido e o outro terminal no sinal de tensão 3,3 Volts do ESP32.

Este esquema irá forçar o terminal a assumir o estado lógico alto. Um esquema para o *pull-up* no ESP32 é demonstrado na Figura 48.

Figura 48 - *Pull-Up* externo

Fonte: Elaborada pelo autor

Contudo, existe um meio de realizar estes procedimentos internamente, pois alguns microcontroladores como o ESP32 dispõem destes mesmos resistores no circuito interno de alguns de seus terminais digitais.

A disponibilidade desses resistores deve ser verificada na folha de dados para cada modelo de microcontrolador.

Por fim, basta definir via software que estado o terminal que possui este resistor deve assumir, sem a necessidade de componentes externos.

Tendo a função *attachInterrupt* à disposição é possível conectar um dos fios ao terminal escolhido para analisar as interrupções, definindo um estado lógico como critério e outro fio a um terminal com nível contrário deste primeiro.

Assim com o uso da condição *FALLING* ou *RISING* será possível saber o momento de ocorrência de um contato entre os fios do anemômetro, pois no momento do contato ocorrido ao término de uma revolução o estado lógico do terminal será alterado, provocando o sinal considerado como uma interrupção.

Para este projeto o terminal 4 do ESP32 é utilizado para a avaliação de interrupções com estado lógico definido em baixo, preparado para receber um dos fios do anemômetro.

O fio remanescente é conectado no sinal de tensão 3,3 Volts do próprio ESP32.

Utilizando o modo *RISING* na função *attachInterrupt*, sempre que um contato é realizado entre os fios o nível lógico do terminal 4 passará de baixo para alto e por ser uma subida irá satisfazer a condição *RISING*, acionando então a função secundária responsável por contar cada momento que este processo ocorre.

A frequência é o número de ciclos que algo em um movimento periódico completa em 1 segundo, então é possível encontrá-la fazendo a divisão do número de revoluções contadas pelo intervalo de tempo decorrido. A equação matemática para o cálculo do período pode ser conferida abaixo.

$$F = \frac{n}{\Delta_t}$$

Em que:

F é a frequência;

n representa o número de ciclos;

Δ_t é o intervalo de tempo.

O esquema de ligação entre o anemômetro e o ESP32 é demonstrado na Figura 49.

Figura 49 - Esquema da ligação do anemômetro



Fonte: Elaborada pelo autor

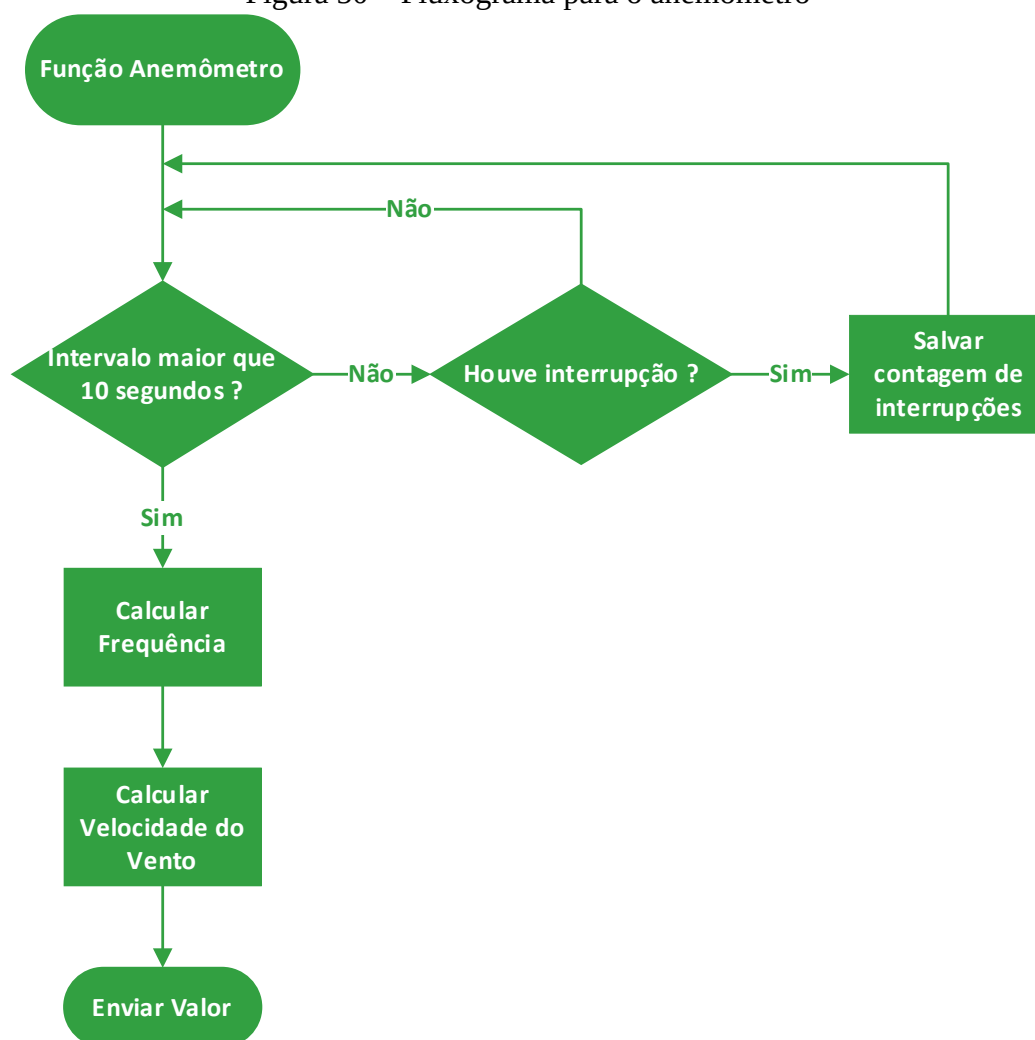
Com o valor da frequência encontrado, finalmente, é possível calcular a velocidade linear escalar do vento.

Note que o valor da frequência neste caso é uma média, isto porque a velocidade do vento está mudando constantemente, então, geralmente, é utilizado um intervalo de tempo que seja suficiente para a contagem de algumas revoluções, sendo assim possível fazer o acompanhamento das oscilações da velocidade do vento.

Caso o intervalo de tempo venha a ser muito longo qualquer oscilação neste período será interpretada como uma única média de medição, diminuindo a precisão dos dados coletados.

O fluxograma apresentado na Figura 50 seguir detalha o modo de operação da rotina desenvolvida no microcontrolador ESP32 para o anemômetro.

Figura 50 – Fluxograma para o anemômetro



Fonte: Elaborada pelo autor

No apêndice 1 se encontra o código-fonte criado para a rotina do ESP32 conectado ao anemômetro de copos.

5.3 CONFIGURAÇÃO DO SENSOR DS18B20 NO ESP32

O sensor de temperatura DS18B20 é um termômetro construído com circuitos integrados que geram uma corrente de saída proporcional ao valor da temperatura, sendo capaz de entregar valores com resolução de 9 a 12 bits, podendo atuar em temperaturas que variam de -55 °C a +125 °C.

O modelo utilizado para o projeto possui um encapsulamento a prova de água, o que permite seu uso em lugares mais húmidos ou para medição dentro de fluidos.

A ponta do sensor onde está localizado o DS18B20 é revestida por aço inoxidável anexado a um cabo com fios extensores conforme demonstrado na Figura 51.

Figura 51 - Sensor de temperatura DS18B20



Fonte: Maxim Integrated [26]

Este sensor é capaz de fornecer dados por um único barramento, sendo necessário então somente um fio para a transmissão das leituras de temperatura.

Também é possível conectar a transmissão de dados de vários destes mesmos sensores em um único fio, o que por consequência diminui a quantidade de terminais utilizados em apenas um terminal do microcontrolador para a entrada de dados.

Esta conexão é possível devido a um ID único de 64 bits presente em cada unidade deste modelo de sensor o que permite identificar individualmente a informação de cada sensor conectado em um único fio.

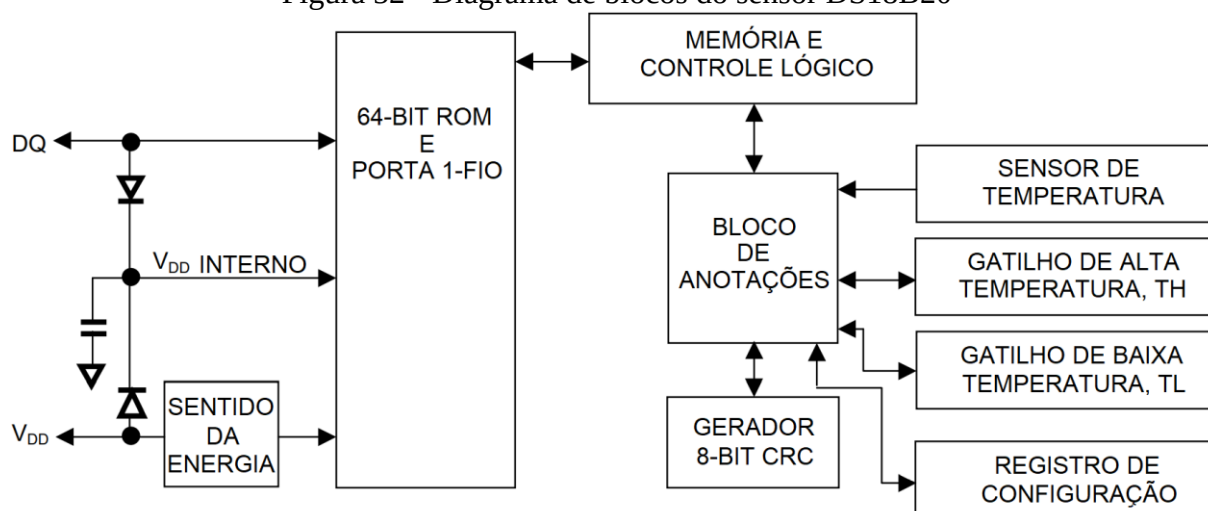
A transmissão de dados por um único fio é interessante para a economia de portas de entrada de dados digitais para projetos que utilizem vários destes sensores.

Outra característica interessante deste sensor é a presença de um circuito de alimentação parasita no terminal de dados, que permite a alimentação do sensor pelo próprio barramento de dados, retirando a necessidade de uma fonte de alimentação externa para seu funcionamento.

Contudo, ainda é possível fornecer energia a este sensor pelo método tradicional, por meio de uma fonte de alimentação externa, caso seja necessário.

Mais informações a respeito do circuito interno do DS18B20 são apresentadas em seu diagrama de blocos demonstrado na Figura 52.

Figura 52 - Diagrama de blocos do sensor DS18B20



Fonte: Adaptada da folha de dados do sensor DS18B20 [26]

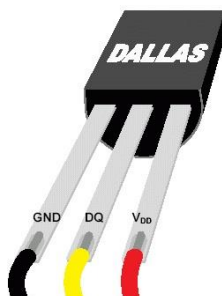
Para a parte externa ao encapsulamento do circuito integrado, existem três terminais que podem ser identificados na versão sem proteção contra a água.

Quando a face plana do sensor está voltada para a frente o primeiro terminal é o aterramento GND, o segundo é saída de dados DQ e o terceiro é a conexão de alimentação externa V_{DD} que deverá estar conectado ao aterramento GND caso o método de alimentação parasita seja utilizado.

Para a versão com proteção a configuração de pinos está classificada por meio de suas cores: a cor preta para o aterramento, a cor amarela para a saída de dados e a cor vermelha para a fonte de alimentação.

Mais detalhes a respeito dos terminais do sensor DS18B20 são ilustrados na Figura 53.

Figura 53 - Configuração dos pinos do sensor DS18B20



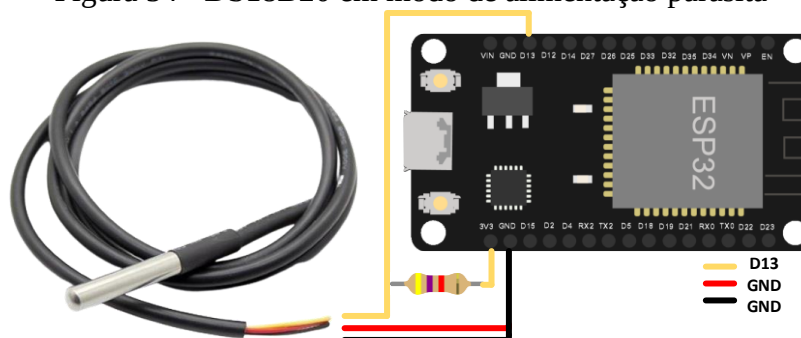
Fonte: Adaptada de Maxim Integrated [26]

Os valores de alimentação para o correto funcionamento do sensor de temperatura DS18B20 podem variar de 3 a 5,5 Volts em tensão contínua, o que deixa o microcontrolador ESP32 dentro da faixa de atuação, uma vez que este possui uma saída de tensão de 3,3 Volts.

Como mencionado anteriormente, o DS18B20 suporta duas configurações de alimentação, o modo de alimentação parasita ou *parasite power mode*, que é capaz de extrair a energia necessária para funcionamento através do barramento de dados, e o modo de alimentação tradicional.

Para o modo de alimentação parasita o esquemático de ligação ao ESP32 ficaria conforme demonstrado na Figura 54.

Figura 54 - DS18B20 em modo de alimentação parasita

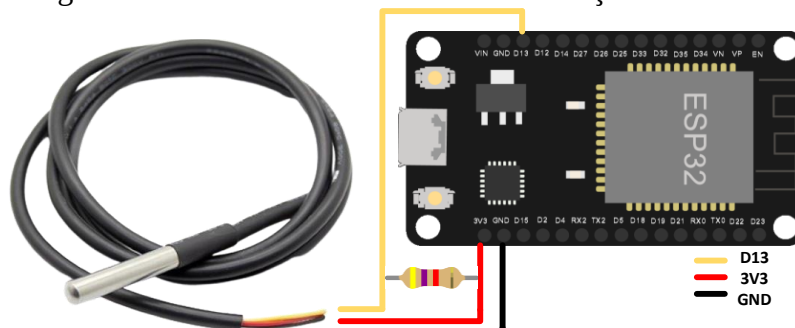


Fonte: Elaborada pelo autor

Neste projeto será utilizado o modo tradicional de alimentação, através da própria saída de tensão contínua de 3,3 Volts do ESP32 e com a porta de dados conectada no terminal 13.

Para isso o esquemático de ligação ficará conforme demonstrado na Figura 55.

Figura 55 - DS18B20 em modo de alimentação tradicional



Fonte: Elaborada pelo autor

Note que ambos os esquemas de alimentação utilizam um resistor de 4,7 kOhms.

Ele é necessário para fazer um *pull-up* externo na conexão de dados, o que permite a alimentação da conexão parasita quando utilizada e a transmissão de dados, além de permitir a conexão de mais sensores.

De acordo com a fabricante do sensor DS18B20, qualquer resistor próximo do valor de 5 kOhms já será suficiente para a sua operação normal.

Com o sensor devidamente conectado, uma maneira simples de estabelecer a comunicação entre o microcontrolador ESP32 e o sensor de temperatura DS18B20 é através de bibliotecas já existentes criadas para uso neste mesmo sensor.

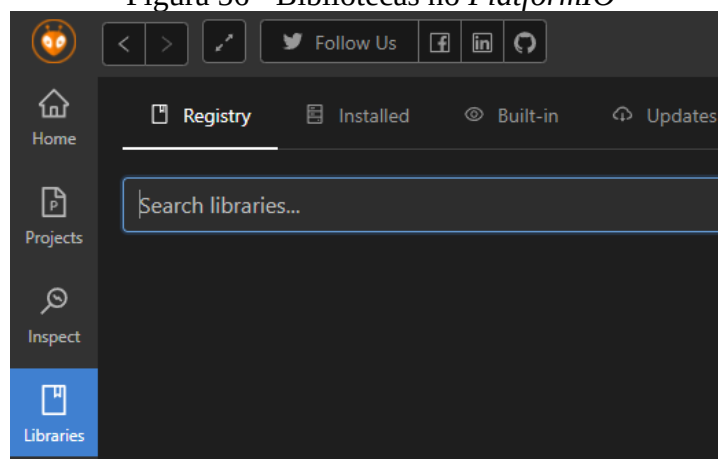
A primeira biblioteca é denominada *OneWire* responsável pelo acesso a sensores com comunicação através de um único fio, a segunda é denominada *DallasTemperature* responsável pelo controle de sensores de circuitos integrados como o DS18B20 e alguns outros modelos.

Juntas estas bibliotecas facilitam a programação no microcontrolador para a leitura de dados de temperatura advindas deste sensor.

Para a instalação destas bibliotecas no projeto é necessário retornar à página inicial da extensão *PlatformIO* e selecionar o ícone *Libraries* localizado ao lado esquerdo da página inicial, onde é possível pesquisar e instalar qualquer biblioteca disponível para uso em projetos.

O ícone para pesquisa de bibliotecas é demonstrado na Figura 56.

Figura 56 - Bibliotecas no *PlatformIO*



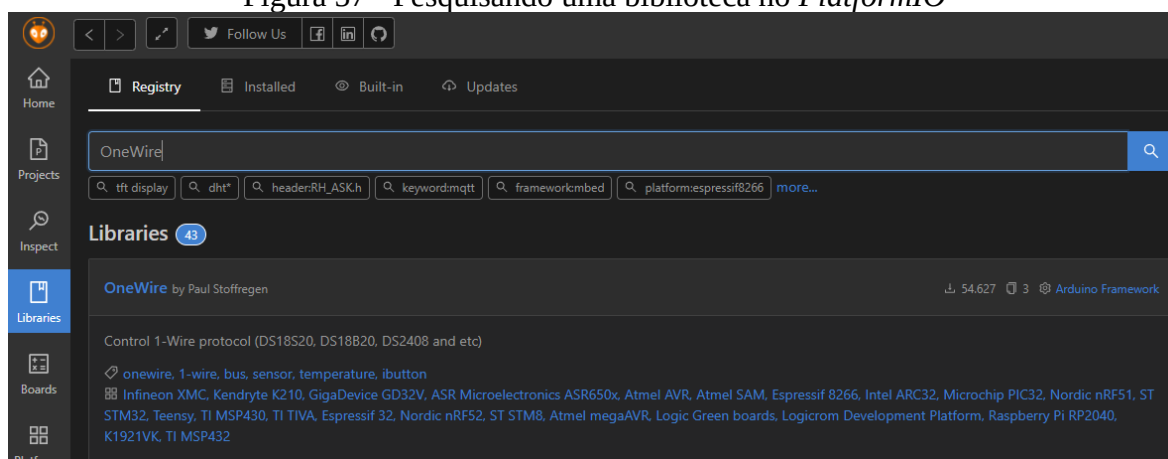
Fonte: Visual Studio Code [23] e PlatformIO [24]

Em seguida cada biblioteca pode ser localizada individualmente digitando seu nome na barra de pesquisa: *OneWire* e *DallasTemperature*, desenvolvidas respectivamente por Paul Stoffregen e Miles Burton.

Ao selecionar cada uma delas o usuário será redirecionado a uma página contendo informações a respeito da biblioteca escolhida e a opção para adição destas ao projeto, onde é possível gerenciar em qual projeto adicionar.

Na Figura 57 é demonstrado a pesquisa pela biblioteca OneWire.

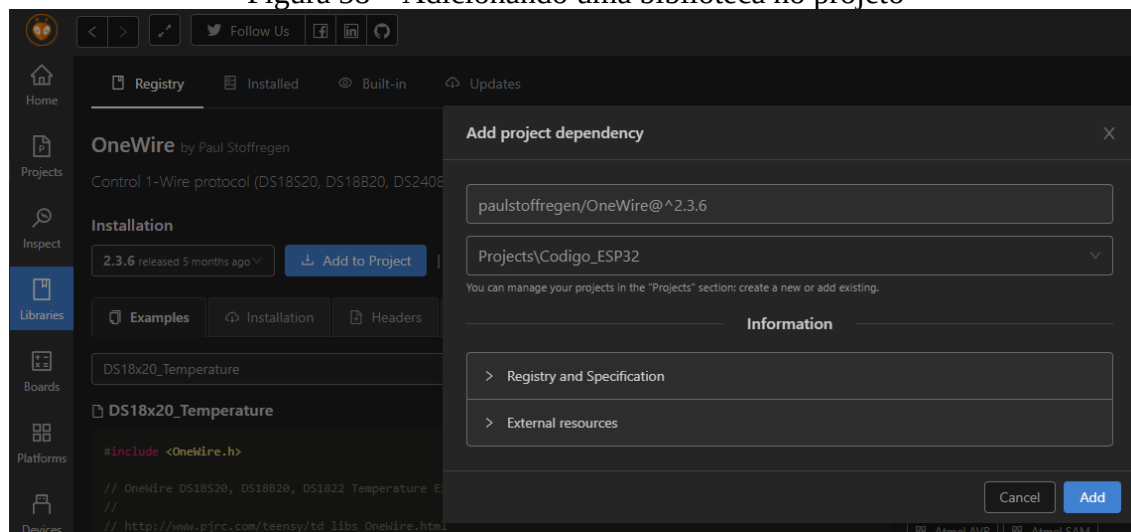
Figura 57 - Pesquisando uma biblioteca no *PlatformIO*



Fonte: Visual Studio Code [23] e PlatformIO [24]

Na Figura 58 é demonstrado o processo de adição da biblioteca ao projeto.

Figura 58 – Adicionando uma biblioteca no projeto



Fone: Visual Studio Code [23] e PlatformIO [24]

Realizando a instalação destas bibliotecas o código para a obtenção dos valores de temperatura pode ser desenvolvido conforme fluxograma apresentado na Figura 59.

Figura 59 - Fluxograma para o sensor DS18B20



Fonte: Elaborada pelo autor

As informações a respeito da programação e implementação das instruções em um código-fonte no microcontrolador ESP32 podem ser encontradas no apêndice 2.

5.4 CONFIGURAÇÃO DO SENSOR SPEKTRON 210 NO ESP32

O sensor de irradiância Spektron 210 é composto por uma única célula fotovoltaica feita de silício, um material semicondutor.

Assim que esta célula é exposta a uma determinada frequência luminosa como a do sol, uma diferença de potencial elétrica é gerada em proporção a luminosidade incidente que pode ser medida através de terminais conectados a esta célula.

O sensor de irradiância Spektron 210 é demonstrado na Figura 60.

Figura 60 - Sensor de irradiância Spektron 210



Fonte: Tritec [19]

A diferença de potencial ocorre devido ao efeito fotovoltaico presente neste material, isto é, a geração de energia elétrica em proporção a incidência de fótons na célula de silício.

Quando um material semicondutor, como o silício, recebe energia suficiente de fótons, seus elétrons presentes na banda de valência passam a possuir energia suficiente para saltar um *gap* existente entre a banda de valência e a banda condutiva, passando a gerar uma corrente elétrica neste material.

Uma célula fotovoltaica é formada por duas metades de sílico, onde cada uma foi eletronicamente alterada em suas composições para possuírem polaridades opostas, e tanto a parte positiva quanto a parte negativa permanecem divididas por um campo elétrico, logo quando o efeito fotovoltaico ocorre os elétrons tendem para o local com menor número de elétrons ou positivamente carregado, isto cria a diferença de potencial elétrico que age de acordo com a intensidade da frequência luminosa incidente.

O sensor de irradiância Spektron 210 tem a proporção de saída em milivolts. De acordo com sua calibragem, 77,48 milivolts equivalem a 1000 watts/m². Em geral é possível se obter um *range* entre 0 a 100 milivolts.

Tendo estes dados a disposição, bastaria realizar a leitura dessas medidas de tensão em uma porta analógica do ESP32 convertendo o valor lido para w/m², de acordo com as informações de calibragem fornecidas.

Contudo, o microcontrolador ESP32 possui limitações em seu conversor analógico-digital que impedem leituras no *range* especificado entre outros detalhes a serem discutidos.

Para a faixa de leitura de tensões elétricas entre 0 a 0,1 Volt o valor lido pelos terminais com conversor analógico-digital do ESP32 será sempre 0 Volt, uma circunstância parecida ocorre para valores na faixa entre 3,2 a 3,3 Volts, onde os terminais sempre apresentarão a leitura de 3,3 Volts.

O microcontrolador ESP32 possui dois controladores analógico-digital com nomenclaturas de ADC1 e ADC2, além da limitação para a leitura de valores localizados entre

0 a 0,1 Volt e 3,2 a 3,3 Volts, estes controladores também não possuem linearidade em sua leitura, gerando uma resposta diferente do esperado, considerando que o ESP32 possui uma resolução de 12 bits.

Como o controlador ADC2 é compartilhado com o driver do módulo de comunicação sem fio do ESP32, os canais ADC2, por padrão, permanecerão inacessíveis para conversão analógica-digital quando o *Wi-Fi* estiver ligado, restando apenas os canais do controlador ADC1 a ficarem disponíveis para as conversões.

As demais funcionalidades dos terminais com conversor analógico-digital não são alteradas. As leituras feitas pelo canal ADC1 também ficarão mais ruidosas quando o *Wi-Fi* estiver ligado.

Como a saída do sensor de irradiância é um valor analógico na ordem de aproximadamente 0 a 0,1 Volt, devido às limitações citadas, estas leituras serão sempre interpretadas como 0 Volt. Para contornar este problema um circuito amplificador de sinal pode ser desenvolvido com o objetivo de elevar o sinal original a uma faixa de 0 a 3,3 Volts.

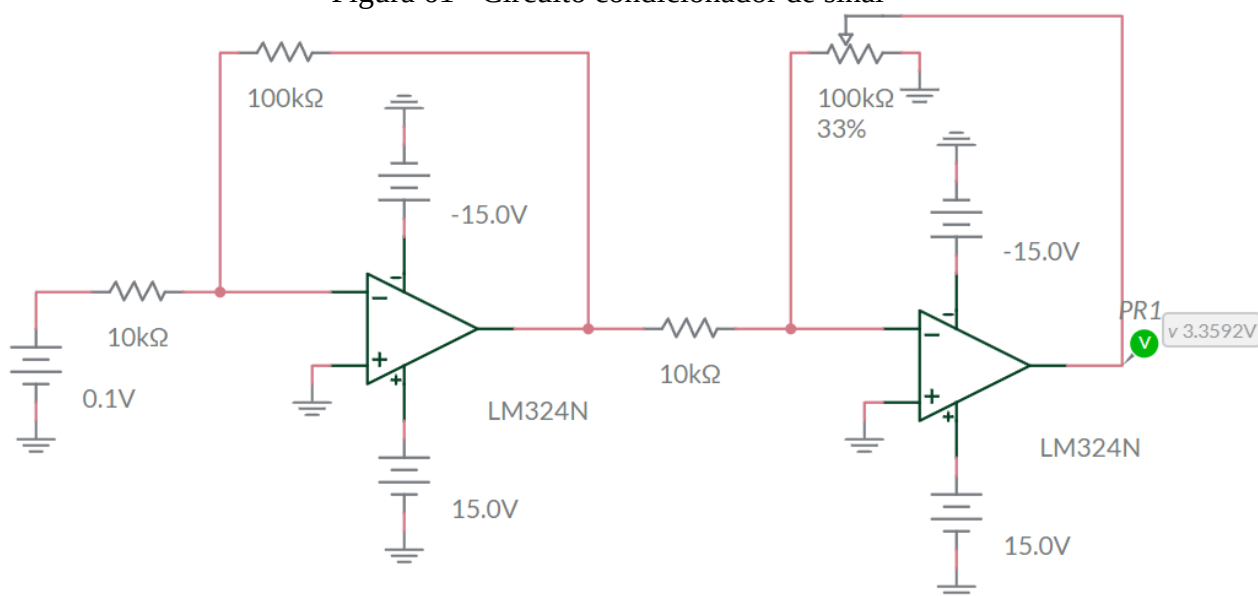
O circuito condicionador de sinal projetado utiliza o circuito integrado LM324N que possui 4 amplificadores operacionais em sua composição.

Juntamente aos amplificadores operacionais são utilizados dois resistores de 10 k Ω , um resistor de 100 k Ω e um potenciômetro de 100 k Ω .

Para a alimentação deste circuito uma fonte simétrica de 15 Volts é utilizada.

O esquema do circuito condicionador de sinal desenvolvido é apresentado na Figura 61.

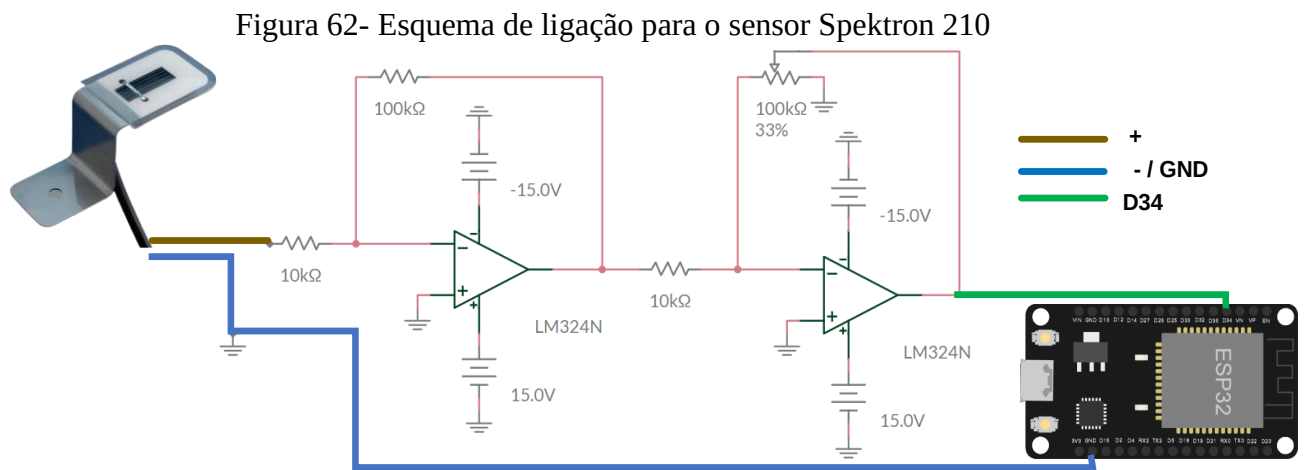
Figura 61 - Circuito condicionador de sinal



Fonte: Elaborada pelo autor

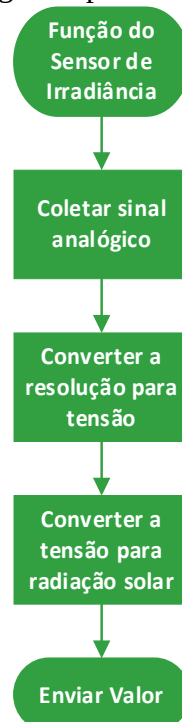
Com o circuito elaborado, sua entrada irá receber a saída positiva do sensor de irradiância Spektron 210, representada pela cor marrom, enquanto a saída negativa, representada pela cor azul, pode ser aterrada, levando-se em consideração que todo o sistema compartilha do mesmo ponto de aterramento.

Por fim a saída do circuito condicionador será conectada com o terminal 34 do ESP32 para a leitura analógica do sinal amplificado, conforme ilustrado na Figura 62.



O fluxograma demonstrado na Figura 63 ilustra a rotina do código desenvolvido.

Figura 63 - Fluxograma para o sensor Spektron 210



O código-fonte para a leitura das medições do sensor de irradiância pode ser encontrado no apêndice 3.

5.5 CONFIGURAÇÕES DA CONEXÃO *WI-FI* PARA O ESP32

Um dos recursos vantajosos presentes no ESP32 é a sua capacidade de estabelecer uma conexão *Wi-Fi* permitindo a troca de dados entre outros dispositivos com o mesmo recurso.

De acordo com suas especificações o *chip* ESP32 possui um sinal de 2,4 GHz sendo capaz de atingir uma taxa de transmissão de dados de até 150 Mbps.

Este microcontrolador também disponibiliza 4 formas de interface *Wi-Fi*, que podem ser configuradas de acordo com as preferências do usuário, sendo os modos AP, STA, APSTA e NULL.

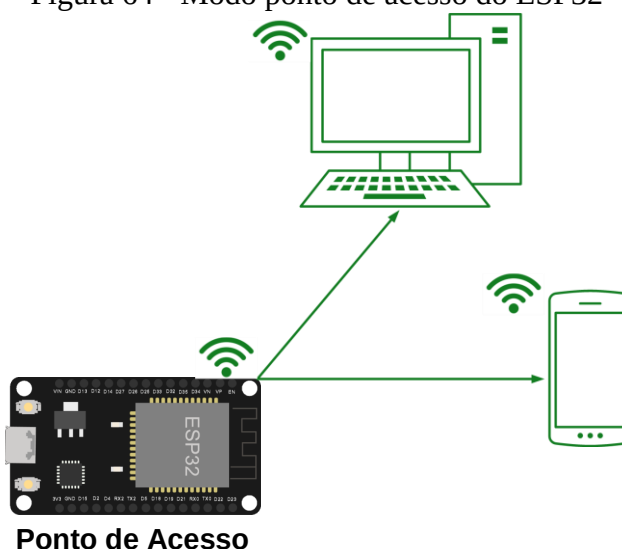
Para o modo AP - *Access Point* o microcontrolador ESP32 é configurado como um ponto de acesso, criando uma rede de conexão *Wi-Fi* própria. A partir deste ponto qualquer dispositivo com o recurso *Wi-Fi* poderá se conectar diretamente ao ESP32.

No entanto, ao utilizar o modo AP, não será possível se conectar à *internet*, logo tanto o ESP32 quanto os demais dispositivos conectados a este microcontrolador não poderão acessar qualquer serviço *online* da *internet*.

Na configuração AP a persistência da conexão entre o microcontrolador e algum outro dispositivo irá depender da distância entre eles por causa do limite de alcance do sinal do ESP32.

Na Figura 64 é apresentado um exemplo de esquema de operação do microcontrolador em modo AP.

Figura 64 - Modo ponto de acesso do ESP32



Fonte: Elaborada pelo autor

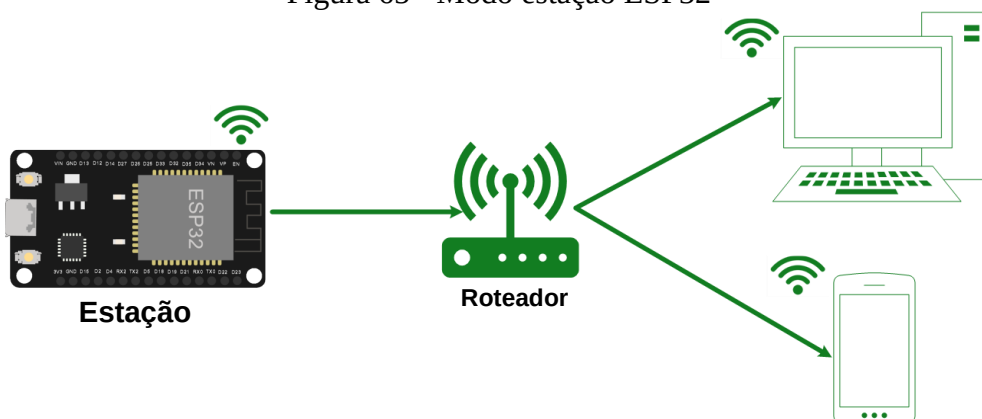
No modo STA - *Station* o microcontrolador ESP32 é configurado como uma estação permitindo a conexão com outras redes, como por exemplo o roteador.

Nesta situação o roteador atribui um endereço IP único para o ESP32, assim qualquer dispositivo com o recurso *Wi-Fi* que tenha posse deste endereço IP poderá estabelecer uma conexão com este microcontrolador e com o diferencial de ser possível acessar a *internet* tanto por parte do ESP32 como pelo dispositivo conectado.

Contudo, como o ponto de acesso deixa de ser o ESP32, apenas os dispositivos conectados ao roteador poderão acessar o microcontrolador e assim como no modo anterior a persistência da conexão irá depender do alcance do sinal do roteador utilizado.

Na Figura 65 é apresentado um exemplo de esquema de operação do microcontrolador em modo STA.

Figura 65 - Modo estação ESP32



Fonte: Elaborada pelo autor

O modo APSTA – *Access Point and Station* é a configuração que permite ao ESP32 operar como um ponto de acesso e como uma estação simultaneamente.

Por último o modo NULL ou OFF é uma configuração onde os modos STA - *Station* e AP - *Access Point* não são inicializados.

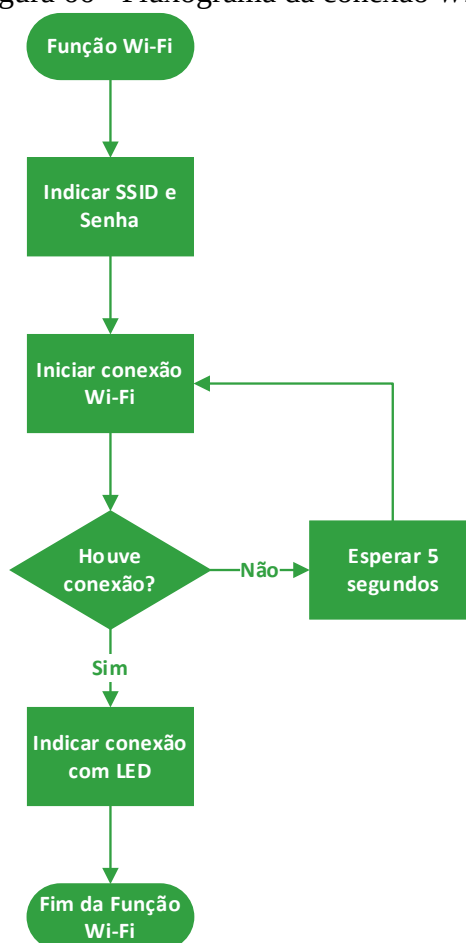
Para utilizar o recurso *Wi-Fi* no microcontrolador ESP32, existe uma biblioteca de nomenclatura *WiFi* que facilita o gerenciamento desta conexão.

Ao criar um projeto utilizando a extensão *PlatformIO* junto ao *Visual Studio Code* esta biblioteca já se encontra, por padrão, instalada.

Para o projeto será adotado o modo estação para o microcontrolador. Assim, será possível transmitir os dados provenientes dos sensores que estão conectados ao ESP32 além de ser possível a conexão com a *internet* tanto pelo microcontrolador como por qualquer outro dispositivo conectado a ele.

O fluxograma apresentado na Figura 66 demonstra a rotina para a conexão *Wi-Fi* no microcontrolador ESP32.

Figura 66 - Fluxograma da conexão *Wi-Fi*



Fonte: Elaborada pelo autor

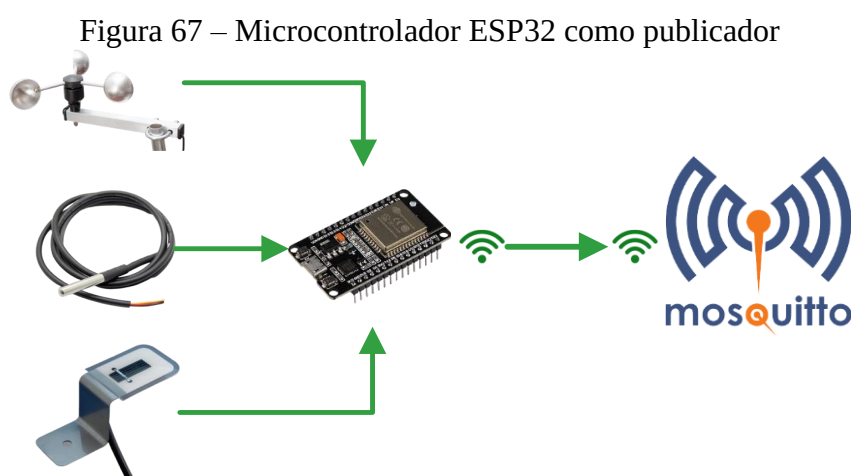
A implementação em código-fonte da biblioteca *WiFi* no microcontrolador ESP32 pode ser encontrada no apêndice 4.

5.6 CONFIGURAÇÕES DO *BROKER*

5.6.1 *Broker* no microcontrolador ESP32

O ESP32 será o publicador de mensagens de três dados distintos, a velocidade do vento em metros por segundo, a temperatura em graus celsius e a irradiância solar em watts por metro quadrado, sendo providos respectivamente pelo anemômetro de copos, sensor de temperatura DS18B20 e sensor de irradiância solar Spektron 210.

Na Figura 67 é demonstrado o ESP32 como um publicador.



Fonte: Elaborada pelo autor

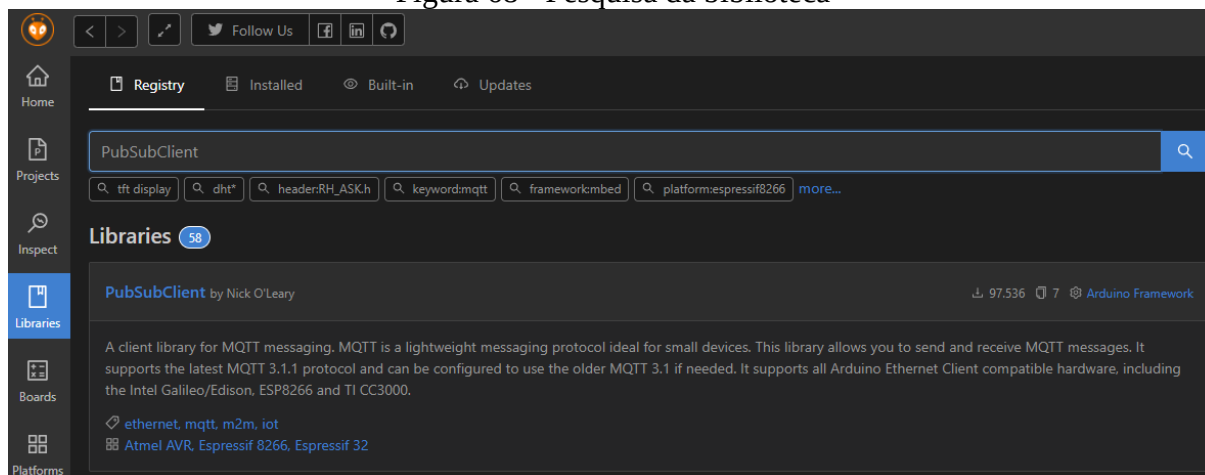
Cada uma das medições será publicada em tópicos separados criados no *broker Mosquitto*.

Para a conexão com o *broker* primeiramente é necessário o *download* da biblioteca *PubSubClient* que facilita o processo de conexão do ESP32 com o *broker Mosquitto*.

A instalação segue o mesmo padrão das bibliotecas anteriormente utilizadas. Na página inicial da extensão *PlatformIO* basta acessar o ícone *libraries* ao lado esquerdo e conduzir a pesquisa com o nome da biblioteca *PubSubClient* desenvolvida por Nick O’Leary.

Na Figura 68 é demonstrado a pesquisa para a biblioteca *PubSubClient*.

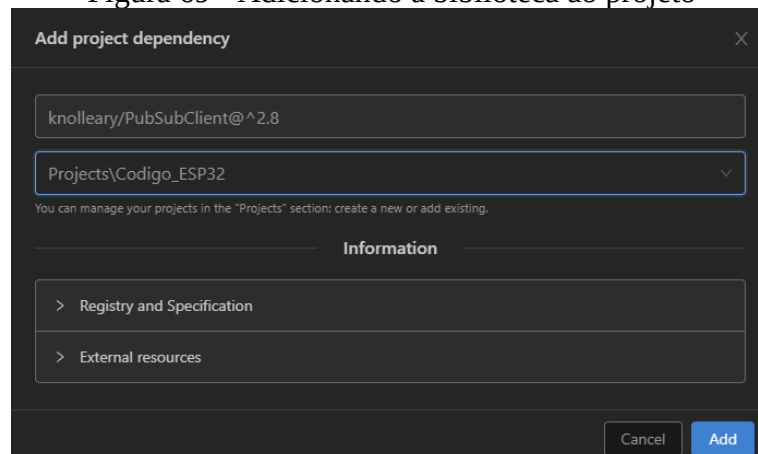
Figura 68 - Pesquisa da biblioteca



Fonte: Visual Studio Code [23] e PlatformIO [24]

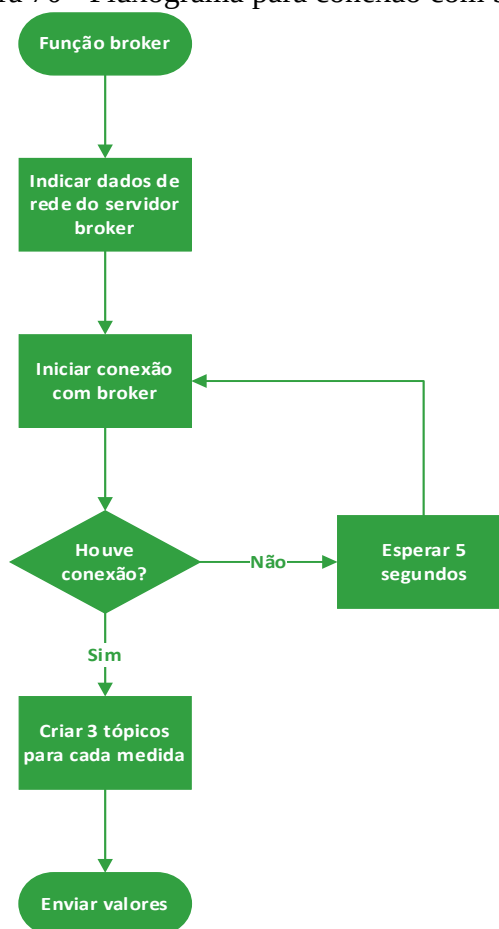
Ao encontrá-la, basta fazer a adição desta biblioteca ao projeto do código-fonte do ESP32 conforme exemplificado na Figura 69.

Figura 69 - Adicionando a biblioteca ao projeto



Fonte: Visual Studio Code [23] e PlatformIO [24]

O fluxograma apresentado na Figura 70 demonstra a rotina executada pelo microcontrolador ESP32 para conexão com o *broker Mosquitto*.

Figura 70 - Fluxograma para conexão com *broker*

Fonte: Elaborada pelo autor

O código-fonte criado no microcontrolador ESP32 para a conexão do *broker Mosquitto* em execução no *Raspberry Pi 3B* pode ser encontrado no apêndice 5.

5.6.2 *Broker no Raspberry Pi 3B*

Com o microcontrolador ESP32 configurado como um dispositivo publicador de mensagens, o próximo passo é preparar o *broker Mosquitto* para o recebimento e gerenciamento destas mensagens.

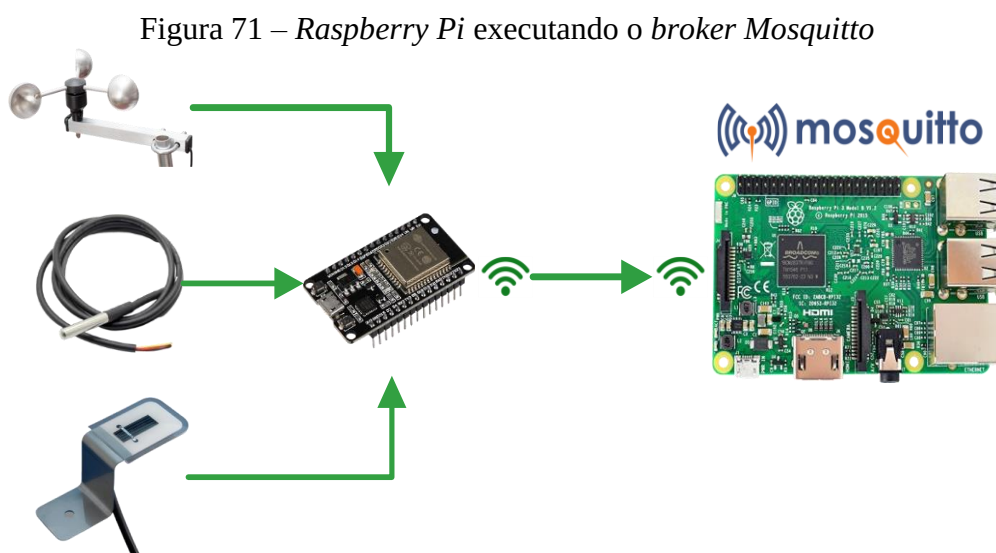
Outra possibilidade para a execução do *broker* é a utilização de um computador do cotidiano, mas que, no entanto, seria uma opção mais dispendiosa e de maior custo para mantê-lo ligado 24 horas por dia e 7 dias por semana, já que o objetivo é manter o funcionamento

ininterrupto, a saber que o *broker* apenas permanece ativo enquanto a máquina que o executa também estiver.

Para evitar o uso de um computador de mesa, existem outras opções, como serviços em nuvem ou dispositivos mais econômicos.

Para o caso dos serviços oferecidos em nuvem, há a cobrança de mensalidades, o que fugiria da proposta inicial de um projeto de baixo custo, restando como opção os dispositivo mais econômicos com o *Raspberry Pi*.

Na Figura 71 é apresentado o esquema do *broker Mosquitto* sendo executado no *Raspberry Pi*.

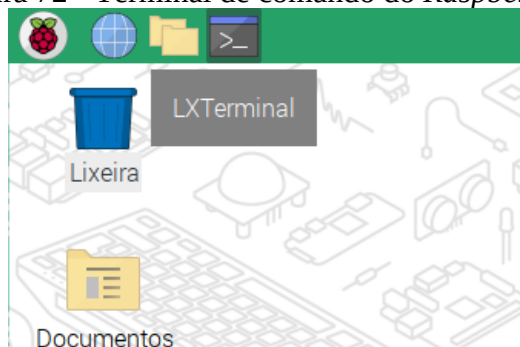


Fonte: Elaborada pelo autor

5.6.3 Instalando o *broker* no *Raspberry Pi* 3B

A instalação do *broker Mosquitto* no *Raspberry Pi* 3B pode ser feita através do acesso de seu terminal de comandos, no sistema operacional *Raspbian*, oficial da própria empresa *Raspberry Pi*.

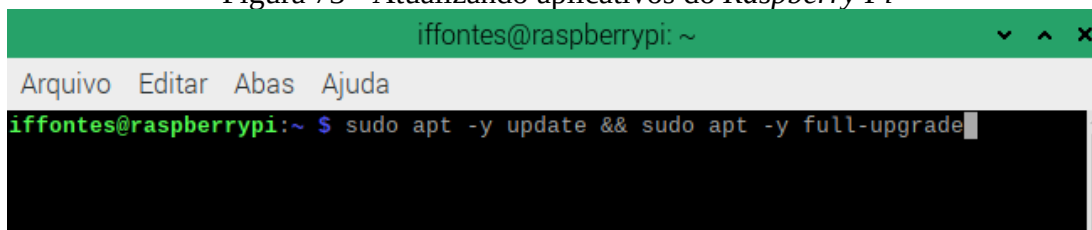
O terminal pode ser acessado pela barra de tarefas do *Raspberry Pi* conforme demonstrado na Figura 72.

Figura 72 - Terminal de comando do *Raspberry Pi*

Fonte: Elaborada pelo autor

Antes de fazer a instalação do *broker Mosquitto* é aconselhável checar se o *Raspberry Pi* precisa realizar alguma atualização de seus aplicativos.

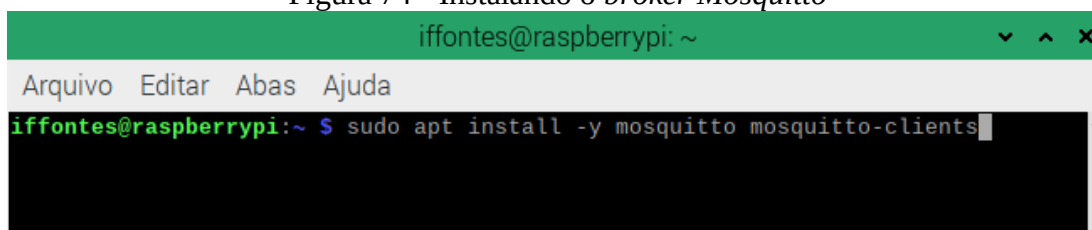
Para a checagem de atualizações, através do terminal é utilizada a seguinte linha de comando `sudo apt -y update && sudo apt -y full-upgrade`, conforme demonstrado na Figura 73.

Figura 73 - Atualizando aplicativos do *Raspberry Pi*

Fonte: Elaborada pelo autor

Este comando dará início ao procedimento de atualização que pode levar alguns minutos a depender do tamanho da atualização.

Com a atualização concluída a instalação do *broker Mosquitto* é realizada inserindo no terminal do *Raspberry Pi* a linha de comando `sudo apt install -y mosquitto mosquitto-clients`, conforme demonstrado na Figura 74.

Figura 74 - Instalando o *broker Mosquitto*

Fonte: Elaborada pelo autor

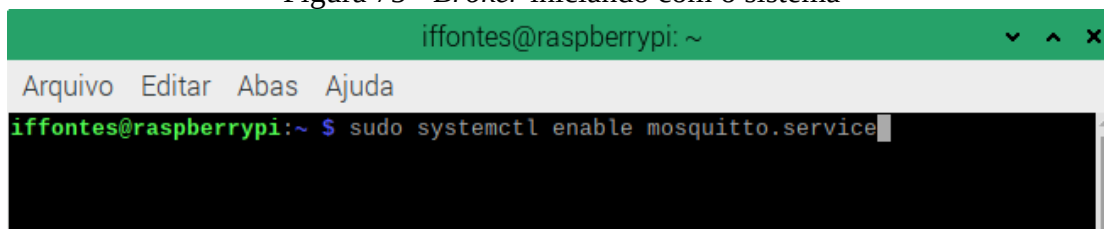
O *broker Mosquitto* será instalado, assim como seu cliente responsável pelo gerenciamento das inscrições e publicações.

Para a próxima etapa por conveniência é interessante permitir a inicialização do *broker* de maneira simultânea ao do sistema operacional *Raspbian*.

Com a inicialização sincronizada, caso ocorra uma interrupção momentânea, como uma queda de energia, o *broker* será automaticamente executado na inicialização do *Raspberry Pi*, removendo a necessidade de uma execução manual e promovendo maior independência para o sistema.

Para tanto, faz-se uso da linha de comando no terminal do *Raspberry Pi* `sudo systemctl enable mosquitto.service`, conforme demonstrado na Figura 75.

Figura 75 - *Broker* iniciando com o sistema



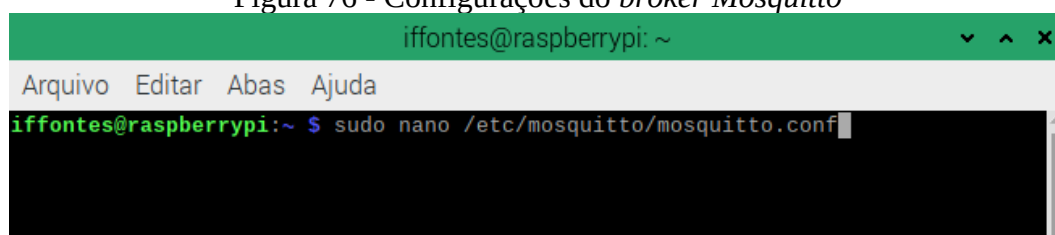
Fonte: Elaborada pelo autor

5.6.4 Configurações de segurança para o *broker*

Com o *broker Mosquitto* instalado e definido para inicialização simultânea ao sistema operacional, ainda é preciso configurá-lo para que seja possível realizar a comunicação com dispositivos externos ao *Raspberry Pi* e restringir o acesso a esta comunicação para somente dispositivos autorizados, como o Microcontrolador EP32.

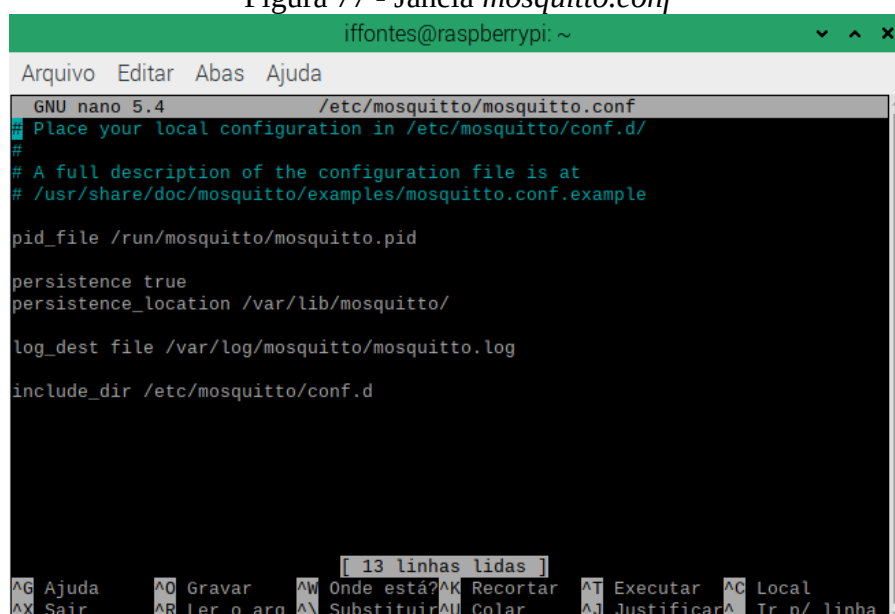
As alterações se fazem necessárias, pois por padrão o *broker* é instalado com configurações que permitem apenas comunicação com dispositivos e aplicações no mesmo local de execução do *broker*.

O acesso as configurações do *broker Mosquitto* é feita pelo terminal através da linha de comando `sudo nano /etc/mosquitto/mosquitto.conf`, conforme demonstrado na Figura 76.

Figura 76 - Configurações do *broker* Mosquitto

Fonte: Elaborada pelo autor

Com o acesso realizado a janela aberta será conforme exibido na Figura 77.

Figura 77 - Janela *mosquitto.conf*

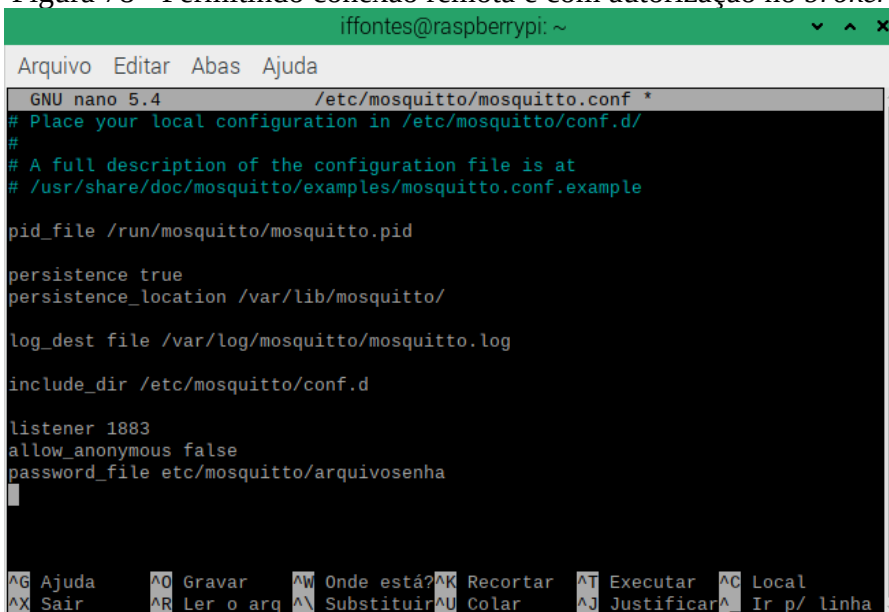
Fonte: Elaborada pelo autor

As instruções podem ser digitadas no final das linhas de configurações. Na primeira linha é digitada a instrução `listener 1883`, para a comunicação pela porta 83.

Na segunda linha é digitada a instrução `allow_anonymous false`, para evitar a conexão de dispositivos não autorizados.

Por fim, na terceira linha é digitado o comando `password_file /etc/mosquitto/arquivosenha`, para indicar o caminho de armazenamento do arquivo denominado “arquivosenha”, este arquivo manterá as credenciais que autorizarão novos dispositivos à serem conectados ao *broker*.

A criação destes arquivos é exemplificada adiante e o terminal ficará conforme demonstrado na Figura 78.

Figura 78 - Permitindo conexão remota e com autorização no *broker*


```

GNU nano 5.4 /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 1883
allow_anonymous false
password_file etc/mosquitto/arquivosenha

```

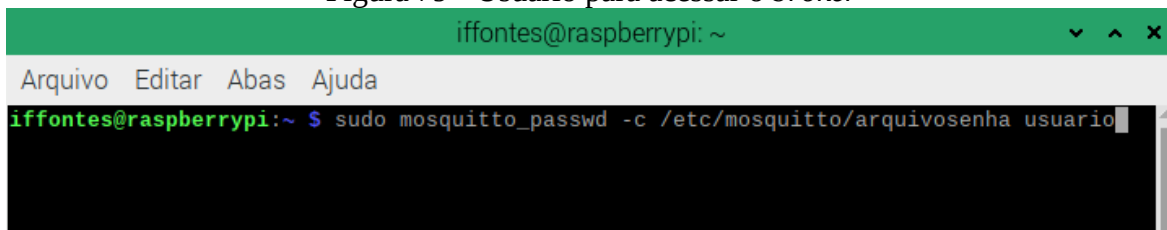
Fonte: Elaborada pelo autor

As novas instruções são salvas com o comando CTRL+X. Com as configurações encerradas o *Raspberry Pi* é reiniciado para que todas as alterações sejam devidamente aplicadas.

Em seguida já é possível definir uma senha e usuário como requisito de acesso ao *broker Mosquitto*, este processo é interessante e conveniente pois assim evita que dispositivos anônimos não autorizados e que estejam no raio de alcance do *Raspberry Pi* publiquem qualquer dado que não esteja de acordo com o projeto em operação.

A criação de um arquivo com usuário e senha é realizada através do comando `sudo mosquitto_passwd -c /etc/mosquitto/arquivosenha usuario`, a parte “usuario” indica o nome de usuário desejado. Esta etapa é demonstrada na Figura 79.

Com o uso deste comando será solicitado uma entrada de senha na linha denominada `Password:`, a senha escolhida é confirmada em `Reenter password:`, e por padrão as senhas não são visíveis quando digitadas no terminal. A tela final é demonstrada na Figura 80.

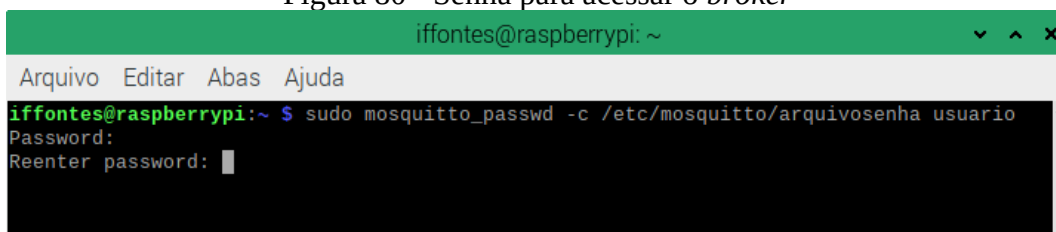
Figura 79 - Usuário para acessar o *broker*


```

iffontes@raspberrypi:~$ sudo mosquitto_passwd -c /etc/mosquitto/arquivosenha usuario

```

Fonte: Elaborada pelo autor

Figura 80 - Senha para acessar o *broker*

```
iffontes@raspberrypi: ~  
Arquivo Editar Abas Ajuda  
iffontes@raspberrypi:~ $ sudo mosquitto_passwd -c /etc/mosquitto/arquivosenha usuario  
Password:  
Reenter password: █
```

Fonte: Elaborada pelo autor

Estas informações devem ser guardadas pois neste ponto apenas os dispositivos de posse do usuário e senha criados poderão se conectar ao *broker*.

Outras credências podem ser criadas atribuindo permissões diferentes a respeito do que os dispositivos podem fazer no *broker*, como apenas publicar ou apenas assinar tópicos, já para o método anterior serão permitidas tanto publicações quanto assinaturas.

5.6.5 Configurações de rede para o *broker*

A próxima etapa consiste na preparação de algumas configurações de interface de rede para uso do *broker Mosquitto*, em outras palavras será definido um IP fixo para a identificação do *Raspberry Pi* na rede local.

A vantagem de se possuir um IP fixo para o *Raspberry Pi* é poder acessá-lo sempre com este mesmo número de IP, seja pelo microcontrolador ESP32 ou por qualquer outro dispositivo que necessite se conectar ao *Raspberry Pi*.

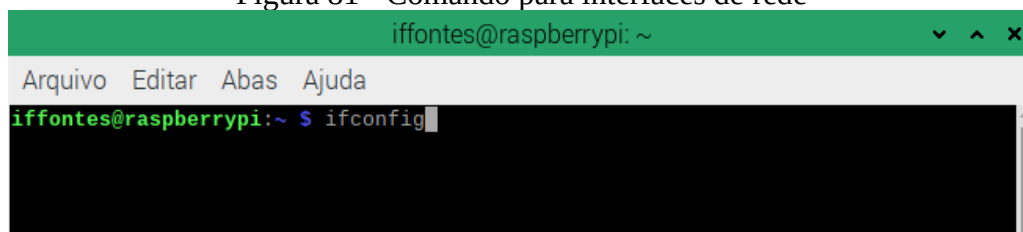
Para a situação de um IP dinâmico será necessário editar o código do microcontrolador ou de qualquer outro dispositivo sempre que o IP do *Raspberry Pi* mudar do antigo para um novo, não sendo uma situação prática.

Quando o *Raspberry Pi* é conectado à *internet* por meio de um roteador, um número de endereço IP passa a ser atribuído ao *Raspberry Pi* pelo próprio roteador.

Este endereço IP pode ser localizado através do terminal do *Raspberry Pi* utilizando o comando `ifconfig`, que exibirá as interfaces de redes deste dispositivo.

Na Figura 81 é demonstrado o comando para acesso as interfaces de rede.

Figura 81 - Comando para interfaces de rede



```

iffontes@raspberrypi: ~
Arquivo Editar Abas Ajuda
iffontes@raspberrypi:~ $ ifconfig

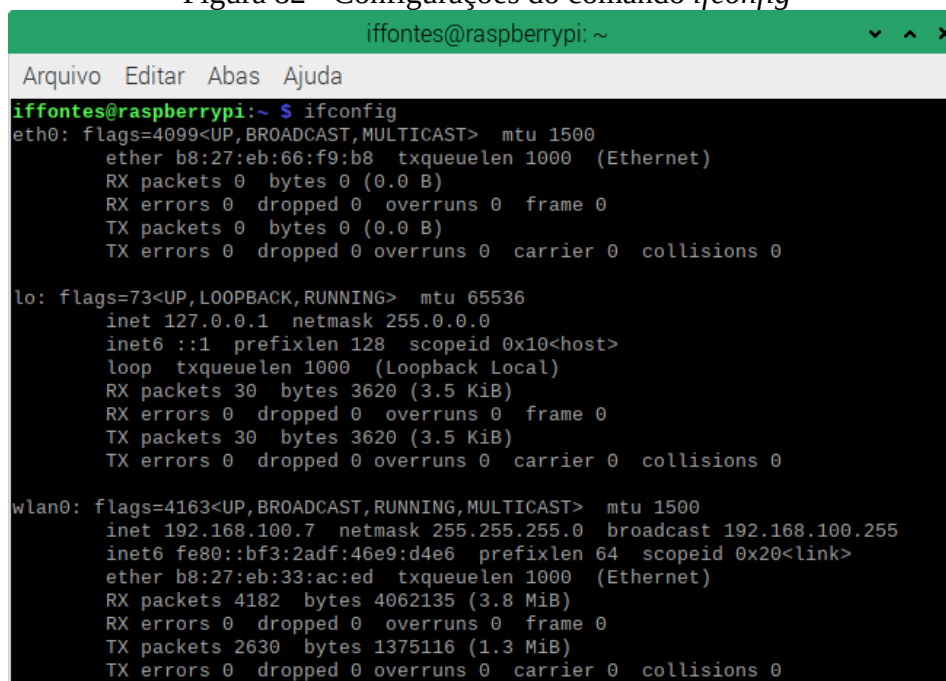
```

Fonte: Elaborada pelo autor

Normalmente são exibidas três configurações, *eth0* quando se está utilizando um cabo de rede para conexão com *internet*, *lo* para o *loopback* e *wlan0* quando o dispositivo está conectado à *internet* através de uma conexão *Wi-Fi*.

O endereço IP será exibido no método utilizado para a conexão com a *internet*.

As configurações podem ser visualizadas na Figura 82.

Figura 82 - Configurações do comando *ifconfig*


```

iffontes@raspberrypi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether b8:27:eb:66:f9:b8 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Loopback Local)
    RX packets 30 bytes 3620 (3.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 30 bytes 3620 (3.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.7 netmask 255.255.255.0 broadcast 192.168.100.255
    inet6 fe80::bf3:2adf:46e9:d4e6 prefixlen 64 scopeid 0x20<link>
    ether b8:27:eb:33:ac:ed txqueuelen 1000 (Ethernet)
    RX packets 4182 bytes 4062135 (3.8 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2630 bytes 1375116 (1.3 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

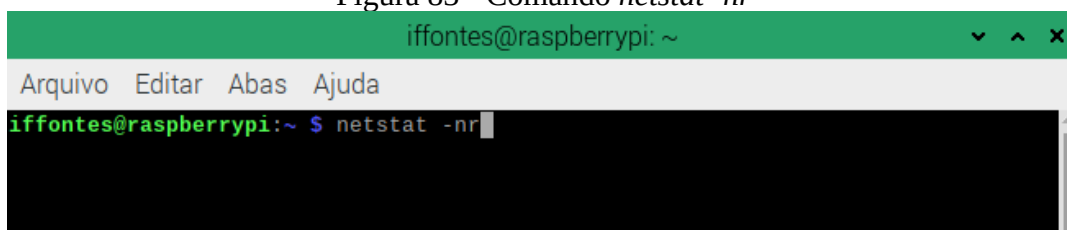
Fonte: Elaborada pelo autor

Neste exemplo o *Raspberry Pi* está conectado à *internet* através de uma conexão *Wi-Fi*, então o endereço IP ficará em exibição na configuração *wlan0*.

O endereço, nesta situação, é identificado como sendo 192.168.100.7 e portando o número para se manter fixo.

Além do endereço IP atribuído ao *Raspberry Pi*, também será necessário ter o conhecimento do endereço IP do próprio roteador, que é encontrado pelo comando `netstat -nr`. Este comando é demonstrado na Figura 83.

Figura 83 - Comando `netstat -nr`



```

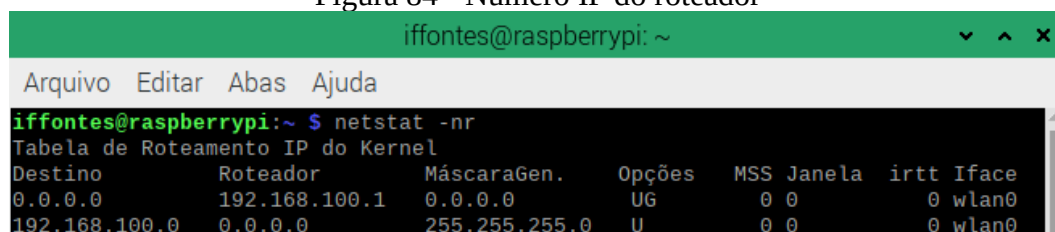
iffontes@raspberrypi: ~
Arquivo Editar Abas Ajuda
iffontes@raspberrypi:~ $ netstat -nr

```

Fonte: Elaborada pelo autor

O endereço está localizado abaixo da nomenclatura “Roteador” conforme demonstrado na Figura 84.

Figura 84 - Número IP do roteador



```

iffontes@raspberrypi: ~
Arquivo Editar Abas Ajuda
iffontes@raspberrypi:~ $ netstat -nr
Tabela de Roteamento IP do Kernel
Destino      Roteador      MáscaraGen.    Opções    MSS Janela    irtt Iface
0.0.0.0      192.168.100.1 0.0.0.0        UG        0 0        0 wlan0
192.168.100.0 0.0.0.0      255.255.255.0 U         0 0        0 wlan0

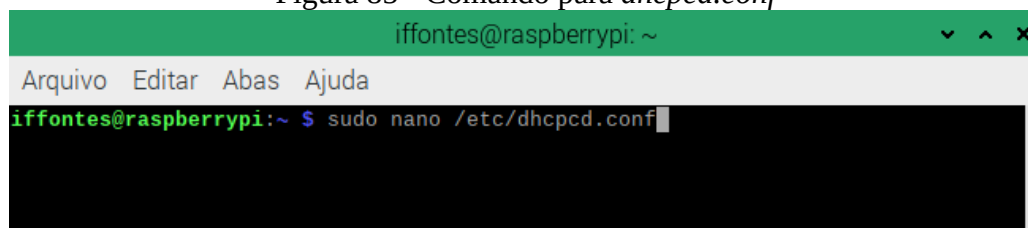
```

Fonte: Elaborada pelo autor

De posse dos dois endereços IP, tanto do *Raspberry Pi* 192.168.100.7 quanto o do roteador 192.168.100.1, a próxima etapa será fixar o IP do *Raspberry Pi* em um arquivo específico.

Este arquivo é encontrado através do comando `sudo nano /etc/dhcpd.conf`, conforme demonstrado na Figura 85.

Figura 85 - Comando para `dhcpd.conf`



```

iffontes@raspberrypi: ~
Arquivo Editar Abas Ajuda
iffontes@raspberrypi:~ $ sudo nano /etc/dhcpd.conf

```

Fonte: Elaborada pelo autor

A nova janela aberta através do comando utilizado anteriormente é demonstrada na Figura 86.

Figura 86 - Janela *dhcpcd.conf*

```

GNU nano 5.4 /etc/dhcpcd.conf
# A sample configuration for dhcpcd.
# See dhcpcd.conf(5) for details.

# Allow users of this group to interact with dhcpcd via the control socket.
#controlgroup wheel

# Inform the DHCP server of our hostname for DDNS.
hostname

# Use the hardware address of the interface for the Client ID.
clientid
# or
# Use the same DUID + IAID as set in DHCPv6 for DHCPv4 ClientID as per RFC4361.
# Some non-RFC compliant DHCP servers do not reply with this set.
# In this case, comment out duid and enable clientid above.
#duid

# Persist interface configuration when dhcpcd exits.
persistent

[ 59 linhas lidas ]
^G Ajuda ^O Gravar ^W Onde está? ^K Recortar ^T Executar ^C Local
^X Sair ^R Ler o arg ^\ Substituir ^U Colar ^J Justificar ^_ Ir p/ linha

```

Fonte: Elaborada pelo autor

A nova entrada de dados é inserida na primeira linha do arquivo aberto, inicialmente é digitado o modo que o *Raspberry Pi* está utilizando para se conectar à *internet*, sendo o comando `interface eth0`, para o cabo de rede ou o comando `interface wlan0`, para a conexão *Wi-Fi*.

Na segunda linha é digitado o IP do *Raspberry Pi* `static ip_address = 192.168.100.7/24`, nesta parte existe o valor extra `/24` que não estava presente na janela anterior, este valor é a notação *CIDR - Classes Inter-Domain Routing* do número de endereço IP utilizado pelo *Raspberry Pi*.

Na terceira linha o endereço IP do roteador é digitado através da linha de instruções `static routers = 192.168.100.1`.

Na quarta linha é digitado as instruções para os endereços DNS `domain_name_servers = 192.168.100.1 8.8.8.8 8.8.4.4`, estes endereços são respectivamente os endereços DNS do roteador e outros dois externos da *Google*, sendo estes externos não obrigatórios.

Com as novas inserções concluídas a tela de configuração ficará conforme demonstrado na Figura 87, com exceção dos números IP que variam para cada indivíduo e local.

Figura 87 - Configuração para IP estático

```

GNU nano 5.4 /etc/dhcpd.conf *
interface wlan0
static ip_address = 192.168.100.7/24
static routers = 192.168.100.1
domain_name_servers = 192.168.100.1 8.8.8.8 8.8.4.4
#
# A sample configuration for dhcpd.
# See dhcpd.conf(5) for details.
#
# Allow users of this group to interact with dhcpd via the control socket.
#controlgroup wheel
#
# Inform the DHCP server of our hostname for DDNS.
hostname
#
# Use the hardware address of the interface for the Client ID.
clientid
# or
# Use the same DUID + IAID as set in DHCPv6 for DHCPv4 ClientID as per RFC4361.
# Some non-RFC compliant DHCP servers do not reply with this set.
# In this case, comment out duid and enable clientid above.

```

Fonte: Elaborada pelo autor

As alterações são confirmadas com o comando CTRL+X. Para garantir a aplicação das alterações o *Raspberry Pi* é reinicializado.

Neste ponto o arquivo estará salvo e o IP do *Raspberry Pi* fixo nas configurações, permanecendo o mesmo até que o usuário queira modificar uma vez mais.

A atribuição de um IP fixo também pode ser feita através do próprio roteador, mas as etapas de configurações podem variar a depender do modelo do roteador utilizado.

5.7 CONFIGURAÇÃO DO *NODE-RED*

5.7.1 *Node-RED* no *Raspberry Pi 3B*

Com o *broker* funcionando e sendo capaz de gerenciar os dados publicados pelo microcontrolador ESP32, passa a ser interessante a elaboração de uma interface gráfica para a exibição dos dados publicados pelo ESP32.

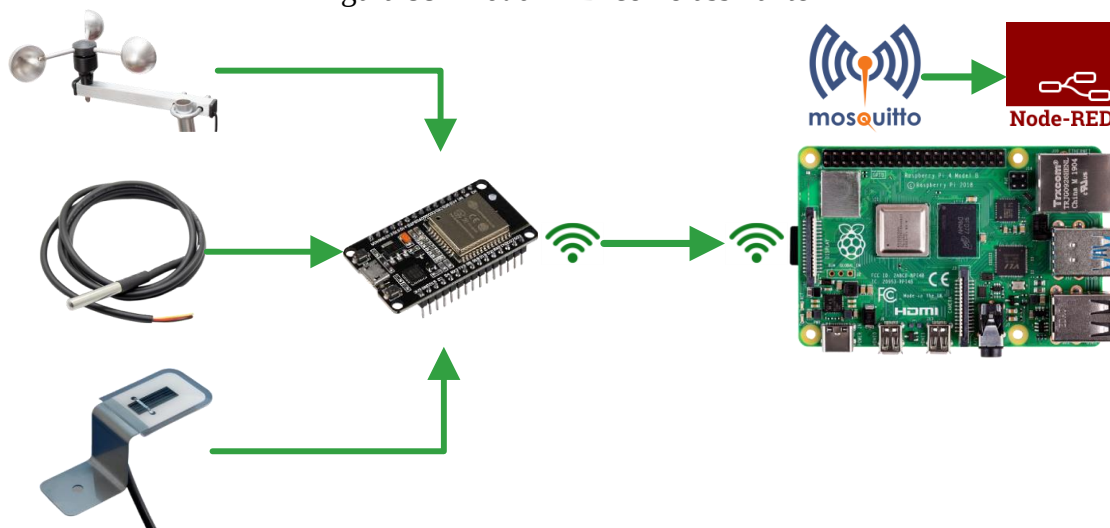
A interface atuará como uma aplicação assinante coletando os dados conforme são publicados pelo microcontrolador, disponibilizando-os em tempo real.

Outra questão interessante é ter uma forma de armazenar toda a informação coletada para consultas.

Uma das várias formas disponíveis de construir estes objetivos citados é utilizando a aplicação *Node-RED*.

O *Node-RED* também é executado no *Raspberry Pi* conforme demonstrado na Figura 88.

Figura 88 - *Node-RED* como assinante

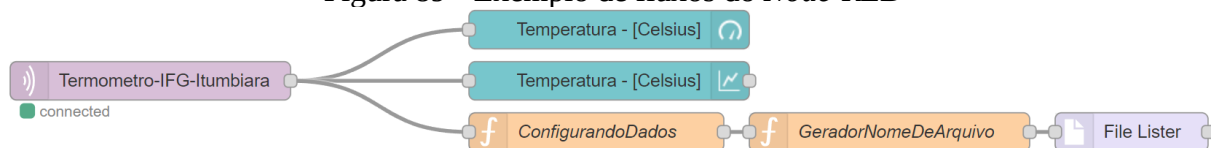


Fonte: Elaborada pelo autor

Este programa é útil para o desenvolvimento de projetos IoT, é uma ferramenta baseada em navegador, possui uma programação de estilo gráfico filtrando consideravelmente a utilização de linhas códigos.

O *Node-Red* utiliza nós ou *nodes* como forma de representar graficamente uma função a ser executada. Quando uma determinada informação passa por este *node*, a função interna é executada fornecendo uma resposta que pode seguir adiante em conformidade com o fluxo elaborado ou encerrando neste mesmo *node*, o fluxo é em resumo o conjunto de *nodes* conectados que formam o caminho por onde os dados passam conforme ilustrado na Figura 89.

Figura 89 - Exemplo de fluxos do *Node-RED*



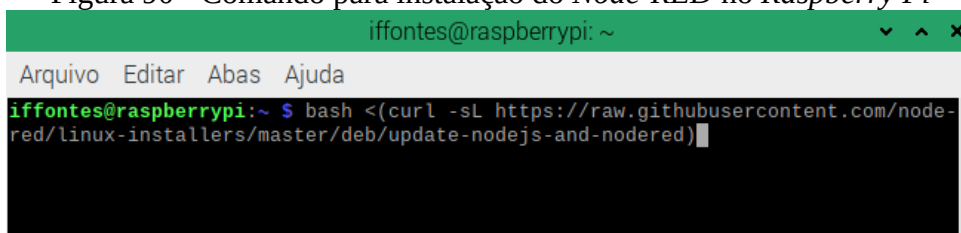
Fonte: Elaborada pelo autor

5.7.2 Instalando o *Node-RED* no *Raspberry Pi*

Para a instalação do *Node-RED* no *Raspberry Pi* primeiro é necessário a verificação de possíveis atualizações para as aplicações instaladas no *Raspberry Pi*, realizada através do comando `sudo apt -y update && sudo apt -y full-upgrade`.

Com as atualizações concluídas, a instalação do *Node-RED* é realizada pelo comando `bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)`, por este comando também será ofertada a instalação de *nodes* específicos para o *Raspberry Pi*, conforme demonstrado na Figura 90.

Figura 90 - Comando para instalação do *Node-RED* no *Raspberry Pi*

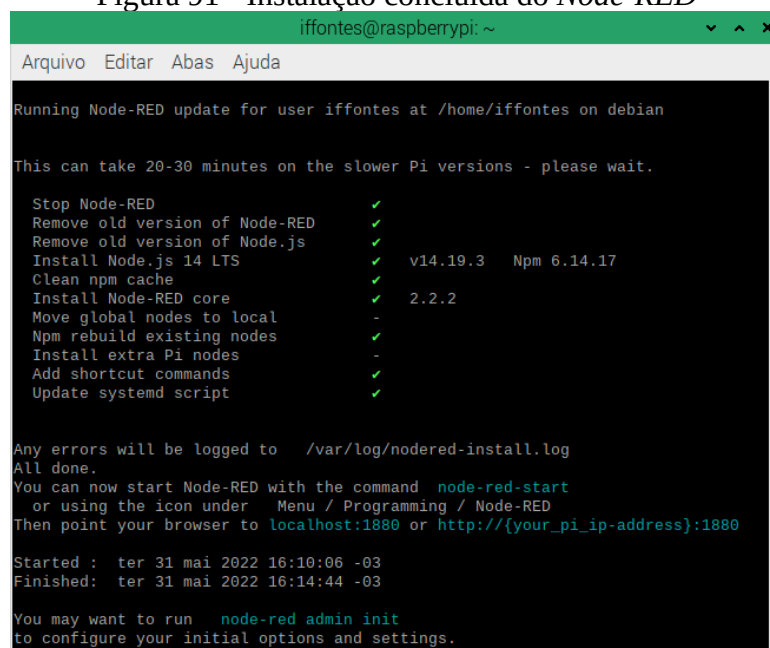


```
iffontes@raspberrypi: ~  
Arquivo Editar Abas Ajuda  
iffontes@raspberrypi:~ $ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

Fonte: Elaborada pelo autor

Com a instalação completa a tela ficará conforme demonstrado na Figura 91.

Figura 91 - Instalação concluída do *Node-RED*

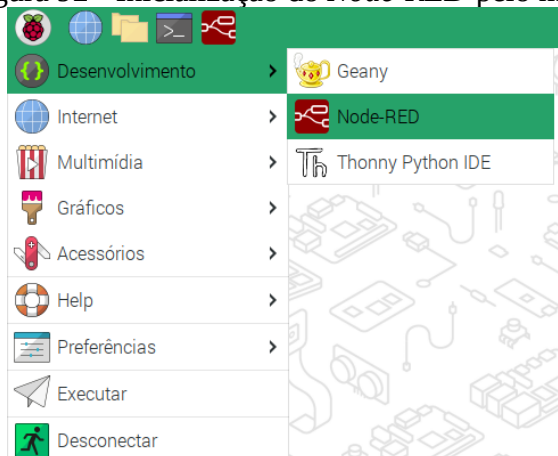


```
iffontes@raspberrypi: ~  
Arquivo Editar Abas Ajuda  
Running Node-RED update for user iffontes at /home/iffontes on debian  
  
This can take 20-30 minutes on the slower Pi versions - please wait.  
  
Stop Node-RED ✓  
Remove old version of Node-RED ✓  
Remove old version of Node.js ✓  
Install Node.js 14 LTS ✓ v14.19.3 Npm 6.14.17  
Clean npm cache ✓  
Install Node-RED core ✓ 2.2.2  
Move global nodes to local -  
Npm rebuild existing nodes ✓  
Install extra Pi nodes -  
Add shortcut commands ✓  
Update systemd script ✓  
  
Any errors will be logged to /var/log/nodered-install.log  
All done.  
You can now start Node-RED with the command node-red-start  
or using the icon under Menu / Programming / Node-RED  
Then point your browser to localhost:1880 or http://{your_pi_ip-address}:1880  
  
Started : ter 31 mai 2022 16:10:06 -03  
Finished: ter 31 mai 2022 16:14:44 -03  
  
You may want to run node-red admin init  
to configure your initial options and settings.
```

Fonte: Elaborada pelo autor

O *Node-RED* pode ser acessado pelo próprio terminal do *Raspberry Pi* com o comando `node-red-start` ou pelo seu ícone disponível no menu desenvolvimento, conforme demonstrado na Figura 92.

Figura 92 - Inicialização do *Node-RED* pelo menu

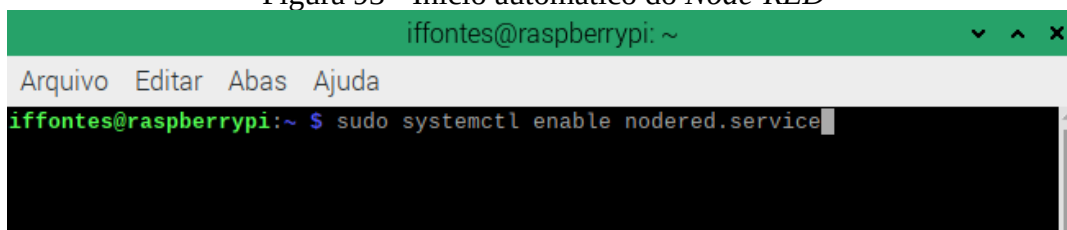


Fonte: Elaborada pelo autor

Nesta etapa é conveniente permitir que o *Node-RED* seja iniciado juntamente ao sistema operacional do *Raspberry Pi* para que no futuro na ocorrência de uma interrupção momentânea, o projeto possa voltar a sua execução com o mínimo de intervenção humana possível.

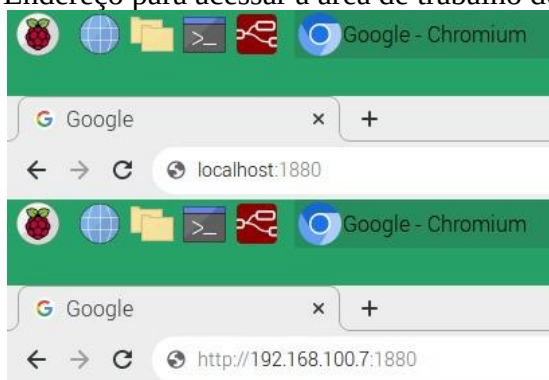
Para esta sincronização é utilizado o comando `sudo systemctl enable nodered.service` conforme demonstrado na Figura 93.

Figura 93 - Início automático do *Node-RED*



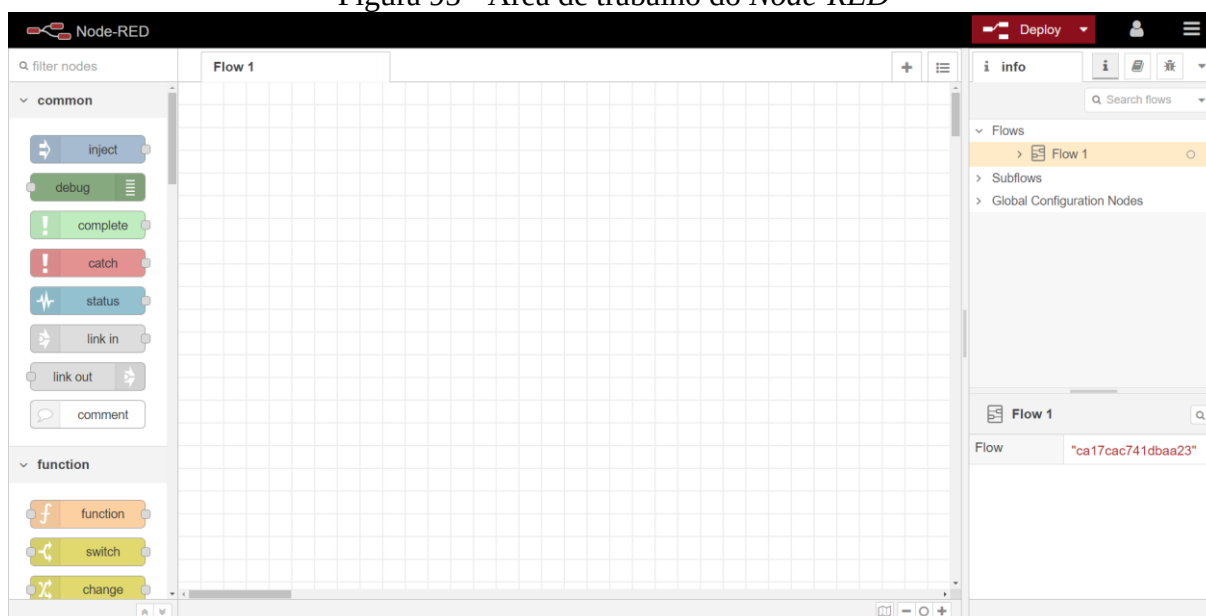
Fonte: Elaborada pelo autor

O acesso ao site de edição do *Node-RED* é realizado através do navegador com a URL `localhost:1880` ou `http://{Enderço-IP-do-Raspberry}:1880`, nesta última opção é utilizado o endereço IP do *Raspberry Pi*, por exemplo, `http://192.168.100.7:1880`, conforme apresentado na Figura 94.

Figura 94 - Endereço para acessar a área de trabalho do *Node-RED*

Fonte: Elaborada pelo autor

Ao acessar o site pelo navegador, a primeira página carregada é a área de trabalho onde os fluxos serão desenvolvidos e configurados para a execução do projeto. A área de trabalho é demonstrada na Figura 95.

Figura 95 - Área de trabalho do *Node-RED*

Fonte: Elaborada pelo autor

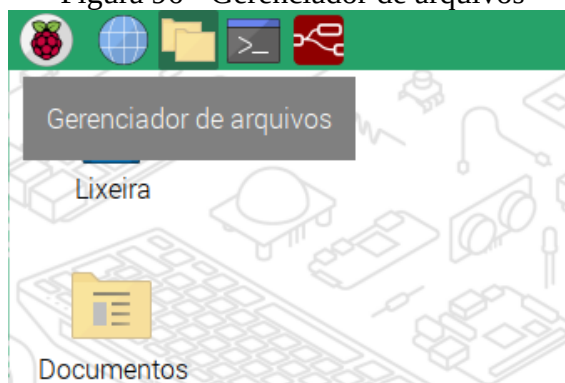
5.7.3 Configurações de segurança no *Node-RED*

Antes de seguir adiante, por questões de segurança, é conveniente estipular um usuário e senha para que apenas pessoas de posse destas credenciais possam fazer edições no projeto em

desenvolvimento no *Node-RED*, pelo contrário qualquer indivíduo conectado no mesmo roteador que o *Raspberry Pi* e com acesso ao *local host* do *Node-RED* poderá acessar a página de edição e alterar o projeto sem restrição alguma.

A definição de um usuário e senha é feita acessando o gerenciador de arquivos do *Raspberry Pi* encontrado em sua barra de tarefas, conforme demonstrado na Figura 96.

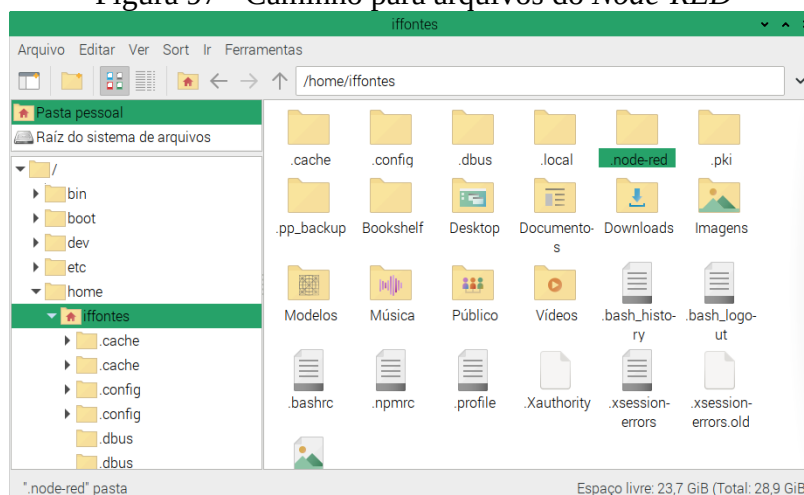
Figura 96 - Gerenciador de arquivos



Fonte: Elaborada pelo autor

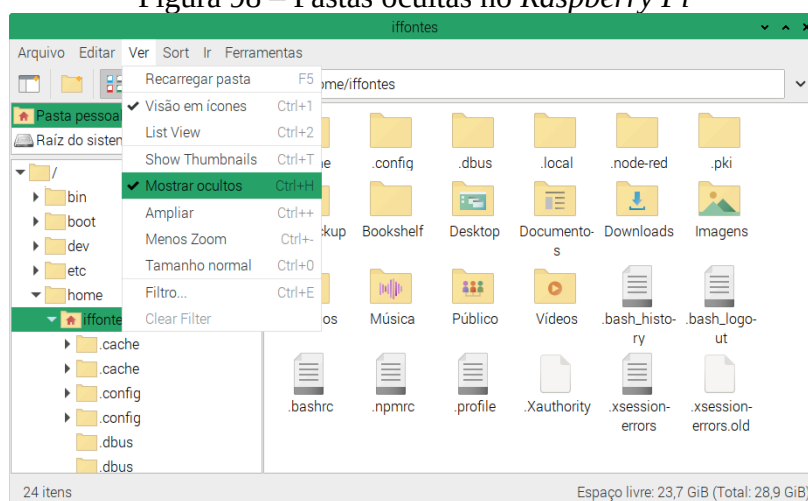
Em seguida é preciso acessar o local de instalação do *Node-RED*, geralmente por padrão encontrado através do caminho `/home/pi/.node-red`, onde “pi” é o nome de usuário do *Raspberry Pi*, conforme demonstrado na Figura 97.

Figura 97 - Caminho para arquivos do *Node-RED*



Fonte: Elaborada pelo autor

Na possibilidade de a pasta estar oculta faz-se uso do comando `CTRL+H` para a exibição ou o uso do menu “Ver” selecionando a opção “Mostrar ocultos” conforme demonstrado na Figura 98.

Figura 98 – Pastas ocultas no *Raspberry Pi*

Fonte: Elaborada pelo autor

O arquivo *settings.js* é aberto para a inclusão de usuário e senha, a ser feito no local referenciado pelo nome *adminAuth*, para que as mudanças sejam reconhecidas os comentários são removidos, conforme demonstrado na Figura 99.

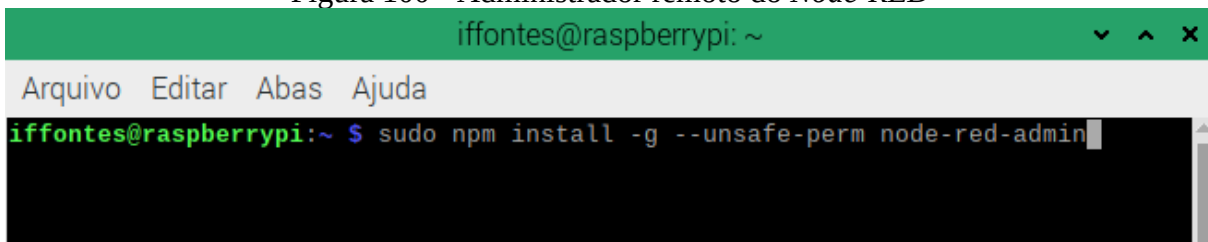
Figura 99 - Local para inserção de usuário e senha do *Node-RED*

```
adminAuth: {
  type: "credentials",
  users: [{
    username: "admin",
    password: "$2a$08$zZWtXTja0fB1pzD4sHcMyOCMyz2Z6dNbM6t18sJogENOMcxwV9DN.",
    permissions: ""
  }],
},
```

Fonte: Elaborada pelo autor

Em username o nome “*admin*” entre aspas é substituído pelo nome de usuário desejado, em seguida abaixo em *password* se encontra um *hash* que representa a criptografia da senha padrão, neste caso a *hash* presente é para a senha “*password*”.

Para a inserção de uma nova senha primeiro é instalada uma ferramenta para administração do *Node-RED*, através do comando `sudo npm install -g --unsafe-perm node-red-admin` digitado no terminal do *Raspberry Pi*, conforme demonstrado na Figura 100.

Figura 100 - Administrador remoto do *Node-RED*


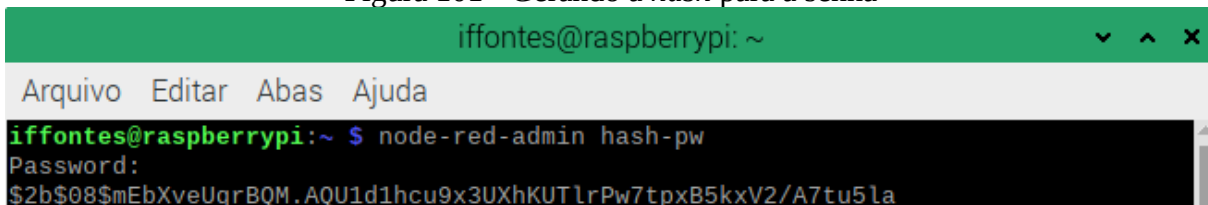
```

iffontes@raspberrypi: ~
Arquivo Editar Abas Ajuda
iffontes@raspberrypi:~ $ sudo npm install -g --unsafe-perm node-red-admin

```

Fonte: Elaborada pelo autor

Com o administrador instalado, o *hash* para a nova senha é disponibilizado utilizando o comando `node-red-admin hash-pw` seguido da inserção da senha desejada conforme ilustrado na Figura 101.

Figura 101 - Gerando a *hash* para a senha


```

iffontes@raspberrypi: ~
Arquivo Editar Abas Ajuda
iffontes@raspberrypi:~ $ node-red-admin hash-pw
Password:
$2b$08$mebXveUqrBQM.AQU1d1hcu9x3UXhKUTlrPw7tpxB5kxV2/A7tu5la

```

Fonte: Elaborada pelo autor

O *hash* gerado é copiado e utilizado para substituir o antigo localizado no arquivo *settings.js* previamente aberto, como exemplo, na Figura 102 é ilustrado como as linhas de código ficaram depois da edição para um usuário de nome “usuariodesejado” e uma *hash* para a senha “senhadesejada”.

Figura 102 – Novas credenciais para um usuário

```

adminAuth: {
  type: "credentials",
  users: [{
    username: "usuariodesejado",
    password: "$2b$08$mebXveUqrBQM.AQU1d1hcu9x3UXhKUTlrPw7tpxB5kxV2/A7tu5la",
    permissions: "*"
  }]
},

```

Fonte: Elaborada pelo autor

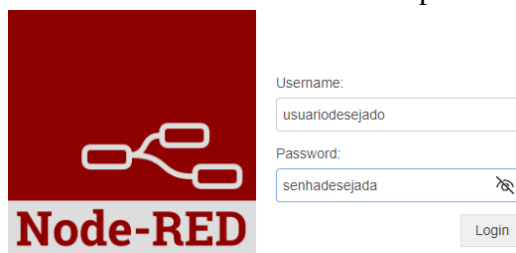
No mesmo arquivo *settings.js* também é possível criar outras credências diferentes desta última, além da possibilidade de atribuir novas permissões, no exemplo acima o usuário e senha possuem o caractere “*” indicando acesso completo para as edições no *Node-RED*.

Com as configurações encerradas o arquivo *settings.js* é salvo e a aplicação *Node-RED* reiniciada.

Deste modo no próximo acesso a página de desenvolvimento do *Node-RED* será apresentada a nova interface exigindo os nomes de usuário e senha criados anteriormente, evitando que pessoas não autorizadas alterem os projetos desenvolvidos no *Node-RED*.

O acesso a página de edição do *Node-RED* é feito inserindo a senha e usuário conforme ilustrado na Figura 103.

Figura 103 – Acessando o *Node-RED* por credencias



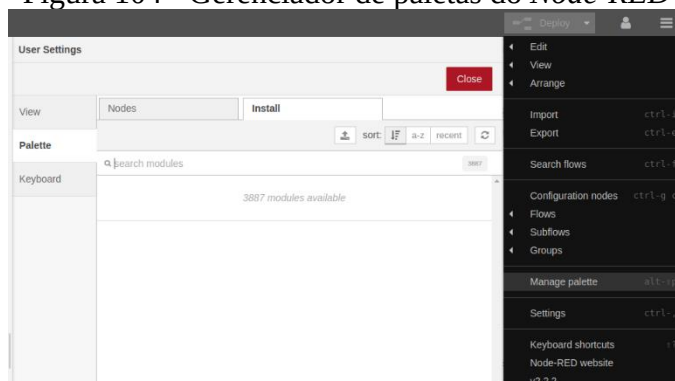
Fonte: Elaborada pelo autor

5.7.4 Instalação de novos *nodes* para o *Node-RED*

Acessando a área de desenvolvimento é observável que o *Node-RED* disponibiliza um conjunto de *nodes* padrões que podem ser encontrados na paleta ao lado esquerdo da área de trabalho, no entanto novos *nodes* podem ser obtidos pelo próprio *Node-RED* em sua tela inicial ou pelo terminal do *Raspberry Pi*.

Para o projeto *nodes* adicionais serão necessários além dos já disponíveis, para instalar novos *nodes* pela área de trabalho, é preciso utilizar o *manage palette* ou gerenciador de paletas acessível na guia presente no canto superior direito, conforme ilustrado na Figura 104.

Figura 104 - Gerenciador de paletas do *Node-RED*



Fonte: Elaborada pelo autor

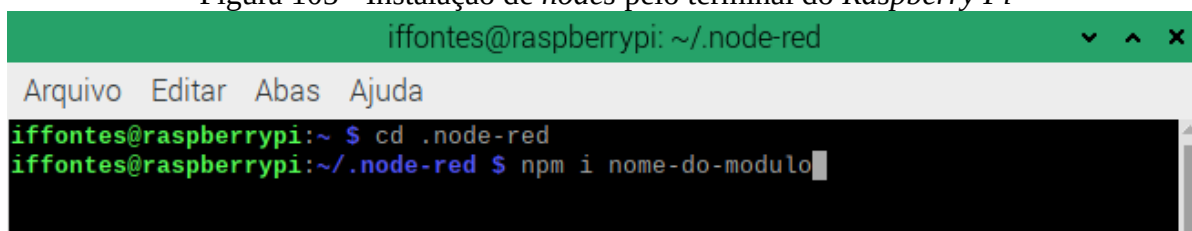
Ao clicar no botão *install* ou instalar uma busca pode ser feita pelo nome de um módulo que irá prover um conjunto de *nodes* em específico.

A instalação dos *nodes* pelo terminal do *Raspberry Pi* é realizada utilizando a linha de comando `npm i nome-do-modulo` substituindo “nome-do-modulo” pelo nome em específico daquele desejado.

No entanto, este comando deve ser executada no diretório do *Node-RED*, geralmente por padrão localizado em `.node-red`, o acesso deste diretório é feito através da linha de comando `cd .node-red` digitada no terminal, conforme ilustrado na Figura 105.

O aplicativo *Node-RED* é reiniciado ao término da instalação para a aplicação das mudanças.

Figura 105 - Instalação de *nodes* pelo terminal do *Raspberry Pi*

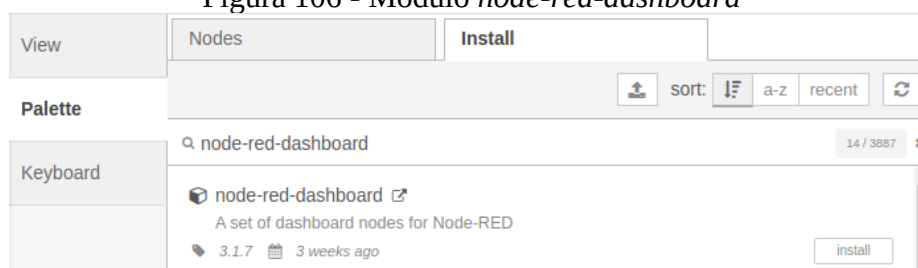


Fonte: Elaborada pelo autor

O primeiro módulo a ser instalado será o *dashboard* que possui um conjunto de *nodes* necessários para a elaboração da interface gráfica e apresentação dos dados recebidos do *broker Mosquitto*.

Utilizando o método do gerenciador de paletas uma busca é feita pelo módulo *node-red-dashboard*, utilizando o botão *install* para a instalação, conforme demonstrado na Figura 106.

Figura 106 - Módulo *node-red-dashboard*



Fonte: Elaborada pelo autor

A instalação pelo terminal de comando é feita buscando inicialmente o diretório do *Node-RED* utilizando o comando `cd .node-red` seguido do comando `npm install node-red-dashboard`, conforme demonstrado na Figura 107.

Figura 107 - Instalação do *dashboard* pelo terminal

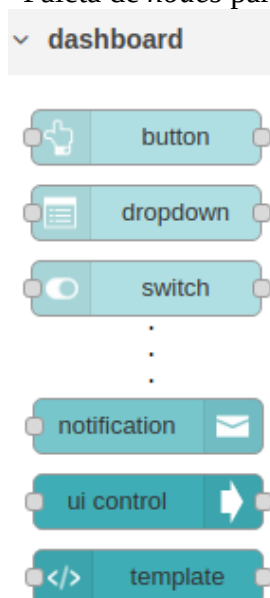
```

iffontes@raspberrypi: ~/.node-red
Arquivo  Editar  Abas  Ajuda
iffontes@raspberrypi:~ $ cd .node-red
iffontes@raspberrypi:~/.node-red $ npm i node-red-dashboard

```

Fonte: Elaborada pelo autor

Com a instalação realizada a nova paleta de *nodes* do *dashboard* irá aparecer a esquerda da área de trabalho junto as demais outras conforme ilustrado na Figura 108.

Figura 108 - Paleta de *nodes* para *dashboard*

Fonte: Elaborada pelo autor

Ainda para a parte gráfica existe um *node* útil para manipulação de imagens a serem inseridas na interface gráfica, a instalação é feita buscando o módulo *node-red-contrib-ui-media* no gerenciador de paletas ou utilizando o comando `npm i node-red-contrib-ui-media` no terminal do *Raspberry Pi* dentro do diretório do *Node-RED*, conforme demonstrado na Figura 109.

Figura 109 – Instalação do *node* para mídia

```

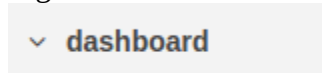
iffontes@raspberrypi: ~/.node-red
Arquivo  Editar  Abas  Ajuda
iffontes@raspberrypi:~ $ cd .node-red
iffontes@raspberrypi:~/.node-red $ npm i node-red-contrib-ui-media

```

Fonte: Elaborada pelo autor

Após a instalação e reinicialização do *Node-RED* o novo *node* aparecerá junto aos demais na paleta do *dashboard*, conforme ilustrado na Figura 110.

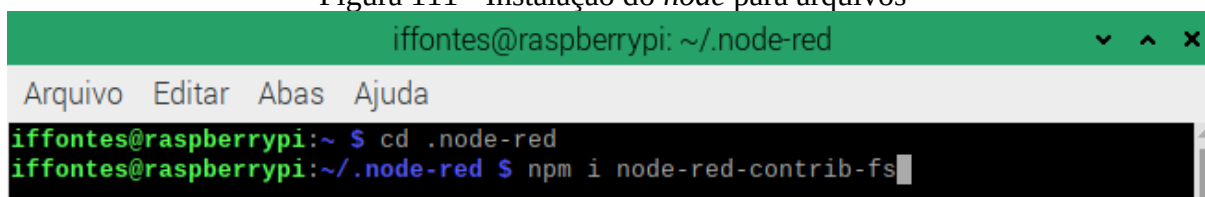
Figura 110 - *Node media*



Fonte: Elaborada pelo autor

A instalação do último *node*, necessário para a etapa de armazenamento e disponibilização de arquivos do projeto, é feita buscando pelo módulo *node-red-contrib-fs* no gerenciador de paletas ou utilizando o comando `npm i node-red-contrib-fs` no terminal do *Raspberry Pi* no diretório do *Node-RED*, conforme demonstrado na Figura 111.

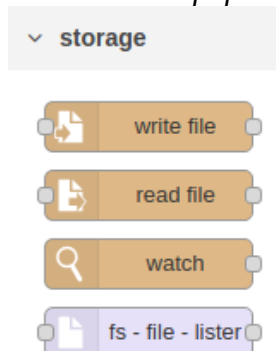
Figura 111 - Instalação do *node* para arquivos



Fonte: Elaborada pelo autor

O novo *node* é localizado com o nome *fs-file-lister* na paleta *storage* do *Node-RED* junto a outros *nodes* para arquivos, disponível após reinicialização da aplicação *Node-RED*, conforme demonstrado na Figura 112.

Figura 112 - *Node fs-file-lister*



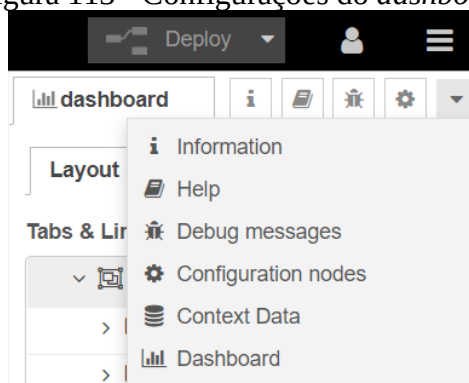
Fonte: Elaborada pelo autor

5.7.5 Manipulações e opções do *dashboard*

Antes de desenvolver a interface gráfica algumas noções de como o *dashboard* do *Node-RED* funciona devem ser repassadas.

Primeiramente o gerenciamento do *dashboard* é feito na janela ao lado direito da área de trabalho, clicando na seta para mais opções é possível encontrar o ícone do *dashboard*, ao selecioná-lo as configurações para a interface gráfica aparecerão revelando o local para as edições da aparência da interface que será exibida. Este ícone é demonstrado na Figura 113.

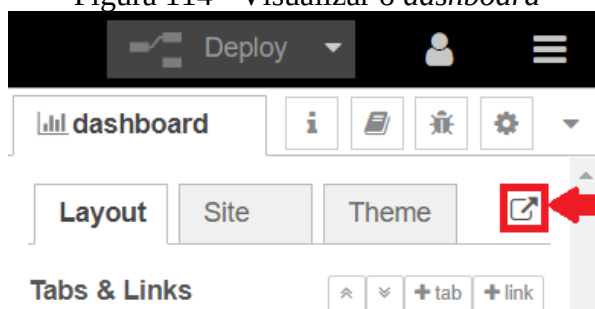
Figura 113 - Configurações do *dashboard*



Fonte: Elaborada pelo autor

Para visualizar a página basta clicar no quadrado com a seta que fica ao lado das opções de menu, conforme indicado na Figura 114.

Figura 114 - Visualizar o *dashboard*

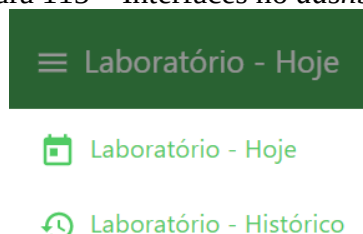


Fonte: Elaborada pelo autor

O *dashboard* oferece a opção para a criação de mais de uma interface gráfica permitindo que o usuário organize melhor seu projeto ao evitar colocar uma quantidade elevada de informações em um único local.

Quando uma interface a mais é criada um guia aparece no canto superior esquerdo localizado no *dashboard* desenvolvido, para a escolha de qual visualizar, conforme ilustrado na Figura 115.

Figura 115 – Interfaces no *dashboard*



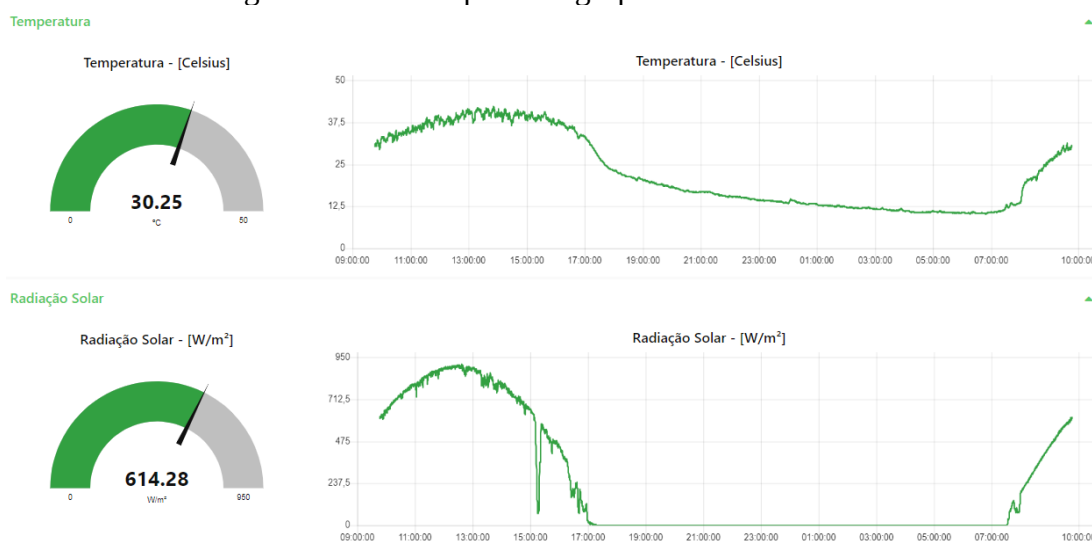
Fonte: Elaborada pelo autor

Para cada interface, grupos podem ser criadas para a melhor organização da página e das informações disponibilizadas, estes grupos são janelas que podem ser configuradas através da área de trabalho.

Existem várias opções de configurações para estes grupos, concedendo liberdade em suas personalizações, as configurações incluem a escolha de suas posições na interface do *dashboard*, seu tamanho para exibição, a escolha de quais informações para se deixar a amostra, configurações de cores, escolha da interface onde estes grupos serão exibidas entre outras opções.

Na Figura 116 é ilustrada um exemplo de interface com dois grupos, um superior utilizado para medições da temperatura e um inferior para as medições da radiação solar.

Figura 116 – Exemplo de 2 grupos em uma interface



Fonte: Elaborada pelo autor

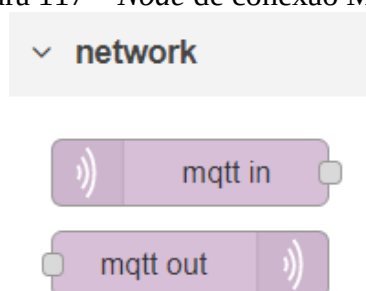
5.7.6 Node de conexão com o broker

Neste ponto o *Node-RED* está preparado para o desenvolvimento dos fluxos para o projeto e execução do *dashboard*. Conforme o que foi configurado em etapas anteriores o *Raspberry Pi* já se encontra executando o *broker Mosquitto* automaticamente em sua inicialização assim como o *Node-RED*.

Como o objetivo é fazer com que o *Node-RED* receba os dados publicados no *broker Mosquitto* pelo microcontrolador ESP32 e os disponibilize na interface gráfica, então o primeiro passo para o desenvolvimento na área de trabalho será fazer a conexão do *Node-RED* com o *broker Mosquitto*.

Para isso o *Node-RED* oferece por padrão os *nodes* necessários para esta conexão, que estão localizados na paleta *network*, o *node* a ser utilizado é identificado como *mqtt in* conforme Figura 117.

Figura 117 – Node de conexão MQTT



Fonte: Elaborada pelo autor

Ao colocar este *node* na área de trabalho as suas configurações podem ser acessadas clicando duas vezes em seu ícone, fazendo isto a janela de edição será exibida para a inserção das propriedades do *broker* a ser conectado.

As informações contidas no *node mqtt in* são apresentadas na Figura 118.

Figura 118 – Propriedades do *node mqtt in*

Fonte: Elaborada pelo autor

O *node mqtt in* contém as entradas, *server*, *topic*, *QoS*, *output* e *name*:

- I. *Server* - A primeira propriedade são as informações de rede do local de execução do *broker Mosquitto*, neste caso o *Raspberry Pi*, que podem ser feitas clicando no botão ao lado, esta parte será detalhada adiante;
- II. *Topic* - A segunda propriedade é o nome exato do tópico a ser assinado, para então obter qualquer informação divulgada nele, neste caso a primeira assinatura será no tópico da temperatura que foi definido como *Temperatura-IFG-Itumbiara* pelo microcontrolador ESP32;
- III. *QoS* - A terceira propriedade é a escolha da qualidade de serviço, será utilizado o valor 2 assim é garantido o recebimento de pelo menos 1 mensagem para cada valor publicado no tópico sem duplicatas;
- IV. *Output* - A quarta propriedade é a forma que a mensagem está sendo disponibilizada, neste caso uma *string*, mas a opção *auto-detect* também pode ser utilizada;
- V. *Name* - Por último existe um espaço apenas para a alteração do nome do *node* para exibição na área de trabalho e, portanto, uma opção de personalização que fica como critério, pode ser útil para a organização e identificação de cada *node* nos fluxos dentro da área de trabalho.

Assim que as propriedades gerais forem colocadas o próximo passo é inserir as configurações do servidor utilizado, isto é, os dados do *Raspberry Pi* que está executando o *broker Mosquitto*.

Conforme apresentado as alterações podem ser feitas clicando no botão com o ícone de um lápis ao lado do submenu *server*. As informações contidas neste submenu são apresentadas na Figura 119.

Figura 119 - Parâmetros do servidor *broker*

The screenshot shows the 'Edit mqtt in node > Edit mqtt-broker node' window. At the top, there are 'Delete', 'Cancel', and 'Update' buttons. Below is a 'Properties' section with a gear icon and a document icon. The 'Name' field is empty. The 'Connection' tab is selected, showing fields for 'Server' (with placeholder 'e.g. localhost'), 'Port' (1883), 'Use TLS' (unchecked), 'Protocol' (MQTT V3.1.1), 'Client ID' (Leave blank for auto generated), 'Keep Alive' (60), and 'Session' (Use clean session checked).

Fonte: Elaborada pelo autor

Nesta janela existem três menus, são eles: *connection*, *security* e *messages*. O menu *connection* é utilizado para os dados de conexão com o servidor, contendo as entradas, *server*, *protocol*, *cliente ID* e *keep alive*:

- I. *Server* – A primeira opção serve para a inserção do nome ou número do *host* e a porta TCP/UDP - *Transmission Control Protocol/ User Datagram Protocol*. O *host* será o IP definido na etapa da configuração de rede do *broker Mosquitto* no *Raspberry Pi*, neste caso 192.168.100.7 e por padrão a porta utilizada será a 1883.
- II. *Protocol* - A segunda opção é a versão do protocolo MQTT utilizado pelo *broker Mosquitto*.
- III. *Client ID* - A terceira opção é o identificador único que auxilia o *broker* a identificar cada dispositivo conectado a ele, podendo ser em sua essência qualquer conjunto de caracteres, neste projeto por exemplo é utilizado NODE-RED-ASSINANTE.
- IV. *Keep Alive* - A opção *Keep Alive* é mantida com o valor padrão de 60 segundos.

As informações preenchidas são apresentadas na Figura 120.

Figura 120 - Menu de conexão com o *broker*

Edit mqtt in node > Edit mqtt-broker node

Delete Cancel Update

⚙ Properties

📁 Name Name

Connection Security Messages

🌐 Server 192.168.100.7 Port 1883

☒ Connect automatically

☐ Use TLS

⚙ Protocol MQTT V3.1.1

📁 Client ID NODE-RED-ASSINANTE

💓 Keep Alive 60

i Session ☒ Use clean session

Fonte: Elaborada pelo autor

Para o menu seguinte de segurança, a senha e usuário a serem utilizadas são as previamente criadas na etapa para a configuração de segurança do *broker*, isto é, as credências que permitem o acesso de dispositivos e aplicações ao *broker Mosquitto*, caso nenhuma credencial tenha sido criada esta opção poderá ser mantida em branco.

A inserção das credencias são demonstradas na Figura 121.

Figura 121 - Menu de segurança com *broker*

Edit mqtt in node > Edit mqtt-broker node

Delete Cancel Update

⚙ Properties

📁 Name Name

Connection Security Messages

👤 Username usuario

🔒 Password senha

Fonte: Elaborada pelo autor

O terceiro e também último menu denominado *messages* para mensagens não requer configurações adicionais.

A aplicação dos dados inseridos é feita clicando no botão vermelho *Update* presente no canto superior direito da janela.

Após a atualização, a janela inicial será reexibida, neste ponto é preciso apertar o botão vermelho *Done*, assim, todas as mudanças serão aplicadas e o *node* estará pronto para se conectar ao *broker* e receber os dados do tópico Temperatura-IFG-Itumbiara.

Sempre que uma nova configuração for feita na área de trabalho do *Node-RED* o botão *Deploy* no canto superior direito ficará vermelho e deverá ser pressionado para aplicação de qualquer mudança realizada, conforme apresentado na Figura 122.

Figura 122 - Botão *Deploy*

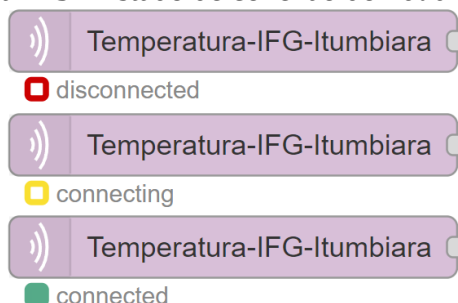


Fonte: Elaborada pelo autor

Ao pressionar *Deploy*, se configurado corretamente, o *node mqtt in* irá estabelecer conexão com o *broker* que pode ser checada visualizando o estado de conexão que aparece abaixo do *node*.

As opções visuais para indício do estado de conexão do *node mqtt in* são apresentadas na Figura 123.

Figura 123 - Estado de conexão do *node mqtt in*

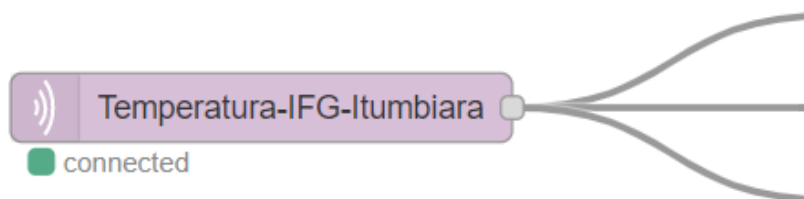


Fonte: Elaborada pelo autor

Se o *node* estiver em estado conectado, significa que ele está inscrito e recebendo os dados publicados no tópico Temperatura-IFG-Itumbiara.

Uma vez que as configurações do *node* é realizada, ela fica gravada para utilização em qualquer outro *node mqtt in* na área de trabalho sem a necessita de repetir o processo de configuração.

Na Figura 124 é demonstrado o *node mqtt in* com conexão estabelecida com o tópico Temperatura-IFG-Itumbiara do *broker Mosquitto*.

Figura 124 - *Node mqtt in* em operação

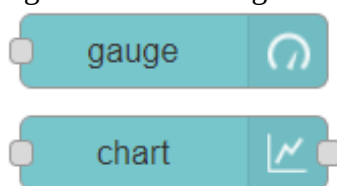
Fonte: Elaborada pelo autor

Com a conexão estabelecida com o um dos tópicos do *broker Mosquitto*, agora é possível trabalhar com os valores que chegam e são disponibilizados pelo *node*.

Note que o *node* assumirá o nome do tópico em que está conectado caso nada tenha sido definido na opção *Name* em suas configurações, isto ajuda a identificar e organizar os fluxos.

5.7.7 *Nodes* para gráficos

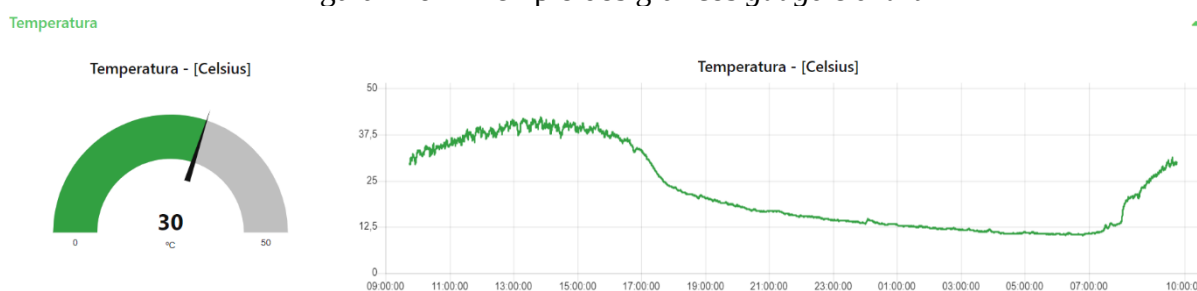
Ao acessar a paleta *dashboard* existem opções de representação visual para implementação na interface como gráficos e medidores com opções de personalização que auxiliam no desenvolvimento de um gráfico para melhor representatividade de uma informação. Os *nodes* gráficos são ilustrados na Figura 125.

Figura 125 - *Nodes* gráficos

Fonte: Elaborada pelo autor

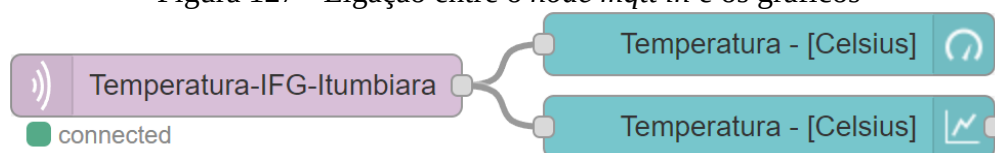
Estes gráficos serão utilizados como forma de exibir os valores fornecidos pelo *node mqtt in* conectado ao *broker*. O projeto utiliza o medidor do *node gauge* para apresentar o último valor recebido e o gráfico do *node chart* para uma representação das mudanças de valores ao longo do tempo.

Um exemplo do visual dos *nodes gauge* e *chart* é demonstrado na Figura 126.

Figura 126 - Exemplo dos gráficos *gauge* e *chart*

Fonte: Elaborada pelo autor

Para formar o fluxo só é preciso arrastar cada *node* para a área de trabalho e fazer a ligação entre o *node mqtt in* de conexão com o *broker* e os *nodes* gráficos conforme apresentado na Figura 127.

Figura 127 - Ligação entre o *node mqtt in* e os gráficos

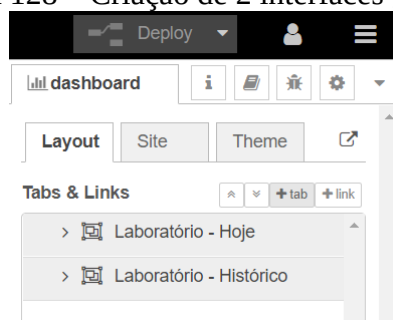
Fonte: Elaborada pelo autor

Como elementos para a interface gráfica foram adicionados é necessário criar a página da interface e os grupos para a exibição destes gráficos na interface.

Primeiramente é preciso acessar as edições do *dashboard* pela janela a direita da área de trabalho, no menu *Layout* uma página pode ser adicionada clicando no botão abaixo denominado “+ tab”.

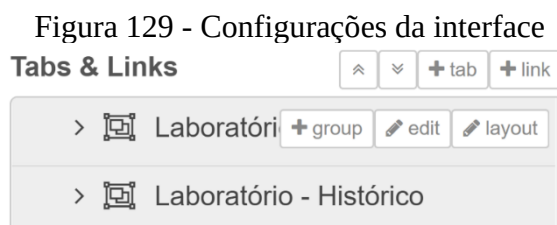
Na Figura 128 é ilustrado a criação de duas interfaces gráficas, a primeira para dados coletados no dia, nomeada como Laboratório - Hoje e a segunda para dados históricos, nomeada Laboratório - Histórico.

Figura 128 – Criação de 2 interfaces gráficas



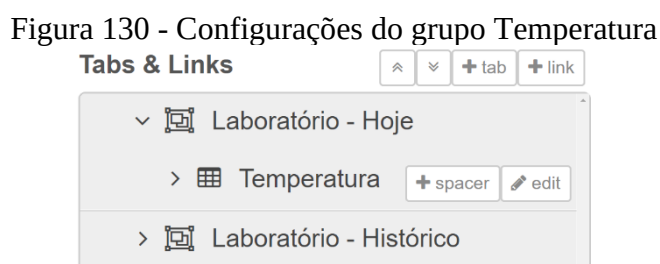
Fonte: Elaborada pelo autor

A interface criada é adicionada automaticamente ao site *dashboard* assim como a aba de edição na área de trabalho, nesse local é possível adicionar os grupos, fazer edição de nomes, personalizações e outros aspectos pertinentes ao visual da interface. O acesso a edições é feito na aba da interface conforme demonstrado na Figura 129.



Fonte: Elaborada pelo autor

O próximo passo é adicionar um grupo que irá conter os gráficos com os valores de temperatura adicionados previamente. O novo grupo também pode ser editado através de sua aba após sua criação conforme demonstrado na Figura 130.



Fonte: Elaborada pelo autor

Com um novo grupo criado através da janela de edição de cada *node* gráfico é possível escolher o grupo que estes gráficos pertencerão, além das edições em suas características.

Na Figura 131 e na Figura 132 são demonstradas as edições feitas nos *node gauge* e *chart* respectivamente, que estão localizados na interface de dados atuais e no grupo denominado Temperatura, criado na etapa anterior.

Figura 131 - Edição do *node gauge*

Edit gauge node

Delete Cancel Done

Properties

Group [Laboratório - Hoje] Temperatura

Size 8 x 6

Type Gauge

Label Temperatura - [Celsius]

Value format {{value}}

Units °C

Range min 0 max 50

Colour gradient

Sectors 0 ... 40 ... 45 ... 50

Class Optional CSS class name(s) for widget

Name

Fonte: Elaborada pelo autor

Entre as edições disponíveis é possível alterar o tamanho tanto da área do grupo bem como a área do gráfico presente neste grupo, contudo, também é possível permitir que o tamanho seja escolhido automaticamente pela própria interface.

Figura 132 – Edição do *node chart*

Edit chart node

Delete Cancel Done

Properties

Group [Laboratório - Hoje] Temperatura

Size 20 x 6

Label Temperatura - [Celsius]

Type Line chart ☐ enlarge points

X-axis last 24 hours OR 1000 points

X-axis Label HH:mm:ss ☐ as UTC

Y-axis min 0 max 50

Legend Show Interpolate linear

Series Colours

Blank label display this text before valid data arrives

Class Optional CSS class name(s) for widget

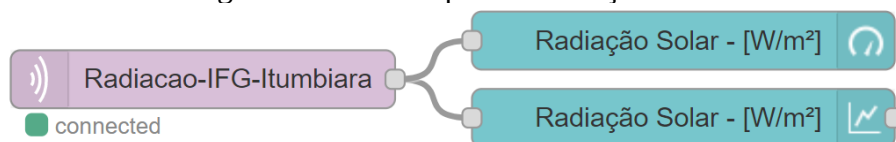
Name Name

Fonte: Elaborada pelo autor

A partir deste ponto a mesma metodologia pode ser aplicada para a inclusão de grupos com os gráficos dos valores da velocidade do vento e da radiação solar.

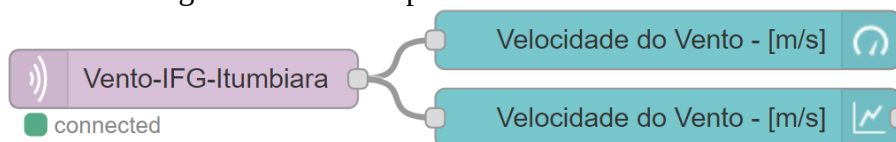
Na Figura 133 e na Figura 134 são demonstrados os *nodes* gráficos para a exibição dos valores da radiação solar e velocidade do vento, respectivamente.

Figura 133 - *Nodes* para a Radiação Solar



Fonte: Elaborada pelo autor

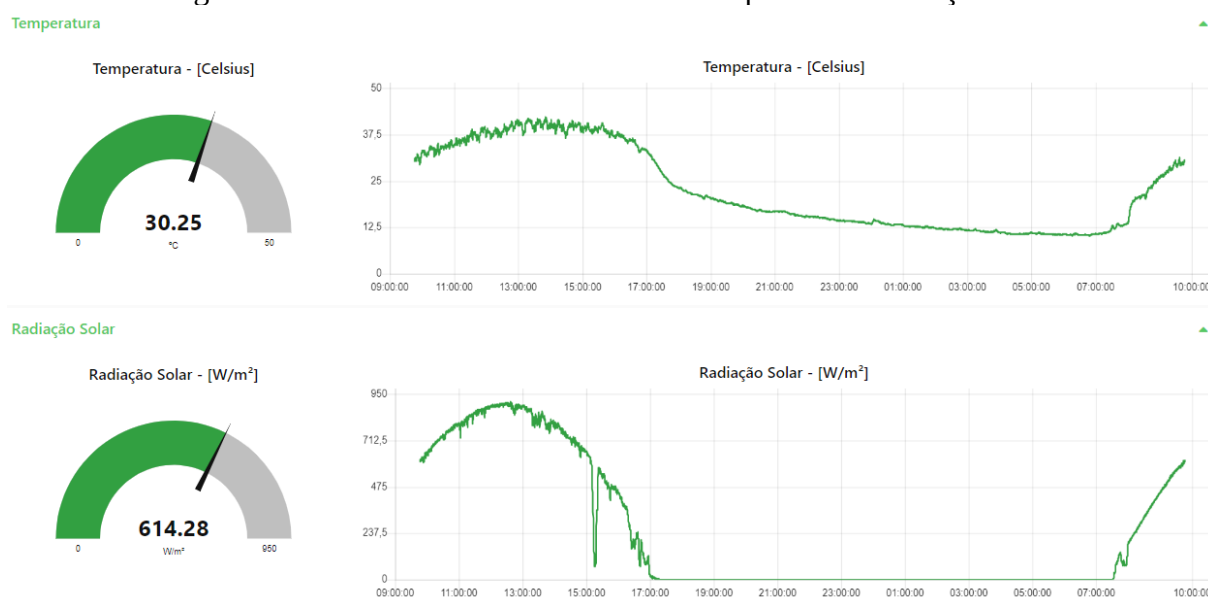
Figura 134 - *Nodes* para a Velocidade do Vento



Fonte: Elaborada pelo autor

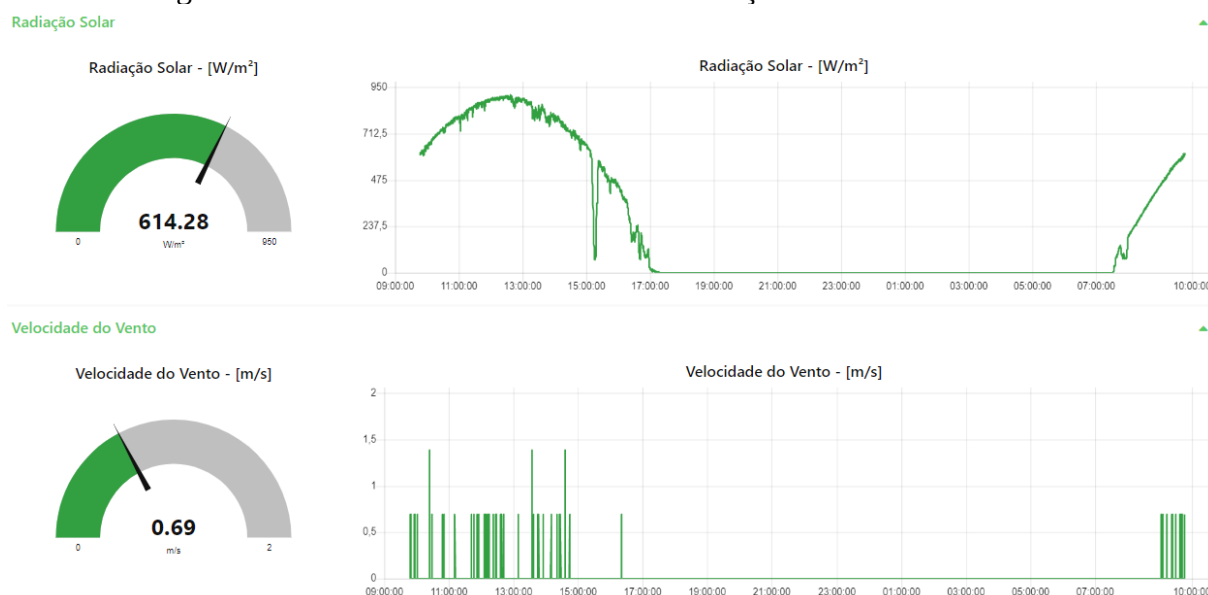
Após as personalizações a interface para exibição de gráficos em tempo real das medições dos valores de temperatura, velocidade do vento e radiação solar ficará conforme ilustrado na Figura 135 e na Figura 136.

Figura 135 - Interface de dados atuais da temperatura e radiação solar



Fonte: Elaborada pelo autor

Figura 136 - Interface de dados atuais da radiação e velocidade do vento



Fonte: Elaborada pelo autor

5.7.8 Armazenamento e exibição de dados históricos

Com a edição dos gráficos a próxima etapa será desenvolver uma forma de armazenar os valores obtidos pelos *nodes* de conexão com o *broker*, para que seja possível ter um histórico salvo de todos os dados que foram recebidos durante o processo de medições.

Este histórico estará disponível para consultas futuras, pesquisas, elaboração de projetos dentre outras utilidades.

Um modo simples de executar este objetivo é salvar os dados no armazenamento do próprio *Raspberry Pi*, isto é, em seu micro SD, se não em uma solução de armazenamento externo como HDD e SSD.

Outra opção é o uso de serviços de armazenamento em nuvem, mas que pode resultar em limitações quanto a quantidade disponível para armazenamento de forma gratuita ou inclusões de pagamentos mensais para a disponibilização de maior armazenamento.

O projeto utilizará o micro SD como forma de armazenamento, como forma de manter a condição de baixo custo.

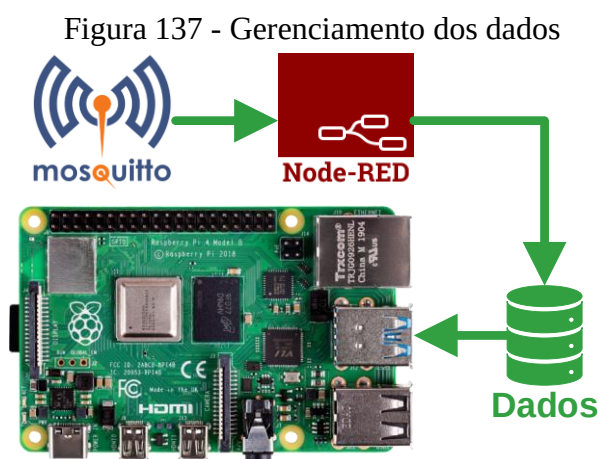
Para esta etapa, como o objetivo é criar uma interface personalizada para exibir e disponibilizar dados históricos, não haverá *nodes*, até o momento de desenvolvimento deste

projeto, com configurações já preparadas, tal como os *nodes* de gráficos apresentados anteriormente, levando a necessidade da criação de *nodes* próprios específicos ao objetivo atual.

A metodologia utilizada para a criação de cada *node* e a construção do fluxo será discorrida em partes nos próximos tópicos ao passo de que o código desenvolvido para cada *node* poderá ser encontrado nos respectivos apêndices.

5.7.8.1 Armazenamento de dados

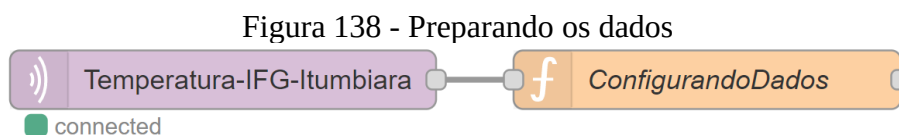
O primeiro passo é coletar os dados disponibilizados pelo *node mqtt in* e envia-los para o armazenamento do *Raspberry Pi*, conforme esquema apresentado na Figura 137.



Fonte: Elaborada pelo autor

Contudo os valores não podem ser enviados da maneira como o *node mqtt in* disponibiliza, de início é preciso organizar cada dado recebido em arquivos que poderão ser acessados posteriormente.

A organização dos nomes dos arquivos será por meio de datas e horários, para isto haverá uma ligação do *node mqtt in* até um *node* de funções responsável por coletar a data e horário atual do recebimento de cada informação, conforme demonstrado na Figura 138.



Fonte: Elaborada pelo autor

Este *node* coletará o ano, mês, dia, horas, minutos e segundos atuais, além do valor da última medição e o carimbo de tempo, que representa o momento que o valor foi coletado que será útil para a etapa de exibição de dados em gráficos.

A maior parte dos dados de tempo não serão utilizados, no entanto é conveniente deixá-los para criação de diferentes arquivos com diferentes variações de intervalo de tempo.

O próximo *node* de funções a ser criado irá gerar um nome de arquivo personalizado. Para este projeto o nome do arquivo contém o nome da grandeza física medida, ano, mês e o dia atual, assim, haverá um arquivo para cada dia do ano com uma série de dados coletados naquele mesmo dia, conforme ilustrado na Figura 139.



Fonte: Elaborada pelo autor

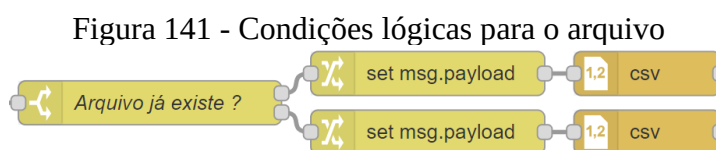
O intervalo de tempo para a geração de mais ou menos arquivos ao longo do dia, pode ser personalizado através deste mesmo *node* de geração de nome.

Em seguida o *node File Lister* também com nomenclatura *fs-file-lister* é utilizado para indicar o caminho onde o arquivo deve ser criado, conforme demonstrado na Figura 140.



Fonte: Elaborada pelo autor

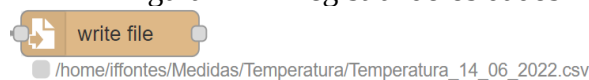
Contudo, antes da listagem é preciso checar se este mesmo arquivo já não existe, com uma lógica elaborada através dos *nodes switch* e *change*, sendo assim, caso o arquivo não exista um novo será criado, caso o arquivo já exista então o último dado coletado será adicionado neste arquivo, conforme demonstrado na Figura 141.



Fonte: Elaborada pelo autor

No final do fluxo o arquivo é criado através do *node Write file* em formato .csv em um local de escolha definido no *node File Lister*, conforme apresentado na Figura 142.

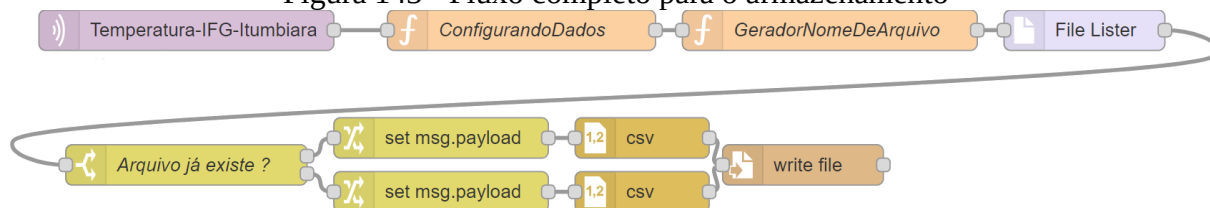
Figura 142 - Registrando os dados



Fonte: Elaborada pelo autor

O fluxo completo com os respectivos *nodes* é apresentado na Figura 143.

Figura 143 - Fluxo completo para o armazenamento



Fonte: Elaborada pelo autor

O código-fonte desenvolvido em cada *node* desta etapa são apresentados respectivamente nos apêndices 7 e 8.

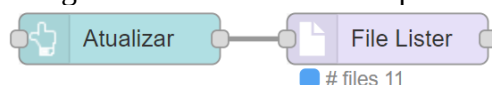
5.7.8.2 Exibição de dados – Listagem na interface

Com os dados criados e armazenados na etapa anterior, a próxima fase é criar o método de obtenção destes arquivos guardados no armazenamento do *Raspberry Pi* para exibição na interface de dados históricos.

A listagem dos arquivos é feita pelo *node File Lister*, responsável por checar e listar os arquivos existentes, para isto é preciso indicar o mesmo caminho utilizado para o armazenamento.

Para acionar o evento de checagem um botão pode ser conectado ao *node File Lister* conforme apresentado na Figura 144.

Figura 144 - Atualizando arquivos



Fonte: Elaborada pelo autor

O *node File Lister* fará uma busca por arquivos no caminho indicado sempre que o botão for pressionado.

Os arquivos encontrados podem ser listados na interface de várias forma, na interface de histórico são utilizados dois métodos, uma tabela e uma barra de rolagem utilizados respectivamente para o *download* e para o gráfico ambos a serem implementados.

O *node File Lister* concede como resposta todo o nome do caminho percorrido até o arquivo, portanto, para o nome do arquivo a ser exibido na tabela primeiro é criada uma função para remover os sinais e nomes das pastas do caminho do arquivo, conforme apresentado na Figura 145.

Figura 145 - Formatando nome do arquivo



Fonte: Elaborada pelo autor

O *node Template* do menu de funções é utilizado para a elaboração da tabela, neste *node* são colocados o título e o caminho dos arquivos para que possam ser exibidos na tabela, conforme demonstrado na Figura 146.

Figura 146 - Nome enviado para a tabela



Fonte: Elaborada pelo autor

No final o *node Template* do menu *dashboard* é utilizado para a personalização da tabela criada anteriormente. Esta edição é feita em HTML. A conexão do *node* fica conforme demonstrado na Figura 147.

Figura 147 - Enviando configurações para o gráfico tabela



Fonte: Elaborada pelo autor

Para o fluxo da barra de rolagem o *node File Lister* utilizado anteriormente é aproveitado seguindo uma ramificação para um novo *node* criado com o mesmo propósito de eliminar sinais e nomes de pastas do caminho do arquivo além de preparar o nome a ser exibido na barra de rolagem, conforme demonstrado na Figura 148.

Figura 148 - Formatando o nome do caminho do arquivo



Fonte: Elaborada pelo autor

Em seguida a informação é passada para o *node dropdown* do menu *dashboard* que representa a barra de rolagem. Este *node* foi denominado com o nome “Arquivo:” conforme demonstrado na Figura 149.

Figura 149 - Enviando nome para a barra de rolagem



Fonte: Elaborada pelo autor

Uma função a mais é adicionada no final para salvar o nome do arquivo selecionado na barra de rolagem da etapa anterior como forma de orientação para identificar qual arquivo deverá ser exibido no gráfico a ser desenvolvido.

A conexão do *node* função denominado Salvar é demonstrada na Figura 150.

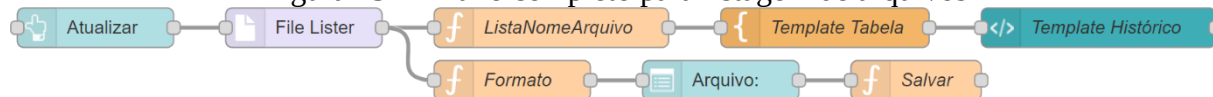
Figura 150 - Salvando o nome do arquivo selecionado



Fonte: Elaborada pelo autor

Com todos os *nodes* conectados o fluxo final para a etapa de listagem dos dados ficará conforme demonstrado na Figura 151.

Figura 151 - Fluxo completo para listagem de arquivos



Fonte: Elaborada pelo autor

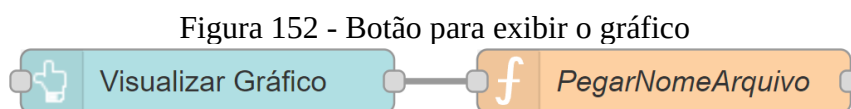
O código-fonte desenvolvido em cada *node* desta etapa são apresentados respectivamente nos apêndices 9, 10 e 11.

5.7.8.3 Exibição de dados - Visualização em um gráfico

O gráfico a ser elaborada tem o objetivo de exibir as informações contidas em cada arquivo anteriormente criado.

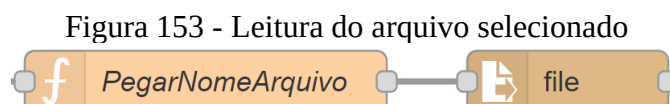
Com a etapa de listagem de arquivos concluída o primeiro *node* utilizado será um desenvolvido para coletar o nome do arquivo armazenado no *node* Salvar da etapa anterior.

Para acionar a exibição dos dados no gráfico um botão é adicionado, conforme demonstrado na Figura 152.



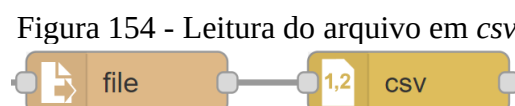
Fonte: Elaborada pelo autor

O *node* que coleta o nome do arquivo é conectado ao *node* File in para a leitura do arquivo, conforme demonstrado na Figura 153.



Fonte: Elaborada pelo autor

Como os arquivos foram salvos no formato .csv o *node* csv é utilizado novamente para a leitura, conforme demonstrado na Figura 154.



Fonte: Elaborada pelo autor

O próximo passo é formatar os dados lidos para a exibição final no gráfico através de um *node* desenvolvido denominado “FormatarDadosParaGráfico”, conforme demonstrado na Figura 155.

Figura 155 - Formatando dados para o gráfico



Fonte: Elaborada pelo autor

Com a formatação dos valores a serem exibidos, o *node chart* do menu *dashboard* é utilizado para a exibição final das informações, conforme demonstrado na Figura 156.

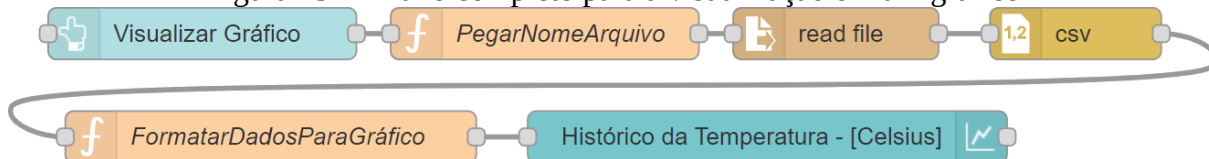
Figura 156 - Enviando dados para o gráfico



Fonte: Elaborada pelo autor

O fluxo final com os *nodes* conectados para a etapa da exibição dos dados históricos em um gráfico é apresentado na Figura 157.

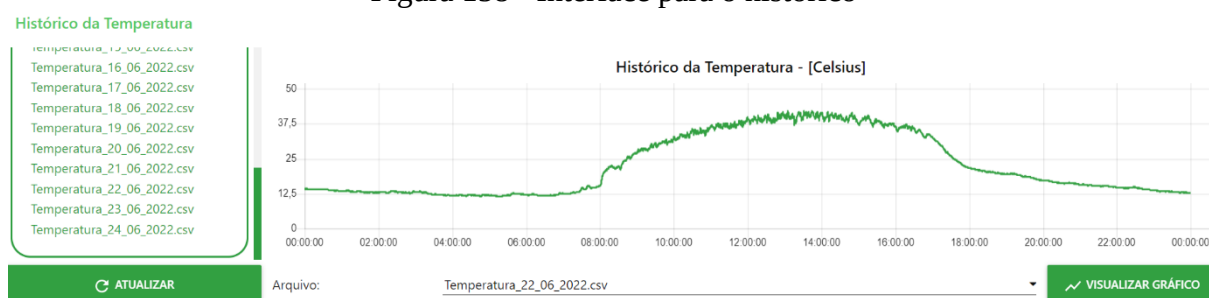
Figura 157 - Fluxo completo para a visualização em um gráfico



Fonte: Elaborada pelo autor

Na Figura 158 é apresenta a forma final da interface para os dados históricos, neste caso, para as medições da temperatura.

Figura 158 - Interface para o histórico



Fonte: Elaborada pelo autor

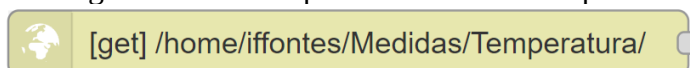
O código-fonte desenvolvido em cada *node* desta etapa são apresentados respectivamente nos apêndices 12 e 13.

5.7.8.4 Exibição de dados - *Download* pela interface

O *download* é feito através da tabela anteriormente criada, para que os dados armazenados possam ser adquiridos por terceiros.

Inicialmente é utilizado o *node http in* com o método *GET* para acessar por meio de URL em uma interface *web* a pasta com o arquivo no *Raspberry Pi*, conforme apresentado na Figura 159.

Figura 159 - *Node* para o caminho do arquivo

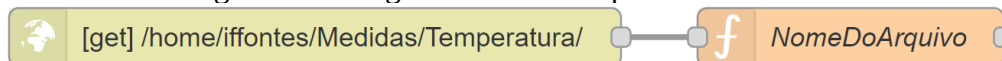


Fonte: Elaborada pelo autor

Com o *node http in* configurado, o próximo passo é indicar qual arquivo em específico se deseja fazer o *download*.

Para que o fluxo reconheça este arquivo um novo *node* função é desenvolvido denominado “NomeDoArquivo”, conforme demonstrado na Figura 160.

Figura 160 - Pegar o nome do arquivo selecionado



Fonte: Elaborada pelo autor

O reconhecimento do arquivo ocorre quando um nome é selecionado na tabela dados históricos criada anteriormente.

Com este arquivo indicado o próximo passo é a sua leitura, feita pelo *node File in* apresentado na Figura 161.

Figura 161 - Leitura do arquivo selecionado



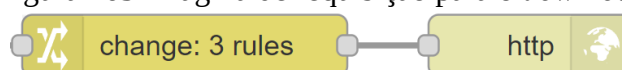
Fonte: Elaborada pelo autor

Após o *node File in* três condições de leitura são alteradas através do *node Change* para que o arquivo possa ser disponibilizado, conforme demonstrado na Figura 162.

Figura 162 - Preparação do arquivo para *download*

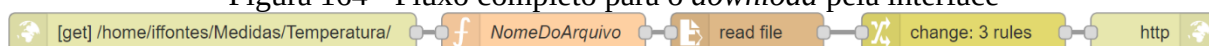
Fonte: Elaborada pelo autor

Com o arquivo preparado o *node http response* encerra o fluxo abrindo como resposta uma guia para o *download* do arquivo, conforme demonstrado na Figura 163.

Figura 163 - Página de requisição para o *download*

Fonte: Elaborada pelo autor

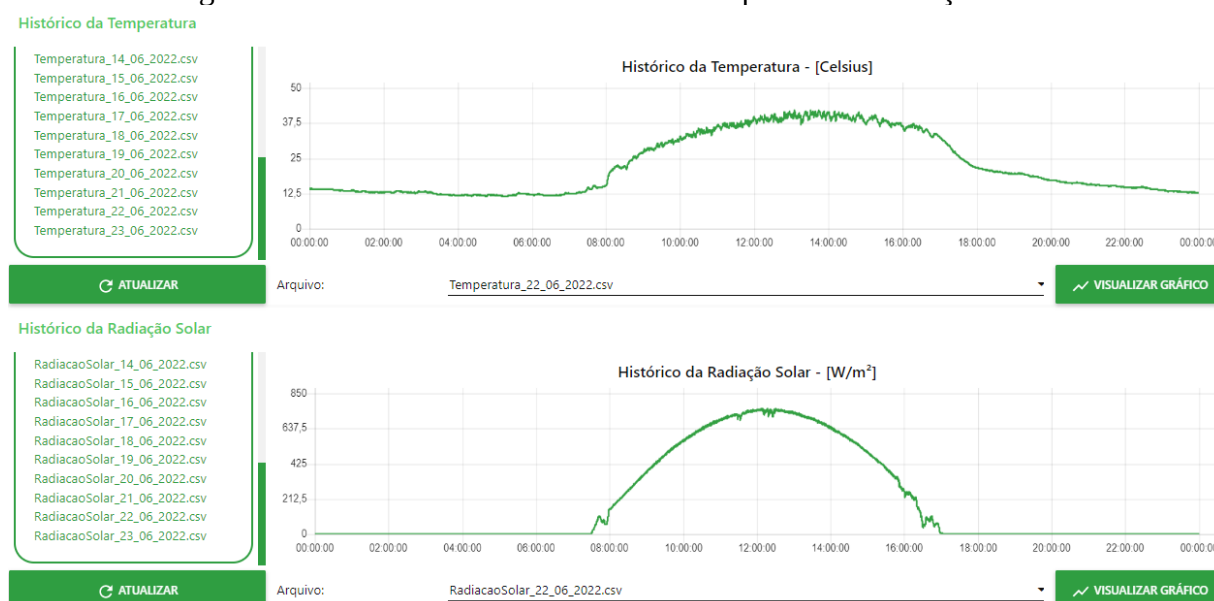
O fluxo final com os *nodes* conectados para a etapa de *download* é apresentado na Figura 164.

Figura 164 - Fluxo completo para o *download* pela interface

Fonte: Elaborada pelo autor

Na Figura 165 são apresentados os gráficos de dados históricos da temperatura e radiação solar.

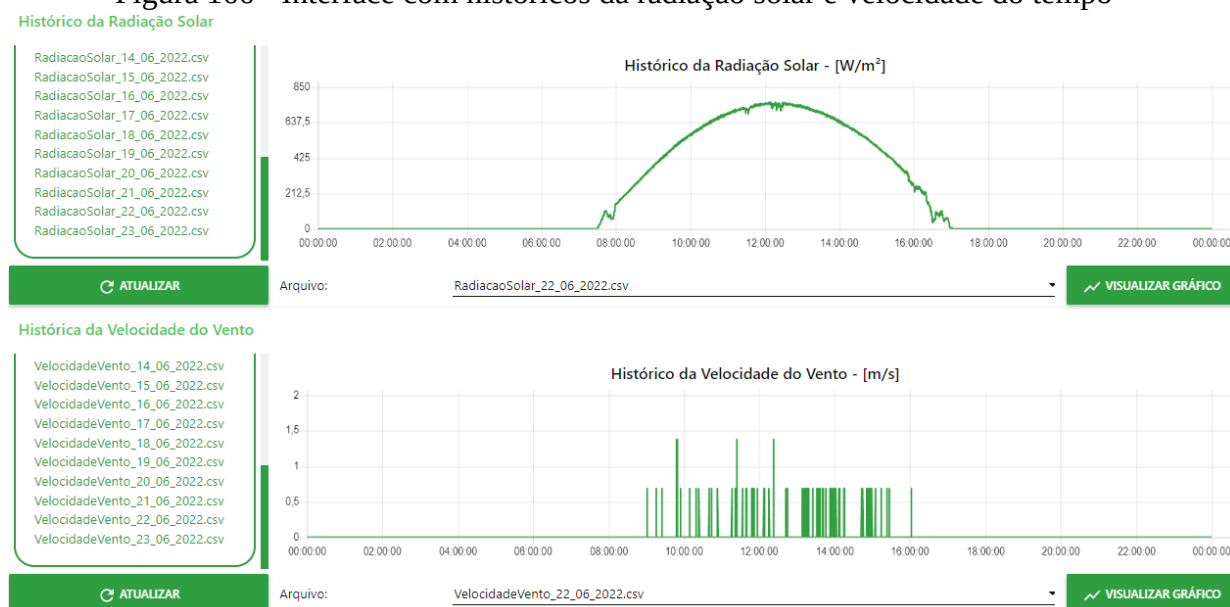
Figura 165 - Interface com históricos da temperatura e radiação solar



Fonte: Elaborada pelo autor

Na Figura 166 são apresentados os gráficos de dados históricos da radiação solar e velocidade do vento.

Figura 166 - Interface com históricos da radiação solar e velocidade do tempo



Fonte: Elaborada pelo autor

O código-fonte desenvolvido no *node* desta etapa é apresentado no apêndice 14.

Os fluxos seguem a mesma metodologia para as demais medições históricas de radiação solar e velocidade do vento.

O código completo das interfaces desenvolvidas no *Node-RED* pode ser encontrado no apêndice 15.

Nos apêndices 16 e 17 são apresentados os fluxos para as interfaces de dados atuais e dados históricos, respectivamente.

6 CAPÍTULO VI – RESULTADOS E COMPARAÇÕES

Neste capítulo são apresentados os resultados do projeto desenvolvido em funcionamento.

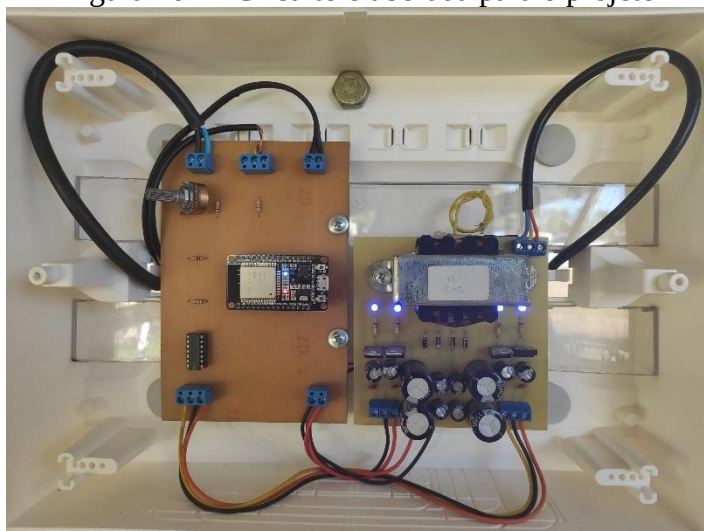
6.1 PARTE FÍSICA

Uma placa de circuito impresso foi construída para a ligação do microcontrolador ESP32 aos sensores juntamente ao circuito condicionador de sinal para a amplificação dos dados analógicos do sensor de irradiância solar Spektron 210.

Em adicional uma fonte simétrica foi anexada ao projeto para alimentação do circuito amplificador de sinal com valores de +12 Volts e -12 Volts e alimentação do microcontrolador com valor de +5 Volts.

Na Figura 167 é apresentado o circuito elaborado para a conexão dos sensores ao microcontrolador.

Figura 167 – Circuito elaborada para o projeto



Fonte: Elaborada pelo autor

O circuito amplificador de sinal conta com o potenciômetro para ajustes e calibrações no sinal analógico do sensor de irradiância.

Com a caixa fechada, apresentada na Figura 168, ainda é possível ter acesso ao microcontrolador para qualquer alteração necessária em seu código-fonte.

Figura 168 - Caixa fechada com a tampa aberta



Fonte: Elaborada pelo autor

A caixa contendo o microcontrolador e seus circuitos adjacentes está localizada abaixo de painéis fotovoltaicos que compõe um sistema de geração fotovoltaico, conforme apresentado na Figura 169.

Figura 169 - Caixa com o microcontrolador ESP32 posicionada



Fonte: Elaborada pelo autor

O sensor de temperatura DS18B20 está localizado entre os painéis fotovoltaicos, em contato direto com as placas, para o acompanhamento diário da temperatura dos painéis, conforme apresentado na Figura 170.

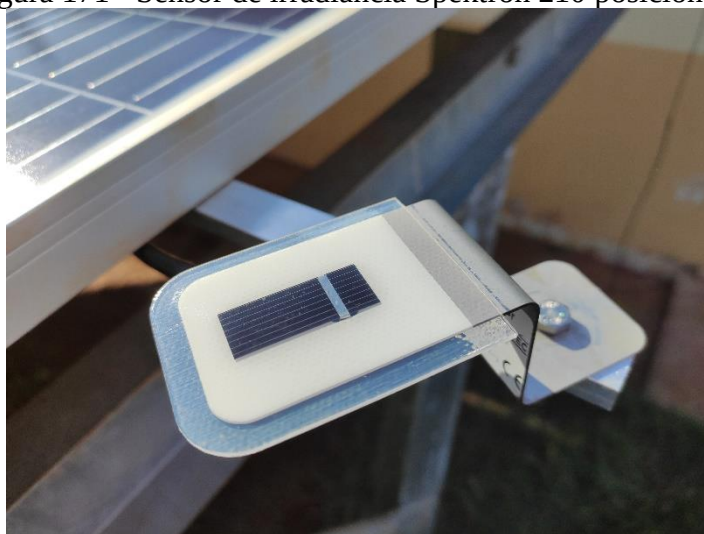
Figura 170 - Sensor de temperatura DS18B20 posicionado



Fonte: Elaborada pelo autor

O sensor de irradiância Spektron 210 está ao lado dos painéis fotovoltaicos, fixado no plano inclinado com angulação de 18° e apontado para o norte, atuando como um sensor de referência da geração de energia do sistema fotovoltaico, conforme apresentado na Figura 171.

Figura 171 - Sensor de irradiância Spektron 210 posicionado



Fonte: Elaborada pelo autor

O anemômetro de copos está localizado próximo ao sistema de geração fotovoltaico de modo a não atrapalhar a geração devido a sombras com altura de 4,2 metros em relação ao solo, conforme apresentado na Figura 172.

Figura 172 - Anemômetro de copos posicionado



Fonte: Elaborada pelo autor

Na Figura 173 é apresentado o local de instalação dos sensores do projeto.

Figura 173 - Local de instalação do projeto e sensores



Fonte: Elaborada pelo autor

Na Figura 174 é apresentado o local de instalação do circuito elétrico do projeto.

Figura 174 - Local de instalação do circuito do projeto



Fonte: Elaborada pelo autor

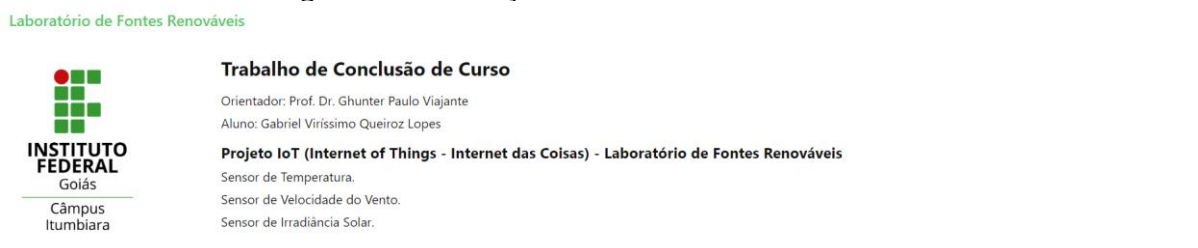
6.2 PARTE VIRTUAL

A parte virtual do projeto se resume à interface gráfica para a exibição dos dados coletados, que estão divididas em duas páginas, uma para os dados atuais e a outra para os dados históricos.

6.2.1 Interface de dados atuais

A interface de dados atuais tem o objetivo de informar o acompanhamento das medições coletadas em tempo em real, nesta interface o primeiro grupo é composto pelo cabeçalho do projeto, conforme apresentado na Figura 175.

Figura 175 - Cabeçalho da Interface de dados atuais

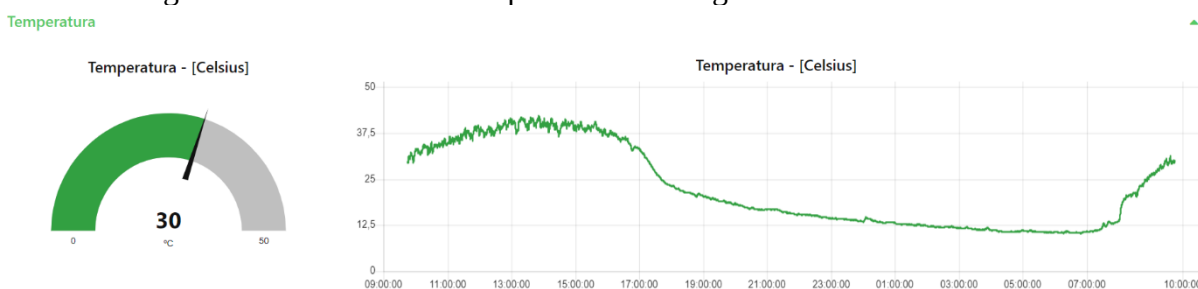


Fonte: Elaborada pelo autor

O grupo seguinte indica através do medidor o valor de temperatura em tempo real medida em grau celsius pelo sensor DS18B20.

Ao lado do medidor o gráfico é indicado os últimos registros de temperatura coletados em um intervalo de 24 horas, conforme apresentado na Figura 176.

Figura 176 - Medidor da temperatura atual e gráfico das últimas 24 horas



Fonte: Elaborada pelo autor

A radiação solar é a próxima medição apresentada, com o medidor a esquerda indicando o valor de radiação solar atual com unidade de W/m^2 e o gráfico a direita com as medições das últimas 24 horas, conforme apresentado na Figura 177.

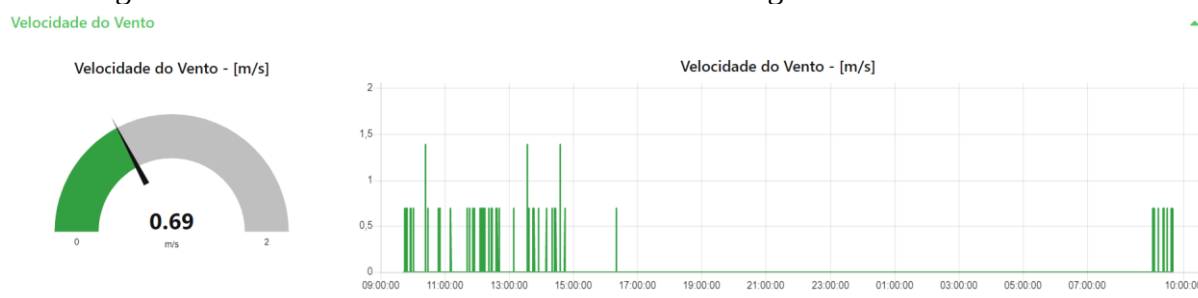
Figura 177 - Medidor da radiação solar atual e gráfico das últimas 24 horas



Fonte: Elaborada pelo autor

O grupo final indica a velocidade do vento em tempo real em metros por segundo através do medidor e o registro das últimas 24 horas de medições no gráfico ao lado, conforme apresentado na Figura 178.

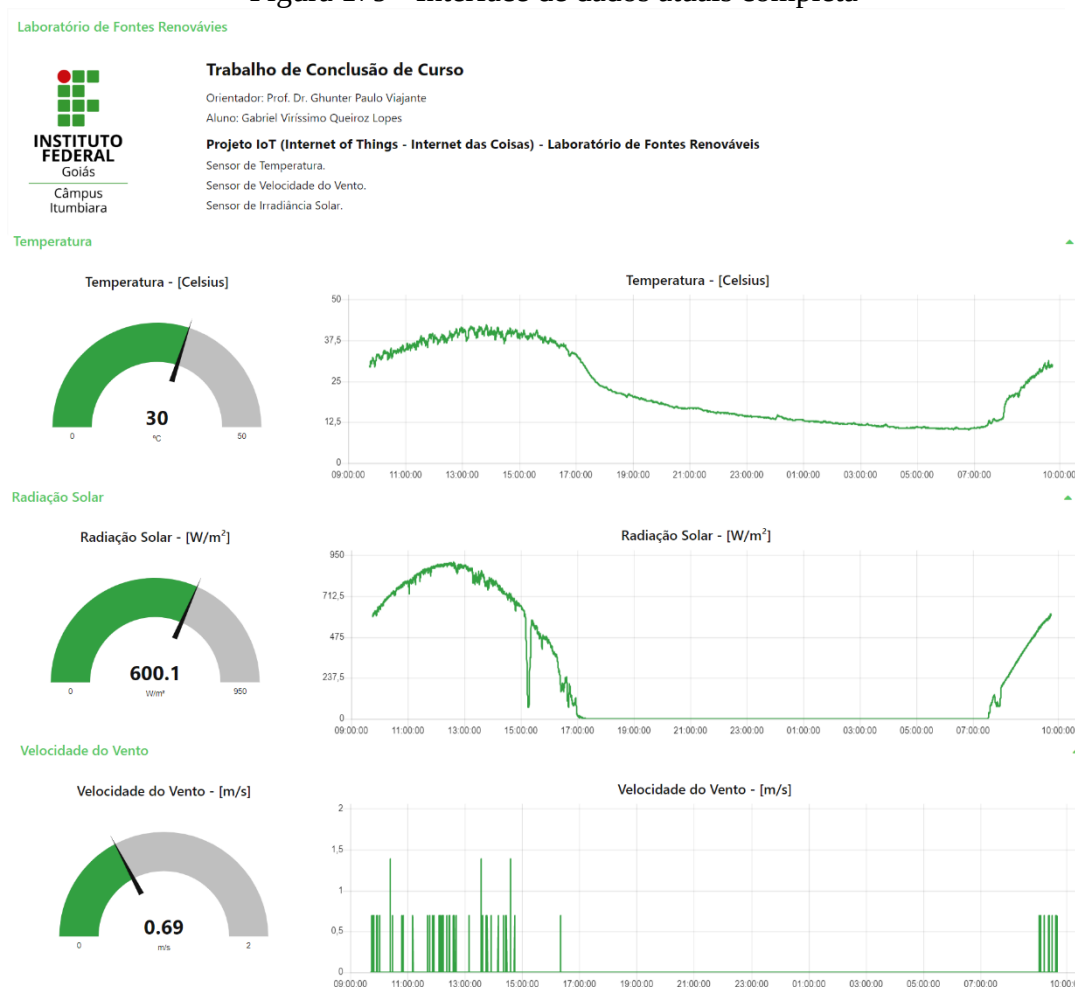
Figura 178 - Medidor da velocidade de vento atual e gráfico das últimas 24 horas



Fonte: Elaborada pelo autor

A interface completa para os dados atuais é apresentada na Figura 179.

Figura 179 - Interface de dados atuais completa



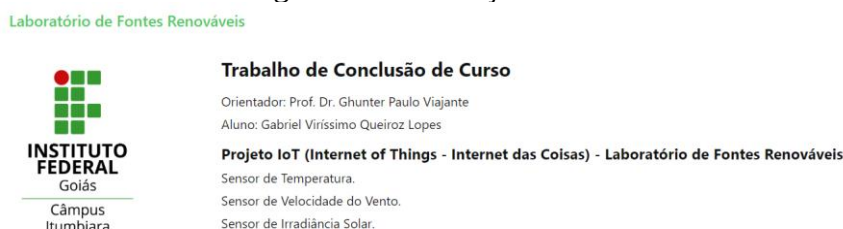
Fonte: Elaborada pelo autor

6.2.2 Interface de dados históricos

A interface de dados históricos tem o objetivo de disponibilizar por meio de *download* os arquivos com as medições realizadas em dias passados, além de uma opção gráfica para pré-visualização do conteúdo a ser adquirido.

A interface é iniciada com um cabeçalho assim como na interface apresentada anteriormente, conforme apresentado na Figura 180.

Figura 180 – Cabeçalho da interface de dados históricos

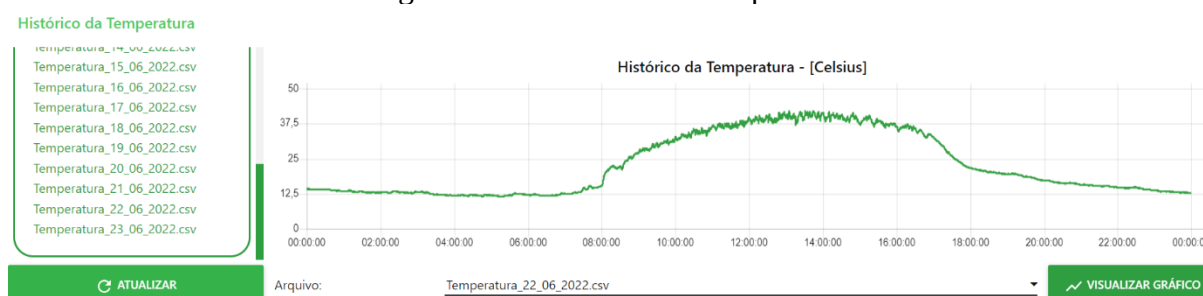


Fonte: Elaborada pelo autor

O segundo grupo é composto por uma tabela a esquerda onde é possível selecionar o arquivo de temperatura desejado para o *download* e um gráfico a direita para visualização do conteúdo contido em cada arquivo, conforme apresentado na Figura 181.

Para a tabela os arquivos podem ser atualizados utilizando o botão “atualizar” localizado abaixo da tabela, já o gráfico irá exibir o arquivo escolhido através da barra de rolagem quando o botão “visualizar gráfico” for selecionado, ambos localizados abaixo do gráfico, conforme apresentado na Figura 181.

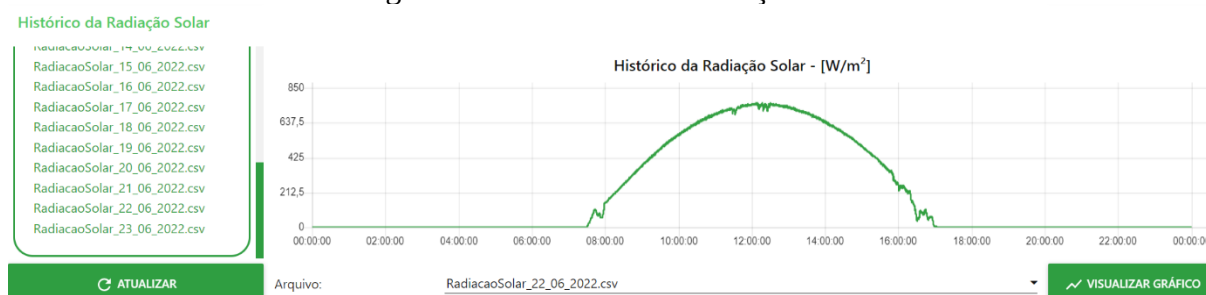
Figura 181 - Histórico da temperatura



Fonte: Elaborada pelo autor

O grupo seguinte disponibiliza o histórico de radiação solar registrado para *download* através da tabela, assim como o gráfico para visualização prévia do arquivo escolhido, conforme apresentado na Figura 182.

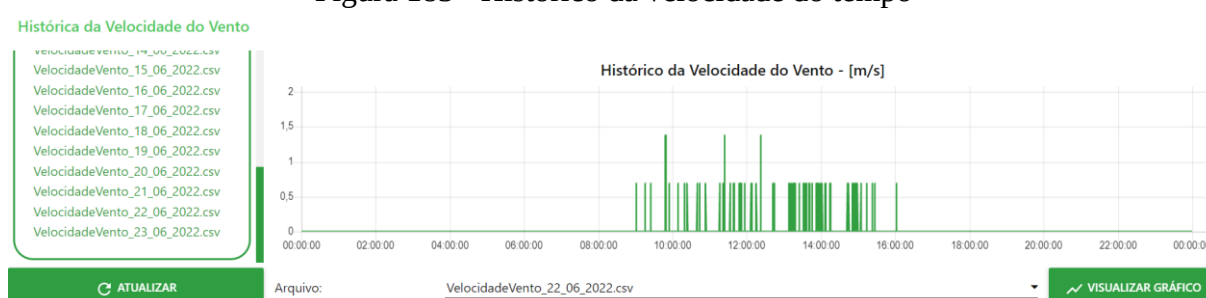
Figura 182 - Histórico da radiação solar



Fonte: Elaborada pelo autor

O próximo grupo disponibiliza o histórico da velocidade do vento para *download* e a opção gráfica para uma visualização prévia, conforme apresentado na Figura 183.

Figura 183 - Histórico da velocidade do tempo



Fonte: Elaborada pelo autor

Todos os arquivos são disponibilizados em formato .csv, onde cada dado é separado por vírgula.

A interface completa da página de dados histórico pode ser visualizada na Figura 184.

Figura 184 - Interface de dados históricos completa

Laboratório de Fontes Renováveis

**Trabalho de Conclusão de Curso**

Orientador: Prof. Dr. Ghunter Paulo Viajante

Aluno: Gabriel Viríssimo Queiroz Lopes

Projeto IoT (Internet of Things - Internet das Coisas) - Laboratório de Fontes Renováveis

Sensor de Temperatura.

Sensor de Velocidade do Vento.

Sensor de Irradiância Solar.

Histórico da Temperatura

Temperatura_15_06_2022.csv
 Temperatura_16_06_2022.csv
 Temperatura_17_06_2022.csv
 Temperatura_18_06_2022.csv
 Temperatura_19_06_2022.csv
 Temperatura_20_06_2022.csv
 Temperatura_21_06_2022.csv
 Temperatura_22_06_2022.csv
 Temperatura_23_06_2022.csv

Histórico da Temperatura - [Celsius]

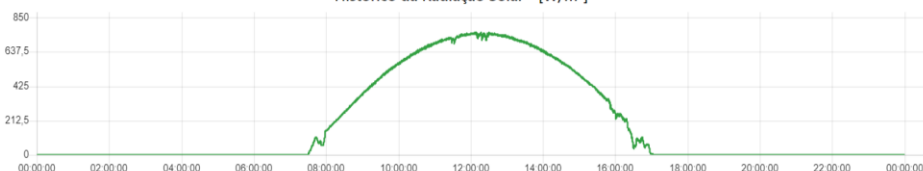
ATUALIZAR

Arquivo: Temperatura_22_06_2022.csv

VISUALIZAR GRÁFICO

Histórico da Radiação Solar

RadiacaoSolar_15_06_2022.csv
 RadiacaoSolar_16_06_2022.csv
 RadiacaoSolar_17_06_2022.csv
 RadiacaoSolar_18_06_2022.csv
 RadiacaoSolar_19_06_2022.csv
 RadiacaoSolar_20_06_2022.csv
 RadiacaoSolar_21_06_2022.csv
 RadiacaoSolar_22_06_2022.csv
 RadiacaoSolar_23_06_2022.csv

Histórico da Radiação Solar - [W/m²]

ATUALIZAR

Arquivo: RadiacaoSolar_22_06_2022.csv

VISUALIZAR GRÁFICO

Histórica da Velocidade do Vento

VelocidadeVento_15_06_2022.csv
 VelocidadeVento_16_06_2022.csv
 VelocidadeVento_17_06_2022.csv
 VelocidadeVento_18_06_2022.csv
 VelocidadeVento_19_06_2022.csv
 VelocidadeVento_20_06_2022.csv
 VelocidadeVento_21_06_2022.csv
 VelocidadeVento_22_06_2022.csv
 VelocidadeVento_23_06_2022.csv

Histórico da Velocidade do Vento - [m/s]

ATUALIZAR

Arquivo: VelocidadeVento_22_06_2022.csv

VISUALIZAR GRÁFICO

Fonte: Elaborada pelo autor

As interfaces são exibidas em um monitor no laboratório de fontes renováveis do Instituto Federal de Goiás, campus Itumbiara.

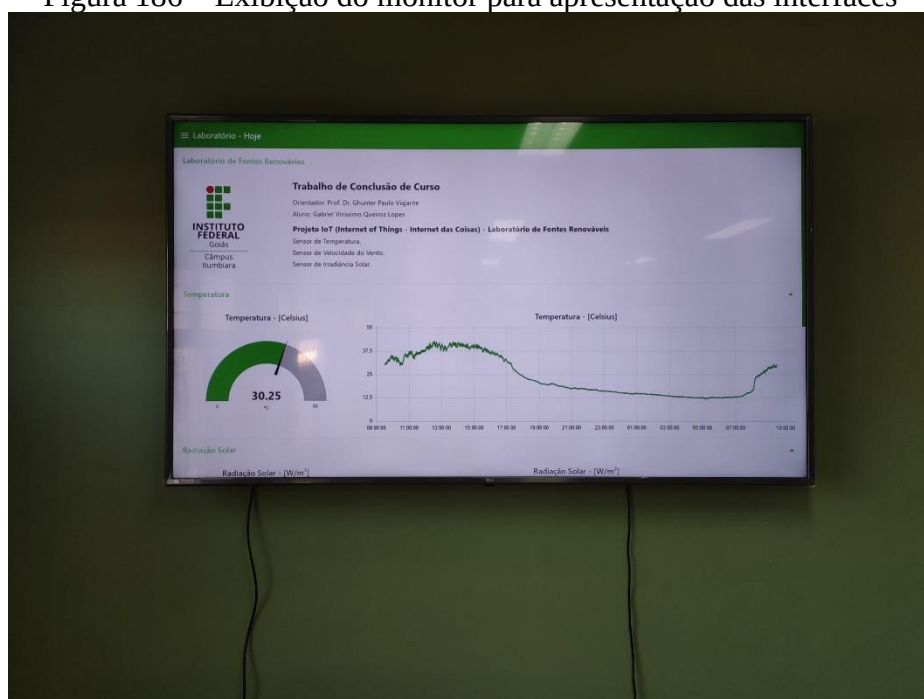
As interfaces em exibição são apresentadas na Figura 185 e Figura 186.

Figura 185 - Monitor para exibição das interfaces no laboratório



Fonte: Elaborada pelo autor

Figura 186 – Exibição do monitor para apresentação das interfaces



Fonte: Elaborada pelo autor

6.3 ANÁLISE GRÁFICA

As medidas realizadas pela estação solarimétrica de baixo custo foram comparadas com as medições registradas no mesmo dia por uma estação solarimétrica padrão EPE – Empresa de Pesquisa Energética também montada no Instituto Federal de Goiás, campus Itumbiara.

A estação solarimétrica no padrão EPE é apresentada na Figura 187.

Figura 187 - Estação solarimétrica no padrão EPE da Instituição



Fonte: Elaborada pelo autor

6.3.1 Resultados e comparações

6.3.1.1 Resultados das medições de temperatura

As medições de temperatura do sensor DS18B20 encapsulado são comparadas com as medições do sensor de temperatura e umidade relativa do ar EE060 com abrigo meteorológico ilustrado na Figura 188.

Figura 188 - Sensor EE060



Fonte: Sigma Sensors [28]

Na Figura 189 é ilustrado o abrigo meteorológico para o sensor EE060.

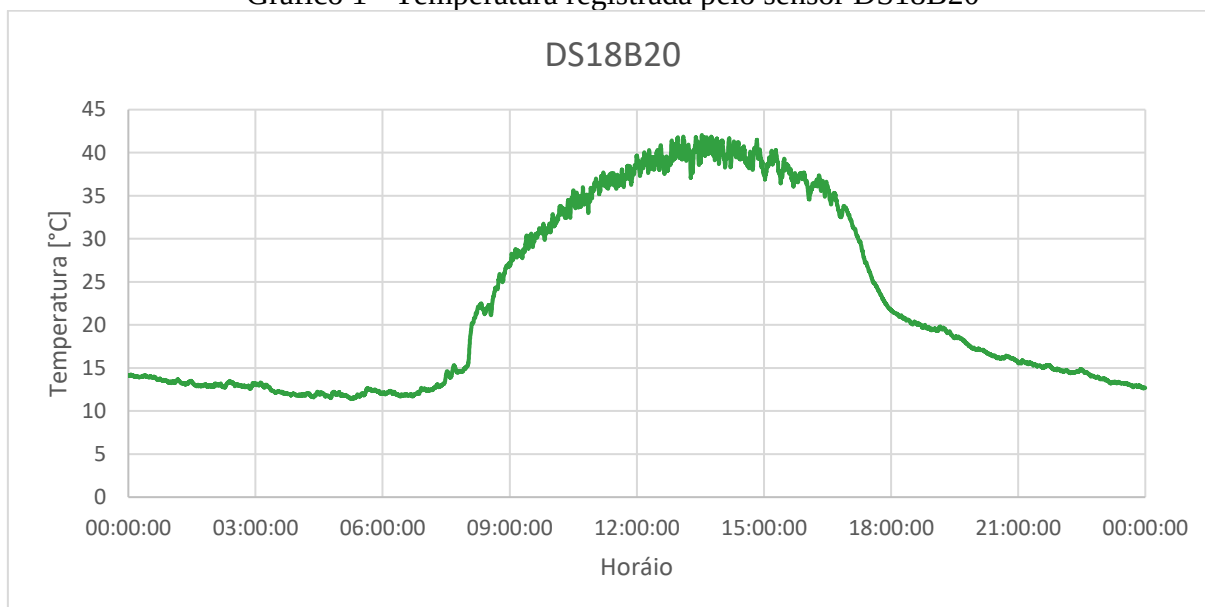
Figura 189 - Abrigo meteorológico



Fonte: Sigma Sensors [28]

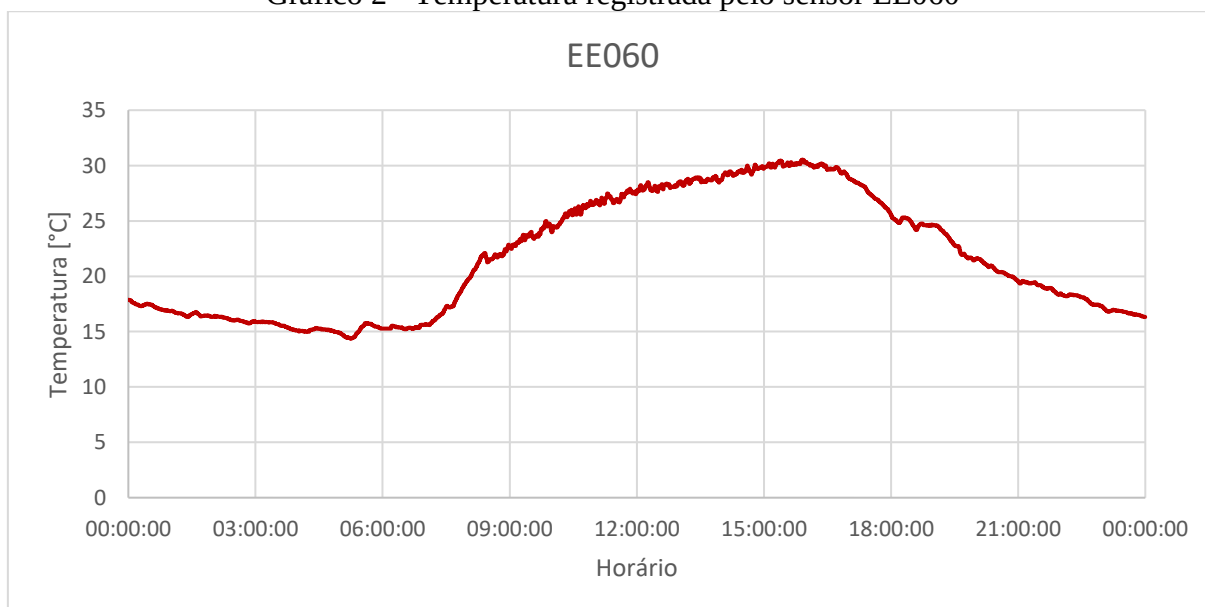
As medições do Gráfico 1 foram realizadas pelo sensor DS18B20 no dia 22 de junho de 2022 em um período de 24 horas. Devido a posição o sensor permanece medindo a temperatura do painel fotovoltaico.

Gráfico 1 - Temperatura registrada pelo sensor DS18B20



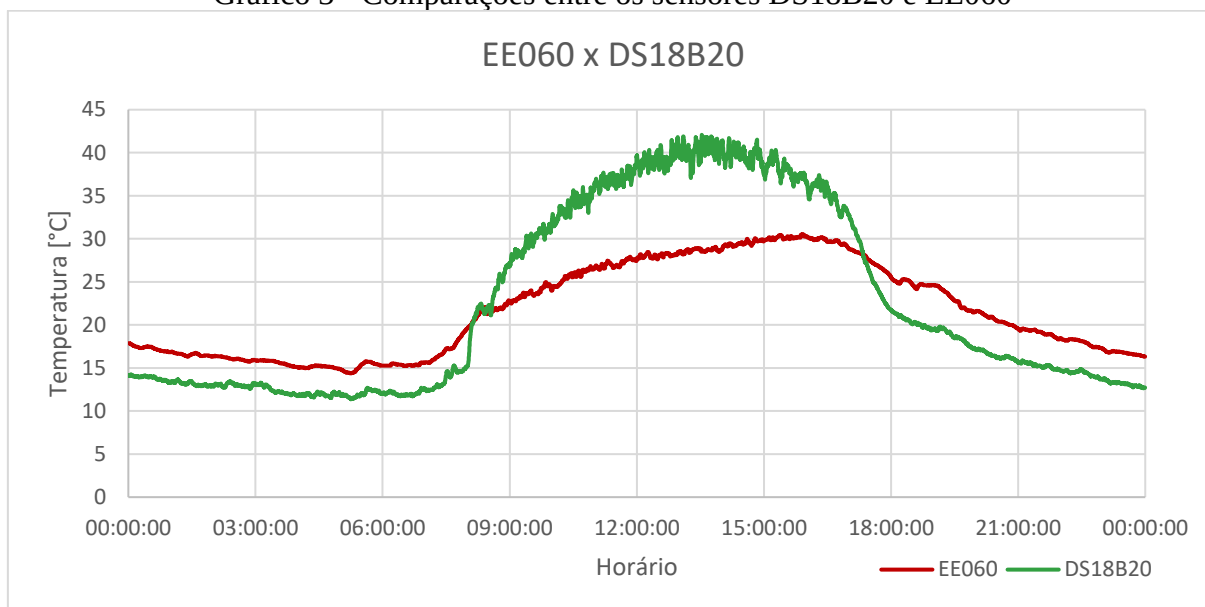
As medições do sensor EE060 para o dia 22 de junho de 2022 em um período de 24 horas estão registradas no Gráfico 2. Devido a posição este sensor permanece medindo a temperatura ambiente.

Gráfico 2 - Temperatura registrada pelo sensor EE060



No Gráfico 3 é feito o comparativo entre as medições feitas pelos sensores de temperatura no mesmo dia e período de 24 horas.

Gráfico 3 - Comparações entre os sensores DS18B20 e EE060



6.3.1.2 Resultados das medições de radiação solar

As medições de radiação solar do sensor Spektron 210 são comparadas com as medições do piranômetro de termopilha SR15-D2A2 conforme ilustrado na Figura 190. Os sensores foram ajustados ambos no plano horizontal, para fins de comparação.

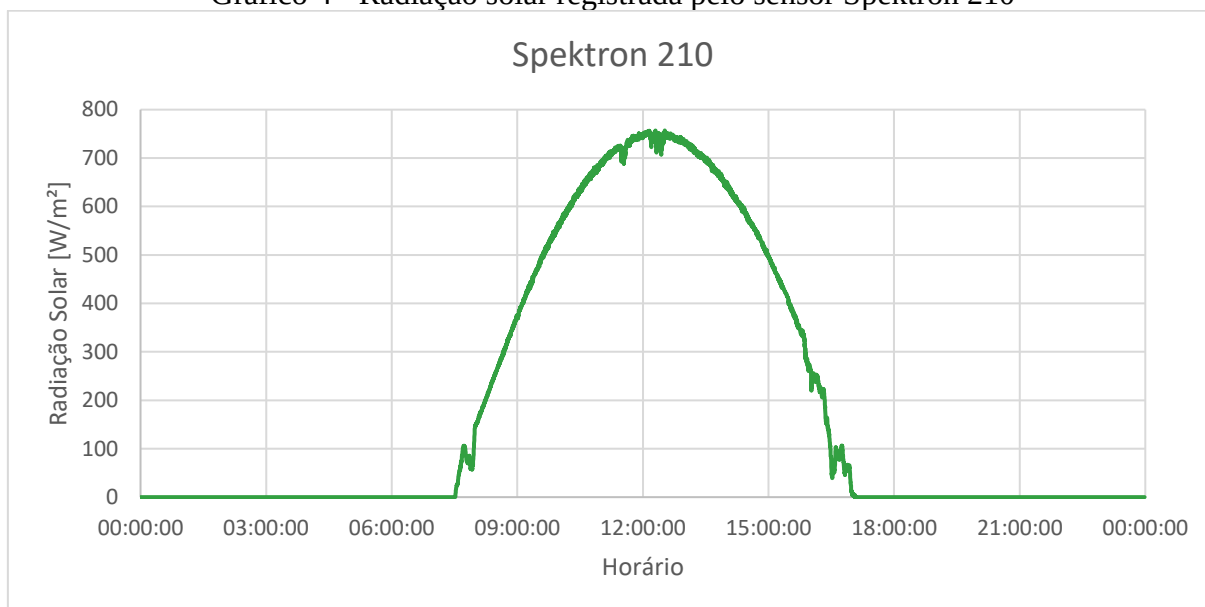
Figura 190 - Piranômetro SR15-D2A2



Fonte: Sigma Sensors [28]

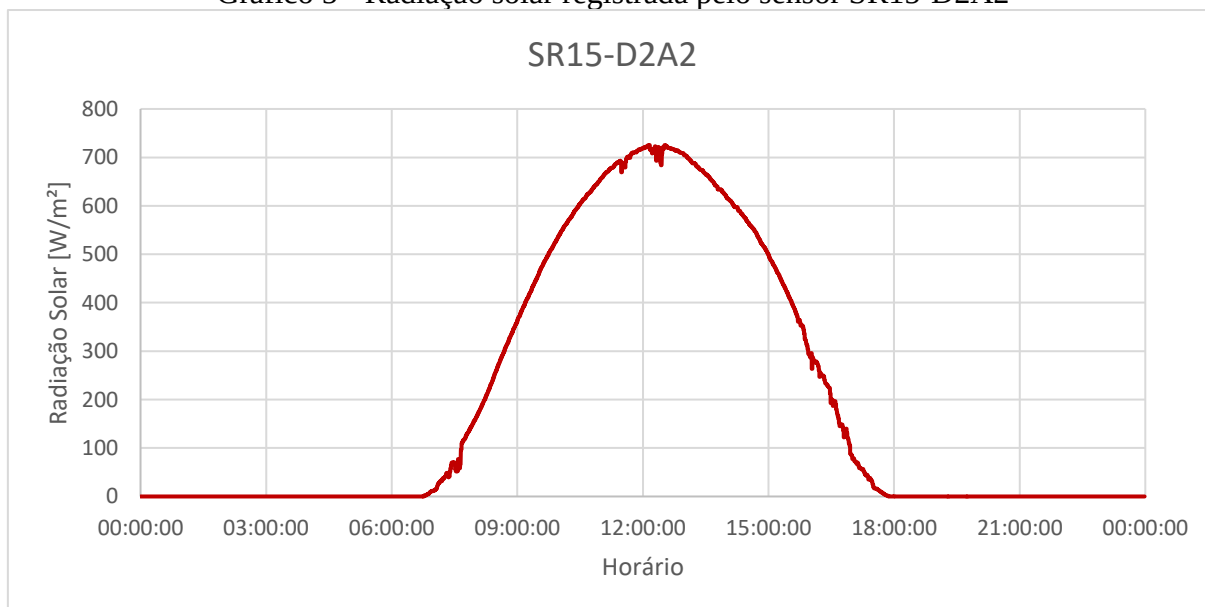
No Gráfico 4 é apresentada as medições da radiação solar realizadas pelo Spektron 210 no dia 22 de junho de 2022 durante um intervalo de 24 horas.

Gráfico 4 - Radiação solar registrada pelo sensor Spektron 210



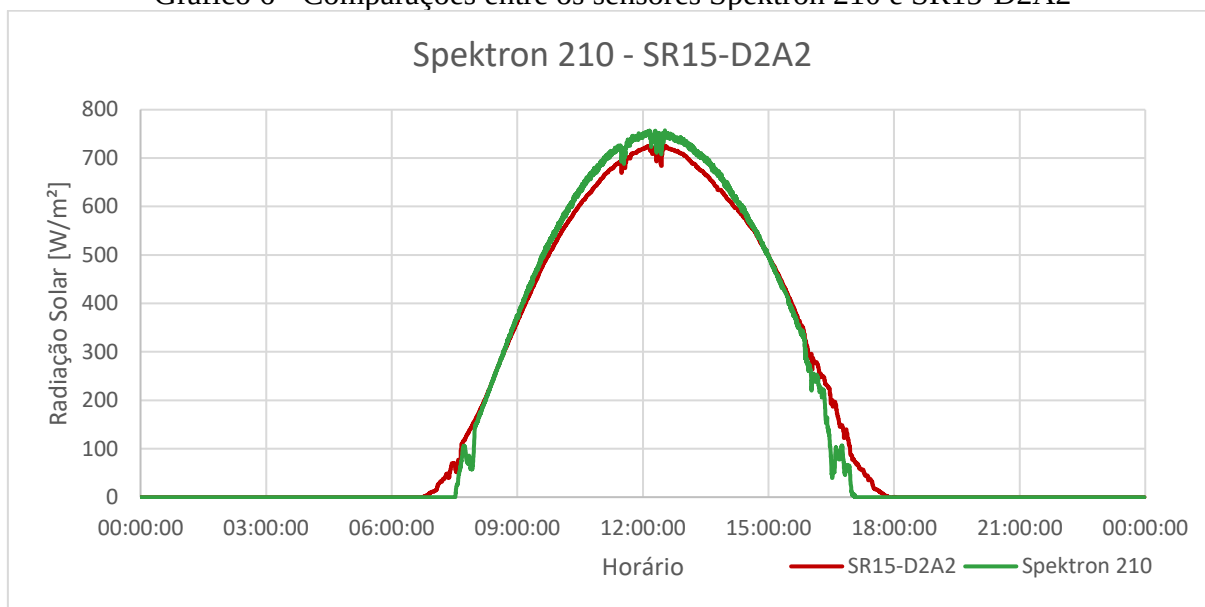
No Gráfico 5 é apresentada as medições da radiação solar realizadas pelo piranômetro SR15-D2A2 no dia 22 de junho de 2022 para o mesmo intervalo de 24 horas.

Gráfico 5 - Radiação solar registrada pelo sensor SR15-D2A2



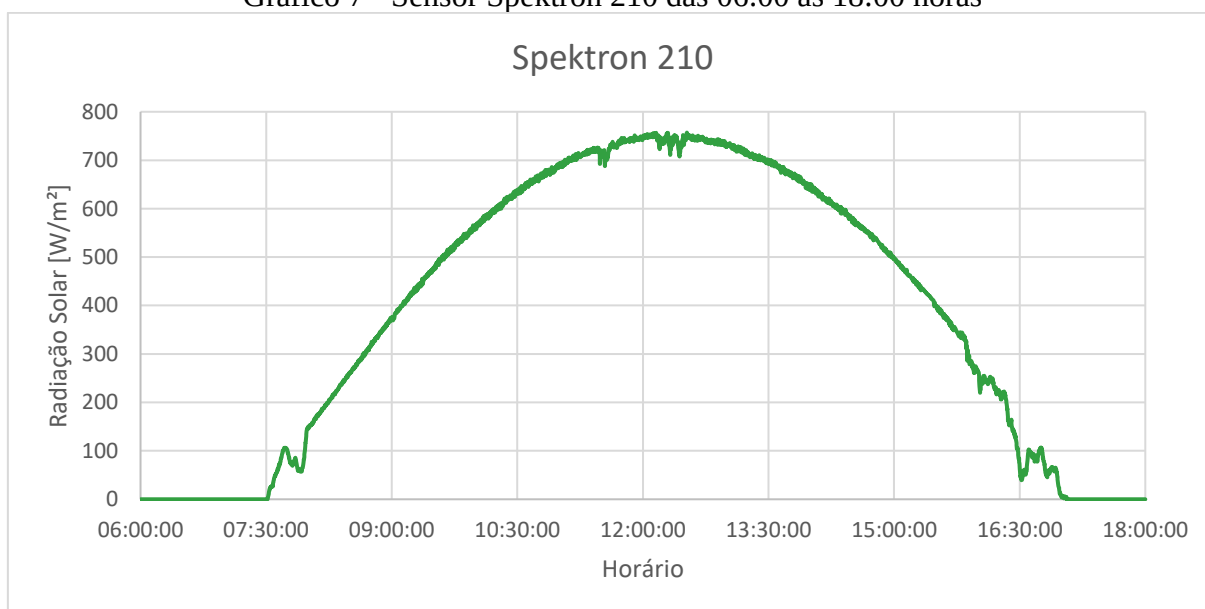
No Gráfico 6 é apresentada as comparações entre as medições do sensor de irradiância Spektron 210 e o piranômetro SR15-D2A2 no dia 22 de junho de 2022 no mesmo intervalo de 24 horas.

Gráfico 6 - Comparações entre os sensores Spektron 210 e SR15-D2A2



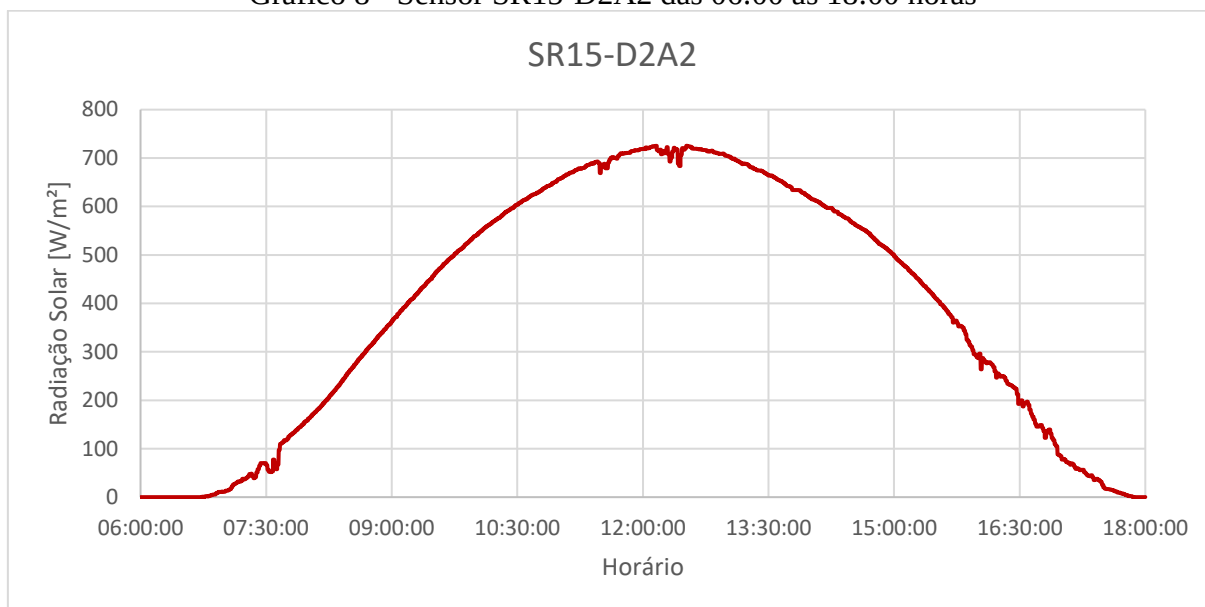
Fazendo uma aproximação para o intervalo das 06:00 às 18:00 horas das medições realizadas pelo sensor Spektron 210 ainda para o dia 22 de junho de 2022 é possível ter uma melhor visualização, conforme apresentado no Gráfico 7.

Gráfico 7 - Sensor Spektron 210 das 06:00 às 18:00 horas



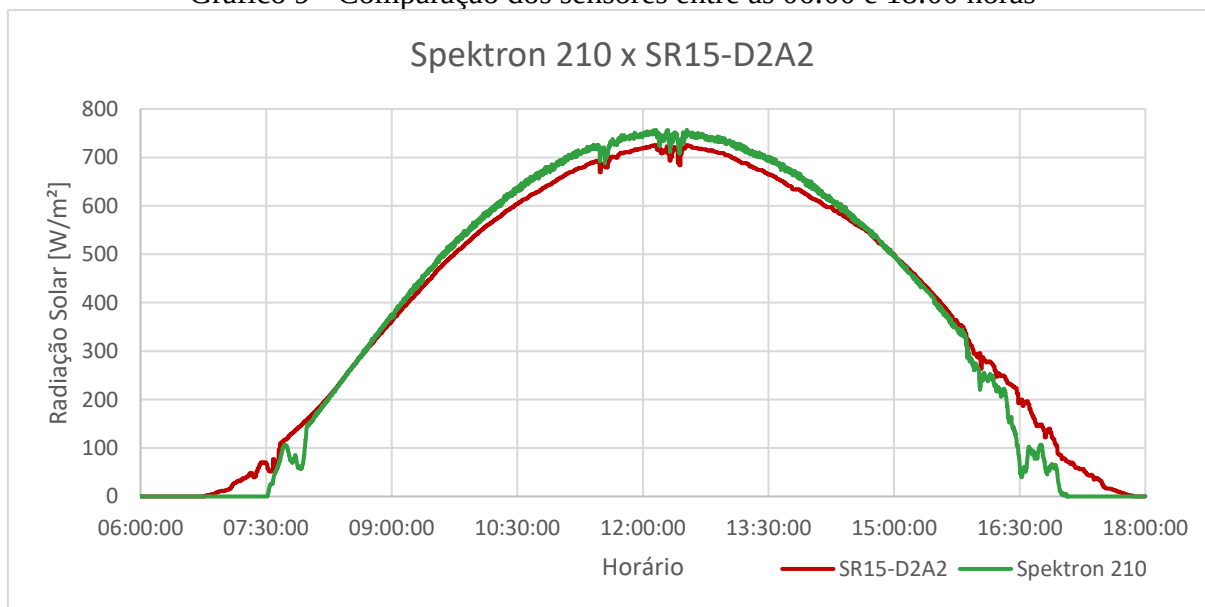
No Gráfico 8 é apresentado os valores do sensor SR15-D2A2 para o mesmo intervalo das 06:00 às 18:00 horas.

Gráfico 8 - Sensor SR15-D2A2 das 06:00 às 18:00 horas



A comparação entre as medições da radiação solar de ambos os sensores no intervalo das 06:00 horas às 18:00 horas do dia 22 de junho de 2022 é apresentado no Gráfico 9.

Gráfico 9 - Comparação dos sensores entre as 06:00 e 18:00 horas



Devido ao posicionamento diferente e com estruturas próximas ao sensor da estação de baixo custo, as medições acabam por se diferenciar para o amanhecer e anoitecer.

O sensor Spektron 210 também foi ajustado para o plano inclinado colocado em direção ao norte, como o projeto está localizado nas coordenadas 18°26'04.4S e -49°12'47.6"W a inclinação utilizada foi igual a latitude de 18° graus.

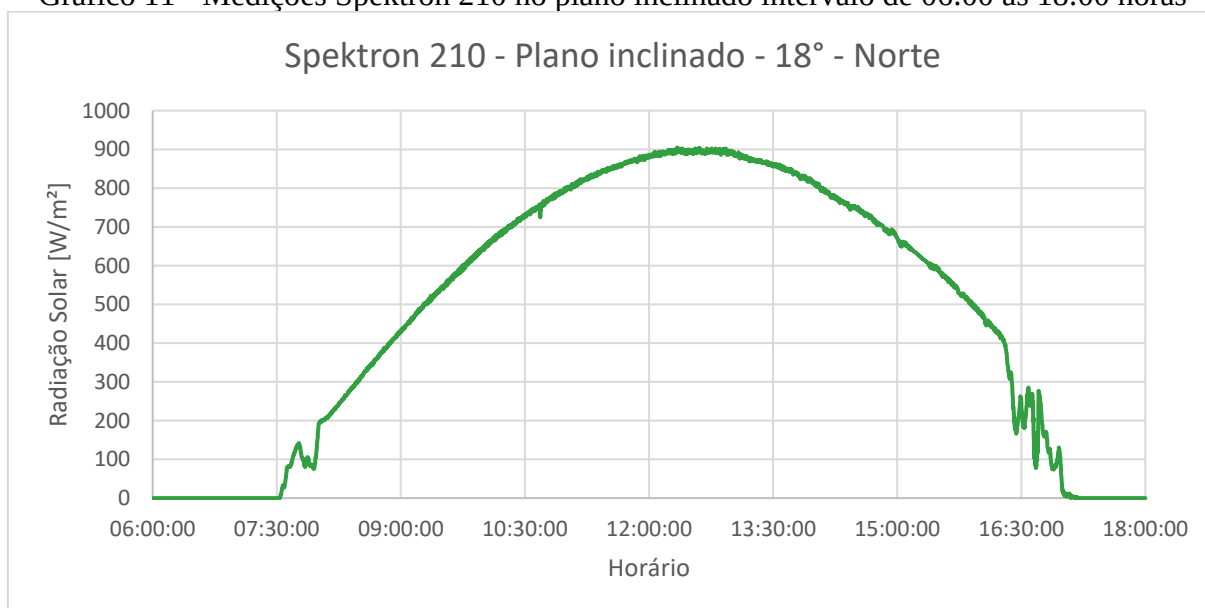
As novas medições de plano inclinado foram coletadas no dia 25 de junho de 2022 conforme apresentado no Gráfico 10.

Gráfico 10 - Medições Spektron 210 no plano inclinado em 18° graus



No Gráfico 11 é apresentado o intervalo de 06:00 às 18:00 horas para as medições no plano inclinado do dia 25 de junho de 2022.

Gráfico 11 - Medições Spektron 210 no plano inclinado intervalo de 06:00 às 18:00 horas



O sensor estando no plano inclinado obteve maiores valores de pico de potência por metro quadrado, o que seria o esperado para um ângulo igual ao da latitude, se comparada as medições de plano horizontal.

6.3.1.3 Resultados das medições da velocidade do vento

As medições da velocidade do vento do anemômetro de copos da WRFComercial são comparadas com as medições do anemômetro S-WSB-M003 localizado a 3,3 metros de altura em relação ao solo, ilustrado na Figura 191.

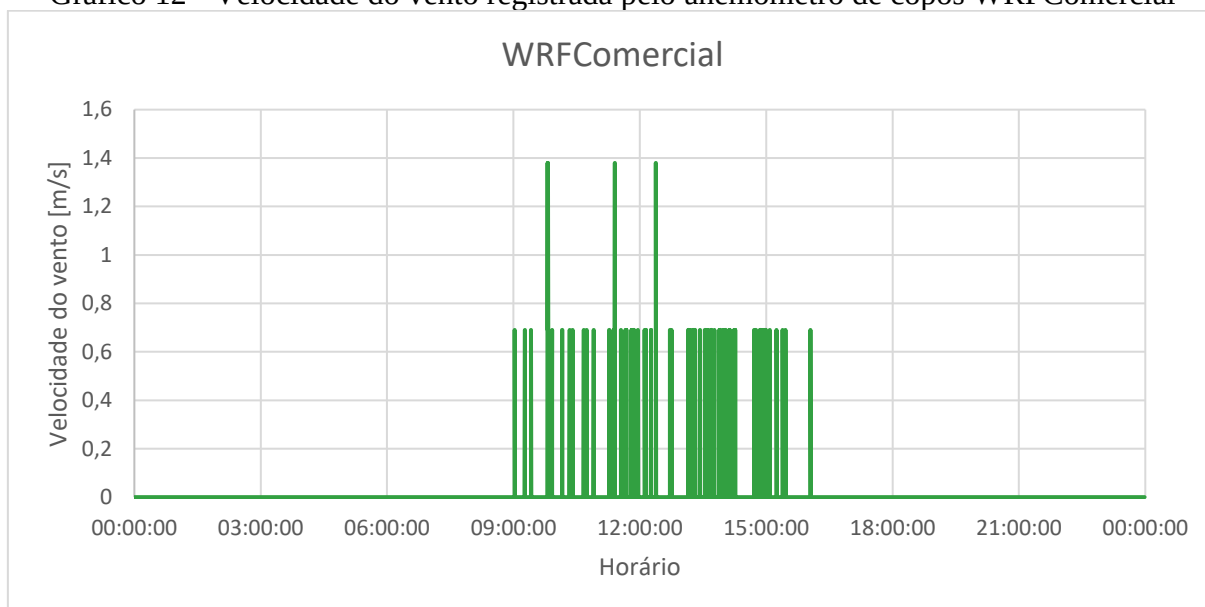
Figura 191 - Anemômetro S-WSB-M003



Fonte: Sigma Sensors [28]

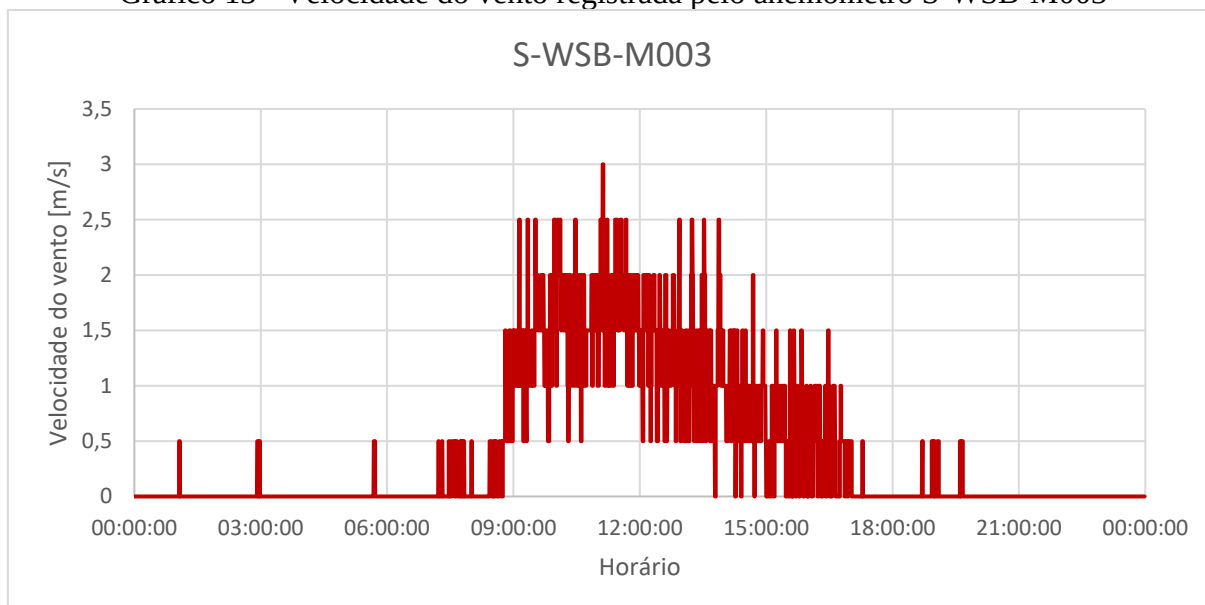
No Gráfico 12 é apresentada as medições da velocidade do vento realizadas pelo anemômetro de copos da WRFComercial no dia 22 de junho de 2022 em um intervalo de 24 horas.

Gráfico 12 - Velocidade do vento registrada pelo anemômetro de copos WRFComercial



No Gráfico 13 é apresentada as medições da velocidade de vento realizada pelo anemômetro S-WSB-M003 no dia 22 de junho de 2022 em um intervalo de 24 horas.

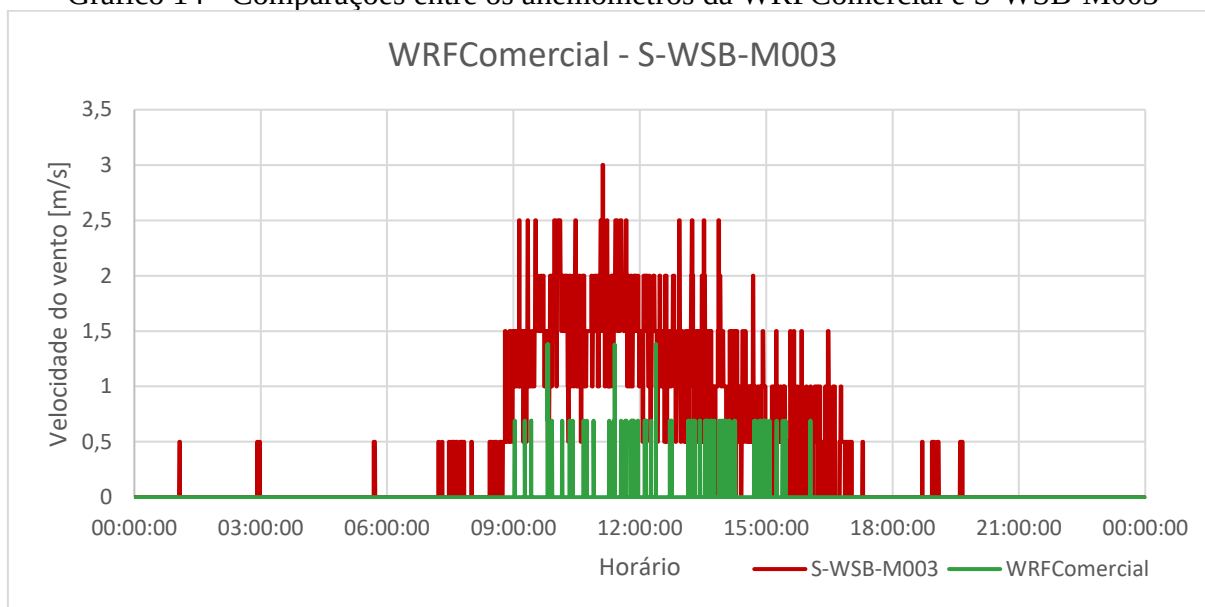
Gráfico 13 - Velocidade do vento registrada pelo anemômetro S-WSB-M003



No Gráfico 14 é apresentada as comparações entre as medições da velocidade do vento realizadas pelo anemômetro de copos da *WRFComercial* e o anemômetro de copos S-WSB-M003 no dia 22 de junho de 2022 para o mesmo intervalo de 24 horas.

Devido ao posicionamento diferente e os prédios próximos da estação solarimétrica de baixo custo, as medições apresentam variações.

Gráfico 14 - Comparações entre os anemômetros da WRFComercial e S-WSB-M003



6.3.2 Perda de dados

Devido a um atraso na troca de informações entre os dispositivos da estação solarimétrica de baixo custo, uma parcela dos dados acaba por não ser registrada.

O projeto registra 1 novo dado, para a temperatura, radiação solar e velocidade do vento a cada 10 segundos ao longo de um período de 24 horas, resultando em 8.640 valores de medição por dia, para cada grandeza.

Devido ao atraso alguns registros podem levar um pouco mais que 10 segundos.

Em média 8.191 valores de medição são registrados para cada arquivo de temperatura, radiação solar e velocidade do vento, sendo 449 valores a menos do que o esperado para um dia com o intervalo adotado de 10 segundos, resultando em uma perda de aproximadamente 5,19 % de dados.

$$\frac{(8640 - 8191)}{8640} * 100\% = 5,19 \%$$

O menor intervalo de medições possível para ser configurado na estação solarimétrica no padrão EPE é de 1 minuto, gerando no máximo 1.440 valores de medições por sensor, durante o mesmo intervalo de 24 horas.

Geralmente, não há perdas de dados na estação no padrão EPE, mesmo na ausência de *internet*, pois a mesma continua registrando em seu armazenamento interno até o reestabelecimento da conexão. Contudo, na presença de algum erro de comunicação interno durante o período de desconexão os dados armazenados também poderão ser perdidos.

6.4 ANÁLISE ECONÔMICA

A seguir são feitas comparação entre os valores da estação solarimétrica de baixa custo e a estação solarimétrica no padrão EPE do Instituto Federal de Goiás, campus Itumbiara. Na Tabela 1 é apresentado os valores de orçamento da estação solarimétrica no padrão EPE.

Tabela 1 - Orçamento da estação solarimétrica padrão EPE.

Descrição	Código	Quant.	Preço total (R\$)
Estação de monitoramento remoto	HOBO RX3000	1	6.846,82
Piranômetro de termopilha	SR15-D2A2	1	21.558,31
Suporte para o piranômetro na torre	SP200B	1	480,00
Sensor de temperatura e umidade relativa do ar	EE060HT1	1	2.854,20
Abrigo meteorológico para o sensor de temperatura e umidade relativa do ar	SES100	1	1.026,00
Sensor de velocidade do vento	S-WSB-M003	1	4.750,30
Suporte para fixação do anemômetro	SAN100	1	495,00
Modem GSM	SS-GSM	1	169,00
Torre de alumínio em treliça de 3 metros	TA3	1	3.350,00
Caixa selada IP66, com bateria 12V/12Amph, carregador de bateria morningstar, painel solar de 20 W, conectores inoxidáveis, kit de aterramento.	CPU-P20	1	6.920,00
Plataforma de monitoramento Dux Analicts anual	-	1	2.100,00
Valor total			50.549,63

Fonte: Elaborada pelo autor com base no orçamento da estação solarimétrica EPE (2022)

Na Tabela 2 é apresentado os valores de comercio da estação solarimétrica desenvolvida.

Tabela 2 - Orçamento da estação solarimétrica de baixo custo

Descrição	Código	Quant.	Preço total (R\$)
Anemômetro de Copos	WRFComercial	1	245,37
Sensor de Temperatura encapsulado	DS18B20	1	16,37
Sensor de irradiância solar	Spektron 210	1	1.400,00
Microcontrolador	ESP32	1	55,00
<i>Raspberry Pi</i>	Pi 3B V1.2	1	371,91
Fonte de alimentação para o <i>Raspberry Pi</i>	Fonte +5 VDC 3 A	1	38,00
Placa de Fenolite	PCB 15x10 cm	1	8,91
Resistor	10 k Ω	2	0,12
Resistor	100 k Ω	1	0,06
Potenciômetro linear	100 k Ω	1	2,53
Amplificador Operacional	LM324N	1	1,86
Fonte simétrica	Fonte +12V/- 12V e +5V/-5V	1	50,00
Caixa para armazenamento	Caixa	1	50,00
Valor total			2.240,13

Fonte: Elaborada pelo autor com base em preços de mercado (2022)

Em média a estação solarimétrica de baixo custo, vale 4,43 % do valor total da estação solarimétrica padrão EPE comparada.

7 CAPÍTULO VII – CONCLUSÃO

Com este projeto foi possível desenvolver e construir uma estação solarimétrica com equipamentos de baixo custo para a medição da temperatura com o sensor DS18B20, da radiação solar com o sensor Spektron 210 e velocidade do vento com o anemômetro de copos da WRFComercial.

Os sensores enviam suas medições ao microcontrolador ESP32 que por meio do código-fonte são manipuladas para valores legíveis a serem apresentados para visualização. Estas informações são publicadas em tópicos distintos no *broker Mosquitto* utilizando o protocolo MQTT com a transmissão dos dados feita pela conexão *Wi-Fi*.

O *broker Mosquitto* em execução no *Raspberry Pi* modelo 3B, acessado apenas com as credenciais de segurança, disponibiliza os tópicos para serem assinados pela aplicação *Node-RED*.

A aplicação *Node-RED* também em execução no *Raspberry Pi* modelo 3B é o meio adotado para a exibição dos dados publicados pelo *broker*, disponibilizando duas interfaces, uma para o acompanhamento das medições em tempo real e outra com os registros das medições históricos.

Pela análise gráfica é possível comparar a proximidade das medições feitas tanto pelo projeto desenvolvido quanto pela estação solarimétrica no padrão EPE e mesmo com o não registro de alguns dados ao longo do dia por causa do atraso na comunicação entre os dispositivos ainda é possível ter mais amostragem de dados se comparada as limitações da estação padrão EPE, além de ter maior liberdade nas configurações dos dados registrados.

A análise financeira indica que o valor do projeto completo fica abaixo do valor de custo de uma estação solarimétrica no padrão EPE e mesmo com equipamentos de baixo custo o projeto foi capaz de manter suas medições próximas as dos sensores de maior custo presentes na estação EPE.

O projeto em si oferece maior liberdade nas configurações dos dados, exibição de dados e o uso de diversas ferramentas atreladas ao conceito de *internet* das coisas que podem ser integrados no projeto desenvolvido.

As sugestões de melhoria incluem o adcionamento de mais sensores, como o sensor de umidade relativa do ar, que em conjunto com as medições de temperatura, seria possível calcular o valor do ponto de orvalho naquele local, a adição de outros sensores de temperatura de modelo DS18B20, que poderiam ser utilizados em uma única porta do microcontrolador

para mais pontos de medições da temperatura. A inclusão de novos sensores é possível pois o microcontrolador ESP32 dispõe de mais portas livres além das demais já utilizadas.

Com a adição de mais sensores ao projeto as interfaces poderiam ser completadas com um maior número de informações de grandezas físicas ao redor da estação. Outra sugestão de melhoria seria a adição de uma forma de backup dos dados sem custo, para preservação dos mesmos para a posterioridade.

Para trabalhos futuros é possível construir uma versão da estação desenvolvida em uma versão móvel ao invés de uma fixa, o que permitiria uma coleta de dados em vários pontos geográficos.

Em suma o trabalho oferece variadas opções de expansão e adaptabilidade para o desenvolvimento e construção de novas versões.

REFERÊNCIAS

1. BROCK, Fred; RICHARDSON, Scott. **Meteorological measurement systems**. 1. ed. Oklahoma: Oxford university press, 2001.
2. DURÃO, D; PEREIRA, J. Anemometria laser. **Gazeta de física**, v. 9, fasc. 3, p. 57-61, Jul. 1986.
3. FRANÇA, Fernando. **Instrumentação e medidas: grandezas mecânicas**. Campinas: UNICAMP, 2007.
4. BALBINOT, Alexandre; BRUSAMARELLO, Valner. **Instrumentação e fundamentos de medidas**. 2. ed. Vol. 1. São Paulo: LTC – Grupo Gen, 2011.
5. KERSCHBAUMER, Ricardo. **Microcontroladores**. Luzerna, 2018.
6. OpenJS foundation and contributors. **About**. Página About. Disponível em: <https://nodered.org>. Acesso em: 21 Jan. 2022.
7. WRFCOMERCIAL. **Conheça nossos produtos**. Disponível em: <https://www.wrfcomercial.com.br/equipamentos>. Acesso em: 02 Abr. 2022.
8. MINIPA. **MDA-10A**. Disponível em: <https://www.minipa.com.br/temperatura-e-ambiente/anemometros/135-mda-10a>. Acesso em: 02 Abr. 2022.
9. UNIVERSITÄT OLDENBURG. **Sphereanemometer**. Disponível em: <https://www.uol.de/en/twist/research/measurement/sphereanemometer>. Acesso em: 02 Abr. 2022.
10. AKROM. **Termo Anemômetro de fio quente – KR865**. Disponível em: <https://www.akrom.com.br/detalhes.php?codigo=426#>. Acesso em: 02 Abr. 2022.
11. NAUTIC EXPO. **Sensor cata-vento WindObserver 65**. Disponível em: <https://www.nauticexpo.com/pt/prod/gill-instruments-ltd/product-39900-485172.html>. Acesso em: 02 Abr. 2022.
12. DANTEC DYNAMICS. **Measurement principles of LDA**. Disponível em: <https://www.dantecdynamics.com/solutions-applications/solutions/fluid-mechanics/laser-doppler-anemometry-lda/measurement-principles-of-lda/>. Acesso em: 02 Abr. 2022.
13. SIKA. **Industrial thermometers with male thread/Special version vibration damped**. Disponível em: <https://www.sika.net/en/series/industrial-thermometers-with-male-thread-special-version-vibration-damped-271-291-bf/>. Acesso em: 02 Abr. 2022.
14. SANTHOSH. **Bimetal thermometer principle**. Disponível em: <https://engineerscommunity.com/t/bimetal-thermometer-principle/2635>. Acesso em: 03 Abr. 2022.

15. VISHAY. **NTCAPIPE3C9 Product information.** Disponível em: <https://www.vishay.com/product?docid=29166>. Acesso em: 03 Abr. 2022.
16. MINIPA. **MT-395A.** Disponível em: <https://www.minipa.com.br/temperatura-e-ambiente/termometros-infravermelho/363-mt-395a>. Acesso em: 03 Abr. 2022.
17. HUKSEFLUX. **Piranômetro para medição de radiação solar.** Disponível em: <https://huksefluxbrasil.com.br/piranometro>. Acesso em: 03 Abr. 2022.
18. EKO. **ML-01 Si-Pyranometer.** Disponível em: <https://eko-eu.com/products/solar-energy/si-pyranometers/ml-01-si-pyranometer>. Acesso em: 03 Abr. 2022.
19. TRITEC. **Spectron irradiance sensors.** Disponível em: <https://www.tritec-energy.com/en/sensors/spectron-irradiance-sensors/>. Acesso em: 03 Abr. 2022.
20. MICROCHIP. **PIC18F25K80.** Disponível em: <https://www.microchip.com/en-us/product/PIC18F25K80>. Acesso em: 03 Abr. 2022.
21. ESPRESSIF. **ESP32-DevKitM-1.** Disponível em: <https://docs.espressif.com/projects/espressif/en/latest/esp32/hw-reference/esp32/user-guide-devkitm-1.html>. Acesso em: 03 Abr. 2022.
22. RASPBERRY PI. **Raspberry Pi 4 Computer.** Disponível em: <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf>. Acesso em: 03 Abr. 2022.
23. VISUAL STUDIO. **Visual Studio Code.** Disponível em: <https://code.visualstudio.com/>. Acesso em: 04 Abr. 2022.
24. PLATFORMIO. **Professional collaborative platform for embedded development.** Disponível em: <https://platformio.org/>. Acesso em: 04 Abr. 2022.
25. MOSQUITTO. **Eclipse Mosquitto, an open source MQTT broker.** Disponível em: <https://mosquitto.org>. Acesso em 04 Abr. 2022.
26. MAXIM INTEGRATED. **DS18B20 Programmable Resolution 1-Wire Digital Thermometer.** Disponível em: <https://www.maximintegrated.com/en/products/sensors/DS18B20.html>. Acesso em 05 Jun. 2022.
27. ST. **LM324.** Disponível em: <https://www.st.com/en/amplifiers-and-comparators/lm324.html#overview>. Acesso em 05 jun. 2022.
28. SIGMA SENSORS. **Estação Solarimétrica EPE Onset com Telemetria de Dados.** Disponível em: <https://sigmasensors.com.br/>. Aceso em 05 Jun. 2022.

APÊNDICE 1 – CÓDIGO-FONTE ESP32 PARA O ANEMÔMETRO

```
#include <Arduino.h>
//-----Declarações para o anemômetro-----//
const int PinoInterrupcao = 4;
int Contador;
const int Intervalo = 10000;
float RaioAnemometro = 0.11;
float Freq;
float VelocidadeMetros;
float VelocidadeKilometros;
//-----//
//---Função Interrupção para o anemômetro--//
void IRAM_ATTR FuncaoInterrupcao() {
    Contador++;
}
//-----//
//-----Função do anemômetro-----//
void Anemometro() {
    Contador = 0;
    attachInterrupt(digitalPinToInterrupt(PinoInterrupcao), FuncaoInterrupcao, RISING);

    int TempoDoInicio = millis();
    while(millis() < TempoDoInicio + Intervalo) {
    }
    Serial.print("Interrupções: ");
    Serial.print(Contador);
    Serial.print(" ");
    if(Contador > 0) {
        Freq = (1000*Contador)/Intervalo;
    }
    else{
        Freq = 0;
    }
    VelocidadeMetros = 2 * PI * Freq * RaioAnemometro;
    VelocidadeKilometros = VelocidadeMetros * 3.6;
    Serial.print("Velocidade do Vento: ");
    Serial.print(VelocidadeMetros);
    Serial.print(" [m/s]");
    Serial.print(" ou ");
    Serial.print(VelocidadeKilometros);
```

```
    Serial.println(" [Km/h]");  
    Serial.println();  
}  
//-----//  
void setup() {  
    Serial.begin(115200);  
    pinMode(PinoInterrupcao, INPUT_PULLDOWN);  
}  
void loop() {  
    Anemometro();  
}
```

APÊNDICE 2 – CÓDIGO-FONTE ESP32 PARA O SENSOR DS18B20

```
#include <Arduino.h>
#include <OneWire.h>
#include <DallasTemperature.h>
//-----Declarações para o DS18B20-----//
const int Dados = 13;
float TemperaturaCelsius;
OneWire oneWire(Dados);
DallasTemperature Sensor(&oneWire);
//-----//
//-----Função do Termômetro-----//
void Termometro() {
  Sensor.requestTemperatures();
  TemperaturaCelsius = Sensor.getTempCByIndex(0);
  Serial.print("Temperatura: ");
  Serial.print(TemperaturaCelsius);
  Serial.println(" °C");
  Serial.println();
}
//-----//
void setup() {
  Serial.begin(115200);
}
void loop() {
  Termometro();
}
```

APÊNDICE 3 – CÓDIGO-FONTE ESP32 PARA O SENSOR DE IRRADIÂNCIA SOLAR

```
#include <Arduino.h>
//-----Declarações do piranometro-----//
const int MiliVoltsLeitura = 34;
float Resolucao = 0;
float RadiacaoWattsMetroQuadrado = 0;
float ValorMiliVoltsRadiacao = 0;
//-----//
//-----Função do piranometro-----//
void RadiacaoSolar() {
    analogReadResolution(12);
    Resolucao = analogRead(MiliVoltsLeitura);
    ValorMiliVoltsRadiacao = 3.3 * (Resolucao/4095);
    RadiacaoWattsMetroQuadrado = (1000)* (ValorMiliVoltsRadiacao/(2.55684));
    //3.3*0.07748/0.1 = 2.55684
    Serial.print("Radiacao Solar: ");
    Serial.print(RadiacaoWattsMetroQuadrado);
    Serial.println(" W/m²");
    Serial.println();
}
//-----//
void setup() {
    Serial.begin(115200);
}
void loop() {
    RadiacaoSolar();
}
```

APÊNDICE 4 – CÓDIGO-FONTE ESP32 PARA A CONEXÃO WIFI

```
#include <Arduino.h>
#include <WiFi.h>
//-----Declarações para conexão WiFi-----//
const char* ssid = "usuario";
const char* senha = "senha";
const int Led_Azul = 2;
//-----//
//-----Função para conexão WiFi-----//
void Conectar_WiFi() {
    if(WiFi.status() == WL_CONNECTED)
        return;
    digitalWrite(Led_Azul, LOW);
    Serial.println();
    Serial.print("MAC: ");
    Serial.println(WiFi.macAddress());
    Serial.print("Conectando ao SSID: ");
    Serial.println(ssid);
    WiFi.begin(ssid, senha);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    digitalWrite(Led_Azul, HIGH);
    Serial.println();
    Serial.print("Conecatado a rede WiFi: ");
    Serial.println(ssid);
    Serial.print("Endereço IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();
}
//-----//
void setup() {
    Serial.begin(115200);
    pinMode(Led_Azul, OUTPUT);
}
void loop() {
    Conectar_WiFi();
}
```

APÊNDICE 5 – CÓDIGO-FONTE ESP32 PARA CONEXÃO COM BROKER MOSQUITTO

```
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>
//-----Declarações para conexão MQTT-----//
const char* MQTT_ID = "ESP32-PUBLICADOR";
const char* MQTT_Broker = "192.168.1.186";
const int MQTT_Porta = 1883;
const char* MQTT_Usuario = "usuario";
const char* MQTT_Senha = "senha";
const char* Topico_Anemometro_IFG = "Vento-IFG-Itumbiara";
const char* Topico_Termometro_IFG = "Temperatura-IFG-Itumbiara";
const char* Topico_Piranometro_IFG = "Radiacao-IFG-Itumbiara";
WiFiClient espClient;
PubSubClient MQTT(espClient);
char Temperatura[10];
char Velocidade[10];
char Radiacao[10];
//-----//
//-----Função para conexão MQTT-----//
void Conectar_MQTT() {
    MQTT.setServer(MQTT_Broker,MQTT_Porta);
    while (!MQTT.connected())
    {
        Serial.print("Tentando conexão com o Broker MQTT: ");
        Serial.println(MQTT_Broker);
        if (MQTT.connect(MQTT_ID,MQTT_Usuario,MQTT_Senha)) {
            Serial.println("Conecatdo ao broker MQTT!");
            Serial.println();
        }
        else{
            Serial.print("Falha na conexão com erro: ");
            Serial.println(MQTT.state());
            Serial.println("Nova tentativa de conexão ao Broker MQTT em 5 segundos.");
            Serial.println();
            delay(5000);
        }
    }
}
```

```

    }
}
//-----//
//-----Função para publicar no broker-----//
void Publicar_MQTT() {
    String str1 = String(VelocidadeMetros);
    String str2 = String(TemperaturaCelsius);
    String str3 = String(RadiacaoWattsMetroQuadrado);
    str1.toCharArray(Velocidade,10);
    str2.toCharArray(Temperatura,10);
    str3.toCharArray(Radiacao,10);
    MQTT.publish(Topico_Anemometro_IFG,Velocidade);
    MQTT.publish(Topico_Termometro_IFG,Temperatura);
    MQTT.publish(Topico_Piranometro_IFG,Radiacao);
}
//-----//
void setup() {
    Serial.begin(115200);
}
void loop() {
    Conectar_MQTT();
    Publicar_MQTT();
}

```

APÊNDICE 6 – CÓDIGO DO MICROCONTROLADOR ESP32 COMPLETO

```
#include <Arduino.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <WiFi.h>
#include <PubSubClient.h>
//-----Declarações para o anemômetro-----//
const int PinoInterrupcao = 4;
int Contador;
const int Intervalo = 10000;
float RaioAnemometro = 0.11;
float Freq;
float VelocidadeMetros;
float VelocidadeKilometros;
//-----//
//---Função Interrupção para o anemômetro---//
void IRAM_ATTR FuncaoInterrupcao() {
    Contador++;
}
//-----//
//-----Função do anemômetro-----//
void Anemometro() {
    Contador = 0;
    attachInterrupt(digitalPinToInterrupt(PinoInterrupcao), FuncaoInterrupcao, RISING);

    int TempoDoInicio = millis();
    while(millis() < TempoDoInicio + Intervalo) {
    }
    Serial.print("Interrupções: ");
    Serial.print(Contador);
    Serial.print(" ");
    if(Contador > 0) {
        Freq = (1000*Contador)/Intervalo;
    }
    else{
        Freq = 0;
    }
    VelocidadeMetros = 2 * PI * Freq * RaioAnemometro;
    VelocidadeKilometros = VelocidadeMetros * 3.6;
```

```

    Serial.print("Velocidade do Vento: ");
    Serial.print(VelocidadeMetros);
    Serial.print(" [m/s]");
    Serial.print(" ou ");
    Serial.print(VelocidadeKilometros);
    Serial.println(" [Km/h]");
    Serial.println();
}
//-----//
//-----Declarações para o DS18B20-----//
const int Dados = 13;
float TemperaturaCelsius;
OneWire oneWire(Dados);
DallasTemperature Sensor(&oneWire);
//-----//
//-----Função do Termômetro-----//
void Termometro(){
    Sensor.requestTemperatures();
    TemperaturaCelsius = Sensor.getTempCByIndex(0);
    Serial.print("Temperatura: ");
    Serial.print(TemperaturaCelsius);
    Serial.println(" °C");
    Serial.println();
}
//-----//
//-----Declarações do piranometro-----//
const int MiliVoltsLeitura = 34;
float Resolucao = 0;
float RadiacaoWattsMetroQuadrado = 0;
float ValorMiliVoltsRadiacao = 0;
//-----//
//-----Função do piranometro-----//
void RadiacaoSolar(){
    analogReadResolution(12);
    Resolucao = analogRead(MiliVoltsLeitura);
    ValorMiliVoltsRadiacao = 3.3 * (Resolucao/4095);
    RadiacaoWattsMetroQuadrado = (1000)* (ValorMiliVoltsRadiacao/(2.55684));
    //3.3*0.07748/0.1 = 2.55684
    Serial.print("Radiacao Solar: ");
    Serial.print(RadiacaoWattsMetroQuadrado);
    Serial.println(" W/m²");
}

```

```

    Serial.println();
}
//-----//
//-----Declarações para conexão WiFi-----//
const char* ssid = "usuario";
const char* senha = "senha";
const int Led_Azul = 2;
//-----//
//-----Função para conexão WiFi-----//
void Conectar_WiFi() {
    if(WiFi.status() == WL_CONNECTED)
        return;
    digitalWrite(Led_Azul, LOW);
    Serial.println();
    Serial.print("MAC: ");
    Serial.println(WiFi.macAddress());
    Serial.print("Conectando ao SSID: ");
    Serial.println(ssid);
    WiFi.begin(ssid, senha);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    digitalWrite(Led_Azul, HIGH);
    Serial.println();
    Serial.print("Conecatado a rede WiFi: ");
    Serial.println(ssid);
    Serial.print("Endereço IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();
}
//-----//
//-----Declarações para conexão MQTT-----//
const char* MQTT_ID = "ESP32-PUBLICADOR";
const char* MQTT_Broker = "192.168.1.186";
const int MQTT_Porta = 1883;
const char* MQTT_Usuario = "usuario";
const char* MQTT_Senha = "senha";
const char* Topico_Anemometro_IFG = "Vento-IFG-Itumbiara";
const char* Topico_Termometro_IFG = "Temperatura-IFG-Itumbiara";
const char* Topico_Piranometro_IFG = "Radiacao-IFG-Itumbiara";

```

```

WiFiClient espClient;
PubSubClient MQTT(espClient);
char Temperatura[10];
char Velocidade[10];
char Radiacao[10];
//-----//
//-----Função para conexão MQTT-----//
void Conectar_MQTT() {
    MQTT.setServer(MQTT_Broker,MQTT_Porta);
    while (!MQTT.connected())
    {
        Serial.print("Tentando conexão com o Broker MQTT: ");
        Serial.println(MQTT_Broker);

        if (MQTT.connect(MQTT_ID,MQTT_Usuario,MQTT_Senha)) {
            Serial.println("Conecatdo ao broker MQTT!");
            Serial.println();
        }
        else{
            Serial.print("Falha na conexão com erro: ");
            Serial.println(MQTT.state());
            Serial.println("Nova tentativa de conexão ao Broker MQTT em 5
segundos.");
            Serial.println();
            delay(5000);
        }
    }
}
//-----//
//-----Função para publicar no broker-----//
void Publicar_MQTT() {
    String str1 = String(VelocidadeMetros);
    String str2 = String(TemperaturaCelsius);
    String str3 = String(RadiacaoWattsMetroQuadrado);
    str1.toCharArray(Velocidade,10);
    str2.toCharArray(Temperatura,10);
    str3.toCharArray(Radiacao,10);
    MQTT.publish(Topico_Anemometro_IFG,Velocidade);
    MQTT.publish(Topico_Termometro_IFG,Temperatura);
    MQTT.publish(Topico_Piranometro_IFG,Radiacao);
}

```

```
//-----//  
void setup() {  
    Serial.begin(115200);  
    pinMode(PinoInterrupcao, INPUT_PULLDOWN);  
    pinMode(Led_Azul, OUTPUT);  
}  
void loop() {  
    Conectar_WiFi();  
    Conectar_MQTT();  
    Anemometro();  
    Termometro();  
    RadiacaoSolar();  
    Publicar_MQTT();  
}
```

APÊNDICE 7 – CÓDIGO-FONTE PARA NODE CONFIGURANDO-DADOS

```
var agora = new Date();

var ano = agora.getFullYear();
var mes = agora.getMonth() < 9 ? "0" + (agora.getMonth() + 1) :
(agora.getMonth() + 1); // getMonth() inicia com zero como padrão
var dia = agora.getDate() < 10 ? "0" + agora.getDate() : agora.getDate();
var horas = agora.getHours() < 10 ? "0" + agora.getHours() : agora.getHours();
var minutos = agora.getMinutes() < 10 ? "0" + agora.getMinutes() :
agora.getMinutes();
var segundos = agora.getSeconds() < 10 ? "0" + agora.getSeconds() :
agora.getSeconds();

msg.payload = {
  "Temperatura": msg.payload,
  "Dia": dia,
  "Mes": mes,
  "Ano": ano,
  "Horas": horas,
  "Minutos": minutos,
  "Segundos": segundos,
  "Tempo": agora.getTime()
}
return msg;
```

APÊNDICE 8 – CÓDIGO-FONTE PARA NODE GERADOR-NOME-DE-ARQUIVO

```
var agora = new Date();
var ano = agora.getFullYear();
var mes = agora.getMonth() < 9 ? "0" + (agora.getMonth() + 1) :
(agora.getMonth() + 1); // getMonth() inicia com zero como padrão
var dia = agora.getDate() < 10 ? "0" + agora.getDate() : agora.getDate();
var horas = agora.getHours() < 10 ? "0" + agora.getHours() : agora.getHours();
var minutos = agora.getMinutes() < 10 ? "0" + agora.getMinutes() :
agora.getMinutes();
var segundos = agora.getSeconds() < 10 ? "0" + agora.getSeconds() :
agora.getSeconds();

msg.fname = "Temperatura_" + dia + "_" + mes + "_" + ano + ".csv";

msg.filename = "/home/iffontes/Medidas/Temperatura/" + msg.fname;

msg.filecontent = msg.payload;

msg.payload = {"pattern":msg.fname};

node.status({fill:"blue",shape:"ring",text:msg.fname});

return msg;
```

APÊNDICE 9 – CÓDIGO-FONTE PARA NODE LISTA-NOME-ARQUIVO

```
for (var i=0; i<msg.payload.length; i++) {  
    msg.payload[i].fname = msg.payload[i].name.replace(/^.*(\\|\/|:)/, '');  
}  
return msg;
```

APÊNDICE 10 – CÓDIGO-FONTE PARA NODE FORMATO

```
msg.options = [];  
for (var i=0; i<msg.payload.length; i++){  
    obj = {};  
    obj      [msg.payload[i].name.replace(/^.*(\\|\/|\\:)/,      '')]      =  
msg.payload[i].name  
    msg.options.push(obj);  
}  
msg.payload={};  
return msg;
```

APÊNDICE 11 – CÓDIGO-FONTE PARA NODE SALVAR

```
flow.set("ArquivoSelecioneadoTemperatura",msg.payload);  
return msg;
```

APÊNDICE 12 – CÓDIGO-FONTE PARA NODE PEGAR-NOME-ARQUIVO

```
let NomeArquivo = flow.get("ArquivoSelecioneadoTemperatura");  
if(NomeArquivo == undefined){  
    return;  
}  
msg.filename = NomeArquivo;  
return msg;
```

APÊNDICE 13 – CÓDIGO-FONTE PARA NODE FORMATAR-DADOS-PARA- GRAFICO

```
var Grafico = [{  
    "series": ["Temperatura"],  
    "data": [],  
}];  
  
var Temperatura = [];  
  
for(var i=0; i<msg.payload.length; i++){  
    Temperatura.push({"x":msg.payload[i].Tempo,  
"y":msg.payload[i].Temperatura});  
}  
  
Grafico[0].data.push(Temperatura);  
  
msg.payload = Grafico;  
  
return msg;
```

APÊNDICE 14 – CÓDIGO-FONTE PARA NODE NOME-DO-ARQUIVO

```
msg.filename = msg.req.query.filename;
msg.contentdisposition = "attachment; filename=\"" +
msg.req.query.filename.replace(/^.*(\\|\/|\/|:)/, '') + "\"";
return msg;
```

APÊNDICE 15 - CÓDIGO-FONTE PARA O NODE-RED COMPLETO COMPACTADO

```
[{"id":"d0b1c77bddc9db6d","type":"tab","label":"InterfaceFinal","disabled":
false,"info":""},{ "id":"d2ed0d9e5ae237bc","type":"mqtt
in","z":"d0b1c77bddc9db6d","name":"","topic":"Temperatura-IFG-
Itumbiara","qos":"2","datatype":"auto","broker":"c3c5e6fb9490c966","nl":fal
se,"rap":true,"rh":0,"inputs":0,"x":130,"y":180,"wires":[["021ad48d1dd6fe30
","eb9a7e0fad67b40d"]]}, {"id":"021ad48d1dd6fe30","type":"ui_chart","z":"d0b
1c77bddc9db6d","name":"","group":"e9cdc14bb3de0c65","order":2,"width":"20",
"height":"6","label":"Temperatura
[Celsius]","chartType":"line","legend":"true","xformat":"HH:mm:ss","interpo
late":"linear","nodata":"","dot":false,"ymin":"0","ymax":"50","removeOlder"
:"24","removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneCol
or":false,"useUTC":false,"colors":["#32a041","#aec7e8","#ff7f0e","#32a041",
"#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferen
tColor":false,"className":"","x":390,"y":200,"wires":[[]]}, {"id":"d12f088b0
478e856","type":"mqtt in","z":"d0b1c77bddc9db6d","name":"","topic":"Vento-
IFG-
Itumbiara","qos":"2","datatype":"auto","broker":"c3c5e6fb9490c966","nl":fal
se,"rap":true,"rh":0,"inputs":0,"x":150,"y":380,"wires":[["0b06b345b60cfdb2
","22156791ef3708eb"]]}, {"id":"0b06b345b60cfdb2","type":"ui_gauge","z":"d0b
1c77bddc9db6d","name":"","group":"27fd740fbc6ce760","order":1,"width":"8","
height":"6","gtype":"gage","title":"Velocidade do Vento
[m/s]","label":"m/s","format":"{{value}}","min":0,"max":"10","colors":["#32
a041","#e6e600","#ca3838"],"seg1":"8","seg2":"9","className":"","x":400,"y"
:360,"wires":[[]]}, {"id":"22156791ef3708eb","type":"ui_chart","z":"d0b1c77bdd
c9db6d","name":"","group":"27fd740fbc6ce760","order":2,"width":"20","height
":"6","label":"Velocidade do Vento
[m/s]","chartType":"line","legend":"true","xformat":"HH:mm:ss","interpolate
":"linear","nodata":"","dot":false,"ymin":"0","ymax":"10","removeOlder":"24
","removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":
false,"useUTC":false,"colors":["#32a041","#aec7e8","#ff7f0e","#2ca02c","#98
df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentCol
or":false,"className":"","x":400,"y":400,"wires":[[]]}, {"id":"64f2c86bba05c
dc3","type":"ui_template","z":"d0b1c77bddc9db6d","group":"fbf8c256b7b80382"
,"name":"Template Informações do Projeto e
IFG","order":5,"width":"23","height":"5","format":"<!DOCTYPE
html>\n\n<style>\n\nna:link {\n color: #2f9e41;\n background-color:
transparent;\n text-decoration: none;\n}\n\n/* width */\n::-webkit-
```

```

scrollbar {\n width: 10px;\n}\n\n/* Track */\n::-webkit-scrollbar-track {\n
background: #f1f1f1; \n}\n\n/* Handle */\n::-webkit-scrollbar-thumb {\n
background: #2f9e41; \n}\n\n/* Handle on hover */\n::-webkit-scrollbar-
thumb:hover {\n background: #2f9e41; \n}\n</style>\n\n<html>\n    <body>\n
<h1>Trabalho de Conclusão de Curso</h1>\n        <p>Orientador: Prof. Dr.
Ghunter Paulo Viajante</p>\n                <p>Aluno: Gabriel Viríssimo Queiroz
Lopes</p>\n                <p></p>\n                <h3>Projeto IoT (Internet of Things -
Internet das Coisas) - Laboratório de Fontes Renováveis</h3>\n                <p>Sensor
de Temperatura.</p>\n                <p>Sensor de Velocidade do Vento.</p>\n
<p>Sensor                de                Irradiância                Solar.</p>\n
</body>\n</html>","storeOutMessages":true,"fwdInMessages":true,"resendOnRef
resh":true,"templateScope":"local","className":"","x":320,"y":100,"wires":[
[]]},{ "id":"4b385916aad4b52c","type":"mqtt
in","z":"d0b1c77bddc9db6d","name":"","topic":"Radiacao-IFG-
Itumbiara","qos":"2","datatype":"auto","broker":"c3c5e6fb9490c966","nl":fal
se,"rap":true,"rh":0,"inputs":0,"x":140,"y":280,"wires":[["26e6f3657811e33c
","991e9ffa000e0215"]]}, {"id":"991e9ffa000e0215","type":"ui_gauge","z":"d0b
1c77bddc9db6d","name":"","group":"3d6c2c5a88b96f7c","order":1,"width":"8","
height":"6","gtype":"gage","title":"Radiação                Solar                -
[W/m²]","label":"W/m²","format":"{{value}}","min":0,"max":"1500","colors":[
"#32a041","#e6e600","#ca3838"],"seg1":"900","seg2":"1000","className":"","x
":390,"y":260,"wires":[]},{ "id":"26e6f3657811e33c","type":"ui_chart","z":"d
0b1c77bddc9db6d","name":"","group":"3d6c2c5a88b96f7c","order":2,"width":"20
","height":"6","label":"Radiação                Solar                -
[W/m²]","chartType":"line","legend":"true","xformat":"HH:mm:ss","interpolat
e":"linear","nodata":"","dot":false,"ymin":"0","ymax":"1500","removeOlder":
"24","removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColo
r":false,"useUTC":false,"colors":["#32a041","#aec7e8","#ff7f0e","#2ca02c",
"#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferent
Color":false,"className":"","x":390,"y":300,"wires":[[]]},{ "id":"6ae8e3367d
5fd801","type":"ui_media","z":"d0b1c77bddc9db6d","group":"fbf8c256b7b80382
","name":"Logo-
IFG","width":"5","height":"5","order":1,"category":"LogoIFG","file":"Logo-
IFG-Vertical-
Itumbiara.png","layout":"adjust","showcontrols":true,"loop":true,"onstart":
false,"scope":"local","tooltip":"","x":80,"y":100,"wires":[["64f2c86bba05cd
c3"]]}, {"id":"995acd3fe4642214","type":"ui_media","z":"d0b1c77bddc9db6d","g
roup":"cfa359d4106e224f","name":"Logo-
IFG","width":"5","height":"5","order":1,"category":"LogoIFG","file":"Logo-
IFG-Vertical-
Itumbiara.png","layout":"adjust","showcontrols":true,"loop":true,"onstart":

```

```
false,"scope":"local","tooltip":"","x":80,"y":520,"wires":[["8fc754aa935335ac"]]}, {"id":"12dbcecf43930c18","type":"comment","z":"d0b1c77bddc9db6d","name":"Interface -> Laboratório - Histórico","info":"","x":160,"y":460,"wires":[]}, {"id":"4355f6219e27601e","type":"ui_button","z":"d0b1c77bddc9db6d","name":"","group":"a34f35553fe7bed7","order":4,"width":"6","height":"1","passthru":false,"label":"Atualizar","tooltip":"","color":"","bgcolor":"","className":"","icon":"refresh","payload":"","payloadType":"str","topic":"topic","topicType":"msg","x":80,"y":1320,"wires":[["d6be7b3b85bc58ea","e5965772c61996c9","a97808fa9676965e"]]}, {"id":"d6be7b3b85bc58ea","type":"fs-file-list","z":"d0b1c77bddc9db6d","name":"","start":"/home/iffontes/Medidas/Vento","pattern":"*.*","folders":"*","hidden":true,"lstype":"files","path":true,"single":true,"depth":0,"stat":true,"showWarnings":true,"x":440,"y":1320,"wires":[["e1c65a8583218e5c","578b3649d68658cd"]]}, {"id":"785bff3a6baf9c24","type":"ui_chart","z":"d0b1c77bddc9db6d","name":"","group":"a34f35553fe7bed7","order":3,"width":"22","height":"5","label":"Histórico da Velocidade do Vento [m/s]","chartType":"line","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":false,"ymin":"0","ymax":"2","removeOlder":"24","removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"useUTC":false,"colors":["#32a041","#aec7e8","#ff7f0e","#32a041","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferentColor":false,"className":"","x":1720,"y":1500,"wires":[[]]}, {"id":"c27efcf55b2ade21","type":"ui_button","z":"d0b1c77bddc9db6d","name":"","group":"a34f35553fe7bed7","order":8,"width":"4","height":"1","passthru":false,"label":"Visualizar Gráfico","tooltip":"","color":"","bgcolor":"","className":"","icon":"show_chart","payload":"","payloadType":"str","topic":"topic","topicType":"msg","x":470,"y":1500,"wires":[["636a0627d02cb4b9"]]}, {"id":"c7014d9f1e428041","type":"ui_template","z":"d0b1c77bddc9db6d","group":"a34f35553fe7bed7","name":"Template Histórico","order":2,"width":"6","height":"5","format":"<style>\n.myDiv {\nborder-radius: 25px;\n  border: 3px solid #2f9e41;\n  padding: 20px; \nwidth: auto;\n  height: auto; \n}\n\nna:link {\n  color: #2f9e41;\nbackground-color: transparent;\n  text-decoration: none;\n}\n\n/* width */\n::-webkit-scrollbar {\n  width: 10px;\n}\n\n/* Track */\n::-webkit-scrollbar-track {\n  background: #f1f1f1; \n}\n\n/* Handle */\n::-webkit-scrollbar-thumb {\n  background: #2f9e41; \n}\n\n/* Handle on hover */\n::-webkit-scrollbar-thumb:hover {\n  background: #2f9e41; \n}\n</style>\n\n<div class=\"myDiv\">\n\nng-bind-html=\"msg.payload\"><br/>\n</div>","storeOutMessages":true,"fwdInMessages":true}
```

```

: true, "resendOnRefresh": true, "templateScope": "local", "className": "", "x": 109
0, "y": 1320, "wires": [ [] ], { "id": "503a3c840e5a89df", "type": "template", "z": "d0
b1c77bddc9db6d", "name": "Template
Tabela", "field": "payload", "fieldType": "msg", "format": "handlebars", "syntax":
"mustache", "template": "<table                                width=\"100%\">\n
<tr><th>VelocidadeVento_Dia_Mês_Ano</th></tr>\n                {{#payload}}\n
<tr>\n                                <td><a
href=\"\"/home/iffontes/Medidas/Vento/?filename={{name}}\"
target=\"blank\">{{fname}}</a></td>\n                                </tr>\n
{{/payload}}\n</table>", "output": "str", "x": 880, "y": 1320, "wires": [ [ "c7014d9f
1e428041" ] ], { "id": "e1c65a8583218e5c", "type": "function", "z": "d0b1c77bddc9db
6d", "name": "ListaNomeArquivo", "func": "for (var i=0; i<msg.payload.length;
i++)          {\n                                msg.payload[i].fname          =
msg.payload[i].name.replace(/^.*(\\\\\\|\\\\/|\\\\:)/,          '');\n\nreturn
msg;", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 670
, "y": 1320, "wires": [ [ "503a3c840e5a89df" ] ], { "id": "636a0627d02cb4b9", "type": "
function", "z": "d0b1c77bddc9db6d", "name": "PegarNomeArquivo", "func": "let
NomeArquivo          =
flow.get(\"ArquivoSelecionadoVelocidadeVento\");\nif (NomeArquivo          ==
undefined) {\n          return;\n}\nmsg.filename    =    NomeArquivo;\nreturn
msg;", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 890
, "y": 1500, "wires": [ [ "c0fe087e06bd1511" ] ], { "id": "c0fe087e06bd1511", "type": "
file
in", "z": "d0b1c77bddc9db6d", "name": "", "filename": "", "format": "utf8", "chunk":
false, "sendError": false, "encoding": "none", "allProps": false, "x": 1070, "y": 150
0, "wires": [ [ "fb35c06b1abd7026" ] ], { "id": "fb35c06b1abd7026", "type": "csv", "z"
: "d0b1c77bddc9db6d", "name": "", "sep": ",", "hdrin": true, "hdrout": "none", "multi
": "mult", "ret": "\\n", "temp": "", "skip": "0", "strings": true, "include_empty_str
ings": "", "include_null_values": "", "x": 1210, "y": 1500, "wires": [ [ "740ca529a74f
1288" ] ], { "id": "740ca529a74f1288", "type": "function", "z": "d0b1c77bddc9db6d",
"name": "FormatarDadosParaGráfico", "func": "var Grafico = [{\n    \"series\":
[\"VelocidadeVento\"],\n    \"data\": [],\n}];\n\nvar VelocidadeVento =
[];\n\nfor(var          i=0;          i<msg.payload.length;          i++){\n
VelocidadeVento.push({\"x\":msg.payload[i].Tempo,
\"y\":msg.payload[i].VelocidadeVento});\n}\n\nGrafico[0].data.push(Velocida
deVento);\n\nmsg.payload          =          Grafico;\n\nreturn
msg;", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 142
0, "y": 1500, "wires": [ [ "785bff3a6baf9c24" ] ], { "id": "ee09b6abb04e8434", "type":
"http
in", "z": "d0b1c77bddc9db6d", "name": "", "url": "/home/iffontes/Medidas/Vento/",
"method": "get", "upload": false, "swaggerDoc": "", "x": 520, "y": 1440, "wires": [ [ "5

```

```

51189d8e937ce82"]]],{"id":"551189d8e937ce82","type":"function","z":"d0b1c77
bddc9db6d","name":"NomeDoArquivo","func":"msg.filename
=
msg.req.query.filename;\nmsg.contentdisposition
=
\"attachment;
filename=\\\"\\\" + msg.req.query.filename.replace(/^.*(\\\\\\|\\\\/|\\\\:)/, '') +
\\\"\\\"\\\";\nreturn
msg;\",\"outputs\":1,\"noerr\":0,\"initialize\":\"\",\"finalize\":\"\",\"libs\":[],\"x\":880
,\"y\":1440,\"wires\":[[\"40f1546053526969\"]]],{"id":"40f1546053526969","type":"
file
in\",\"z\":\"d0b1c77bddc9db6d\",\"name\":\"\",\"filename\":\"\",\"format\":\"\",\"chunk\":fals
e,\"sendError\":false,\"encoding\":\"none\",\"allProps\":false,\"x\":1070,\"y\":1440,\"w
ires\":[[\"159d7ad753439289\"]]],{"id":"159d7ad753439289","type":"change\",\"z\":
\"d0b1c77bddc9db6d\",\"name\":\"\",\"rules\":[{\"t\":\"set\",\"p\":\"headers\",\"pt\":\"msg\",\"
to\":\"{}\",\"tot\":\"json\"},{\"t\":\"set\",\"p\":\"headers.content-
type\",\"pt\":\"msg\",\"to\":\"txt/csv\",\"tot\":\"str\"},{\"t\":\"set\",\"p\":\"headers.Conten
t-
Disposition\",\"pt\":\"msg\",\"to\":\"contentdisposition\",\"tot\":\"msg\"}],\"action\":\"
\",\"property\":\"\",\"from\":\"\",\"to\":\"\",\"reg\":false,\"x\":1240,\"y\":1440,\"wires\":[[\"e
5bc6975f4a52be5\"]]],{"id":"e5bc6975f4a52be5","type\":\"http
response\",\"z\":\"d0b1c77bddc9db6d\",\"name\":\"\",\"statusCode\":\"\",\"headers\":{\"x\"
:1410,\"y\":1440,\"wires\":[]},{\"id\":\"080a0b46c2657e7e\",\"type\":\"ui_dropdown\",\"z
\":\"d0b1c77bddc9db6d\",\"name\":\"\",\"label\":\"Arquivo:\",\"tooltip\":\"\",\"place\":\"Sel
ecione um arquivo de Velocidade do
Vento\",\"group\":\"a34f35553fe7bed7\",\"order\":5,\"width\":\"18\",\"height\":\"1\",\"pass
thru\":true,\"multiple\":false,\"options\":[{\"label\":\"\",\"value\":\"\",\"type\":\"str\"}
],\"payload\":\"\",\"topic\":\"topic\",\"topicType\":\"msg\",\"className\":\"\",\"x\":800,\"y\"
:1380,\"wires\":[[\"099bff2f76e4b4d2\"]]],{"id":"578b3649d68658cd\",\"type\":\"func
tion\",\"z\":\"d0b1c77bddc9db6d\",\"name\":\"Formato\",\"func\":\"msg.options = [];\nfor
(var i=0; i<msg.payload.length; i++){\\n      obj = {};\n      obj
[msg.payload[i].name.replace(/^.*(\\\\\\|\\\\/|\\\\:)/, '')]
=
msg.payload[i].name\\n      msg.options.push(obj);\n}\\nmsg.payload={};\nreturn
msg;\",\"outputs\":1,\"noerr\":0,\"initialize\":\"\",\"finalize\":\"\",\"libs\":[],\"x\":640
,\"y\":1380,\"wires\":[[\"080a0b46c2657e7e\"]]],{"id":"099bff2f76e4b4d2\",\"type\":\"
function\",\"z\":\"d0b1c77bddc9db6d\",\"name\":\"Salvar\",\"func\":\"flow.set(\\\"Arquivo
SelecionadoVelocidadeVento\\\",msg.payload);\nreturn
msg;\",\"outputs\":1,\"noerr\":0,\"initialize\":\"\",\"finalize\":\"\",\"libs\":[],\"x\":950
,\"y\":1380,\"wires\":[]]],{"id\":\"cdbda271e7300e25\",\"type\":\"ui_button\",\"z\":\"d0
b1c77bddc9db6d\",\"name\":\"\",\"group\":\"36e31970e049e5e0\",\"order\":4,\"width\":\"6\",
\"height\":\"1\",\"passthru\":false,\"label\":\"Atualizar\",\"tooltip\":\"\",\"color\":\"\",
\"bgcolor\":\"\",\"className\":\"\",\"icon\":\"refresh\",\"payload\":\"\",\"payloadType\":\"str
\",\"topic\":\"topic\",\"topicType\":\"msg\",\"x\":80,\"y\":660,\"wires\":[[\"a97808fa96769
65e\",\"e5965772c61996c9\",\"d6be7b3b85bc58ea\"]]],{"id\":\"a97808fa9676965e\",\"typ

```

```

e":"fs-file-
lister","z":"d0b1c77bddc9db6d","name":"","start":"/home/iffontes/Medidas/Te
mperatura","pattern":"*.*","folders":"*","hidden":true,"lstype":"files","pa
th":true,"single":true,"depth":0,"stat":true,"showWarnings":true,"x":440,"y
":660,"wires":[["843f96a6ed6a74cb","695b719d00ef632e"]],{"id":"3f4cbd2f5dd
2e3ec","type":"ui_chart","z":"d0b1c77bddc9db6d","name":"","group":"36e31970
e049e5e0","order":3,"width":"22","height":"5","label":"Histórico da
Temperatura
[Celsius]","chartType":"line","legend":"false","xformat":"HH:mm:ss","interp
olate":"linear","nodata":"","dot":false,"ymin":"0","ymax":"50","removeOlder
":"24","removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneCo
lor":false,"useUTC":false,"colors":["#32a041","#aec7e8","#ff7f0e","#32a041"
,"#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDiffere
ntColor":false,"className":"","x":1460,"y":820,"wires":[[]]},{id":"d8df6fb
4be28ca5a","type":"ui_button","z":"d0b1c77bddc9db6d","name":"","group":"36e
31970e049e5e0","order":8,"width":"4","height":"1","passthru":false,"label":
"Visualizar
Gráfico","tooltip":"","color":"","bgcolor":"","className":"","icon":"show_c
hart","payload":"","payloadType":"str","topic":"topic","topicType":"msg","x
":470,"y":820,"wires":[["e992c844b3fa7924"]],{"id":"8736c42b8e31e07d","typ
e":"ui_template","z":"d0b1c77bddc9db6d","group":"36e31970e049e5e0","name":"
Template
Histórico","order":2,"width":"6","height":"5","format":"<style>\n.myDiv {\n
border-radius: 25px;\n border: 3px solid #2f9e41;\n padding: 20px; \n
width: auto;\n height: auto; \n}\n\n<na:link {\n color: #2f9e41;\n
background-color: transparent;\n text-decoration: none;\n}\n\n/* width
*/\n::-webkit-scrollbar {\n width: 10px;\n}\n\n/* Track */\n::-webkit-
scrollbar-track {\n background: #f1f1f1; \n}\n\n/* Handle */\n::-webkit-
scrollbar-thumb {\n background: #2f9e41; \n}\n\n/* Handle on hover */\n::-
webkit-scrollbar-thumb:hover {\n background: #2f9e41; \n}\n\n</style>\n\n<div
class=\`myDiv\`
ng-bind-
html=\`msg.payload\`><br/>\n</div>","storeOutMessages":true,"fwdInMessages"
:true,"resendOnRefresh":true,"templateScope":"local","className":"","x":105
0,"y":660,"wires":[[]]},{id":"2f2f33d322659e7a","type":"template","z":"d0b
1c77bddc9db6d","name":"Template
Tabela","field":"payload","fieldType":"msg","format":"handlebars","syntax":
"mustache","template":"<table
width=\`100%\`>\n
<tr><th>Temperatura_Dia_Mês_Ano</th></tr>\n {{#payload}}\n
<tr>\n
<td><a
href=\`/home/iffontes/Medidas/Temperatura/?filename={{name}}\`"
target=\`"blank"\`>{{fname}}</a></td>\n
</tr>\n
{{/payload}}\n</table>","output":"str","x":840,"y":660,"wires":[["8736c42b8

```



```

0b1c77bddc9db6d", "name": "", "rules": [{"t": "set", "p": "headers", "pt": "msg", "to": "{}", "tot": "json"}, {"t": "set", "p": "headers.content-type", "pt": "msg", "to": "txt/csv", "tot": "str"}, {"t": "set", "p": "headers.Content-Disposition", "pt": "msg", "to": "contentdisposition", "tot": "msg"}], "action": "", "property": "", "from": "", "to": "", "reg": false, "x": 1130, "y": 760, "wires": [{"3251d8543c2e7dd4"}], {"id": "3251d8543c2e7dd4", "type": "http response", "z": "d0b1c77bddc9db6d", "name": "", "statusCode": "", "headers": {}, "x": 1280, "y": 760, "wires": []}, {"id": "8c8db1e337d9b5b5", "type": "ui_dropdown", "z": "d0b1c77bddc9db6d", "name": "", "label": "Arquivo:", "tooltip": "", "place": "Seleção um arquivo de Temperatura", "group": "36e31970e049e5e0", "order": 5, "width": "18", "height": "1", "passthru": true, "multiple": false, "options": [{"label": "", "value": "", "type": "str"}], "payload": "", "topic": "topic", "topicType": "msg", "className": "", "x": 760, "y": 700, "wires": [{"07facfafde9612e0"}], {"id": "695b719d00ef632e", "type": "function", "z": "d0b1c77bddc9db6d", "name": "Formato", "func": "msg.options = []; \n for (var i=0; i<msg.payload.length; i++){ \n     obj = {}; \n     obj[msg.payload[i].name.replace(/\\.*(\\\\\\\\|\\\\\\\\/|\\\\\\\\:)/, '\\')] = msg.payload[i].name \n     msg.options.push(obj); \n } \n msg.payload={}; \n return msg;", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 600, "y": 700, "wires": [{"8c8db1e337d9b5b5"}], {"id": "07facfafde9612e0", "type": "function", "z": "d0b1c77bddc9db6d", "name": "Salvar", "func": "flow.set('ArquivoS elecionadoTemperatura\\', msg.payload); \n return msg;", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 910, "y": 700, "wires": []}, {"id": "a6e14644c709a592", "type": "ui_button", "z": "d0b1c77bddc9db6d", "name": "", "group": "344574d16e88f295", "order": 4, "width": "6", "height": "1", "passthru": false, "label": "Atualizar", "tooltip": "", "color": "", "bgcolor": "", "className": "", "icon": "refresh", "payload": "", "payloadType": "str", "topic": "topic", "topicType": "msg", "x": 80, "y": 980, "wires": [{"e5965772c61996c9", "a97808fa9676965e", "d6be7b3b85bc58ea"}], {"id": "e5965772c61996c9", "type": "fs-file-list", "z": "d0b1c77bddc9db6d", "name": "", "start": "/home/iffontes/Medidas/Radiacao", "pattern": "*.*", "folders": "*", "hidden": true, "lstype": "files", "path": true, "single": true, "depth": 0, "stat": true, "showWarnings": true, "x": 440, "y": 980, "wires": [{"1b7a37ccefd973b", "cb53d90c15a8f99e"}], {"id": "2400634282a0b423", "type": "ui_chart", "z": "d0b1c77bddc9db6d", "name": "", "group": "344574d16e88f295", "order": 3, "width": "22", "height": "5", "label": "Histórico da Radiação Solar - [W/m²]", "chartType": "line", "legend": "false", "xformat": "HH:mm:ss", "interpolate": "linear", "nodata": "", "dot": false, "ymin": "0", "ymax": "850", "removeOlder": "24", "removeOlderPoints": "", "removeOlderUnit": "3600", "cutout": 0, "useOneColor":

```

```

r":false,"useUTC":false,"colors":["#32a041","#aec7e8","#ff7f0e","#32a041","#
#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"outputs":1,"useDifferent
Color":false,"className":"","x":1710,"y":1160,"wires":[[]]},{id:"5cd25e57
284f746a","type":"ui_button","z":"d0b1c77bddc9db6d","name":"","group":"3445
74d16e88f295","order":8,"width":"4","height":"1","passthru":false,"label":"
Visualizar
Gráfico","tooltip":"","color":"","bgcolor":"","className":"","icon":"show_c
hart","payload":"","payloadType":"str","topic":"topic","topicType":"msg","x
":470,"y":1160,"wires":[["3bf39e3ecfddf384"]]}, {"id":"f8a6ae0fb4e9f9df","ty
pe":"ui_template","z":"d0b1c77bddc9db6d","group":"344574d16e88f295","name":
"Template
Histórico","order":2,"width":"6","height":"5","format":"<style>\n.myDiv {\n
border-radius: 25px;\n  border: 3px solid #2f9e41;\n  padding: 20px; \n
width: auto;\n  height: auto; \n}\n\na:link {\n  color: #2f9e41;\n
background-color: transparent;\n  text-decoration: none;\n}\n\n/* width
*/\n::-webkit-scrollbar {\n  width: 10px;\n}\n\n/* Track */\n::-webkit-
scrollbar-track {\n  background: #f1f1f1; \n}\n\n/* Handle */\n::-webkit-
scrollbar-thumb {\n  background: #2f9e41; \n}\n\n/* Handle on hover */\n::-
webkit-scrollbar-thumb:hover {\n  background: #2f9e41; \n}\n</style>\n\n<div
class=\"myDiv\"
ng-bind-
html=\"msg.payload\"><br/>\n</div>","storeOutMessages":true,"fwdInMessages"
:true,"resendOnRefresh":true,"templateScope":"local","className":"","x":109
0,"y":980,"wires":[[]]},{id:"8b711dc8ff1a47ee","type":"template","z":"d0b
1c77bddc9db6d","name":"Template
Tabela","field":"payload","fieldType":"msg","format":"handlebars","syntax":
"mustache","template":"<table
width=\"100%\">\n
<tr><th>RadiacaoSolar_Dia_Mês_Ano</th></tr>\n  {{#payload}}\n    <tr>\n
<td><a
href=\"/home/iffontes/Medidas/Radiacao/?filename={{name}}\"
target=\"blank\">{{fname}}</a></td>\n
</tr>\n
{{/payload}}\n</table>","output":"str","x":880,"y":980,"wires":[["f8a6ae0fb
4e9f9df"]]}, {"id":"1b7a37ccefd973b","type":"function","z":"d0b1c77bddc9db6
d","name":"ListaNomeArquivo","func":"for (var i=0; i<msg.payload.length;
i++) {\n
msg.payload[i].fname =
msg.payload[i].name.replace(/^(.*\\\\\\\\|\\\\|\\\\:)/,
');\n}\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":670
,"y":980,"wires":[["8b711dc8ff1a47ee"]]}, {"id":"3bf39e3ecfddf384","type":"f
unction","z":"d0b1c77bddc9db6d","name":"PegarNomeArquivo","func":"let
NomeArquivo = flow.get(\"ArquivoSelecioneadoRadiacaoSolar\");\nif(NomeArquivo
== undefined){\n  return;\n}\nmsg.filename = NomeArquivo;\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":890
,"y":1160,"wires":[["33a84ff25d05a5c2"]]}, {"id":"33a84ff25d05a5c2","type":"

```

file

```
in", "z": "d0b1c77bddc9db6d", "name": "", "filename": "", "format": "utf8", "chunk":
false, "sendError": false, "encoding": "none", "allProps": false, "x": 1070, "y": 116
0, "wires": [{"id": "ad348c9f48e516d5", "type": "csv", "z":
"d0b1c77bddc9db6d", "name": "", "sep": ",", "hdrin": true, "hdrout": "none", "multi
": "mult", "ret": "\\n", "temp": "", "skip": "0", "strings": true, "include_empty_str
ings": "", "include_null_values": "", "x": 1210, "y": 1160, "wires": [{"id": "2822fb9c3ddb
bcc5", "type": "function", "z": "d0b1c77bddc9db6d",
"name": "FormatarDadosParaGráfico", "func": "var Grafico = [{\n    \"series\":
[\"RadiacaoSolar\"],\n    \"data\": [],\n}];\n\nvar RadiacaoSolar =
[];\n\nfor(var i=0; i<msg.payload.length; i++){
RadiacaoSolar.push({\"x\":msg.payload[i].Tempo,
\"y\":msg.payload[i].RadiacaoSolar});\n\nGrafico[0].data.push(RadiacaoSo
lar);\n\nmsg.payload = Grafico;\n\nreturn
msg;\", \"outputs\":1, \"noerr\":0, \"initialize\": \"\", \"finalize\": \"\", \"libs\": [], \"x\":142
0, \"y\":1160, \"wires\": [{\"id\": \"996fb1af74ad107d\", \"type\":
\"http
in\", \"z\": \"d0b1c77bddc9db6d\", \"name\": \"\", \"url\": \"/home/iffontes/Medidas/Radiacao
/\", \"method\": \"get\", \"upload\": false, \"swaggerDoc\": \"\", \"x\": 530, \"y\": 1100, \"wires\": [
[\"5329e06ec4879dca\"]], {\"id\": \"5329e06ec4879dca\", \"type\": \"function\", \"z\": \"d0b1
c77bddc9db6d\", \"name\": \"NomeDoArquivo\", \"func\": \"msg.filename
= msg.req.query.filename;\nmsg.contentdisposition = \"attachment;
filename=\\\"\\\" + msg.req.query.filename.replace(/^(.*(\\\\\\\\|\\\\/|\\\\:))/, '') +
\\\"\\\"\\\";\nreturn
msg;\", \"outputs\":1, \"noerr\":0, \"initialize\": \"\", \"finalize\": \"\", \"libs\": [], \"x\": 880
, \"y\": 1100, \"wires\": [{\"id\": \"3aa0775331a98474\", \"type\": \"
file
in\", \"z\": \"d0b1c77bddc9db6d\", \"name\": \"\", \"filename\": \"\", \"format\": \"\", \"chunk\": fals
e, \"sendError\": false, \"encoding\": \"none\", \"allProps\": false, \"x\": 1070, \"y\": 1100, \"w
ires\": [{\"id\": \"3d34efb19eeb945f\", \"type\": \"change\", \"z\":
\"d0b1c77bddc9db6d\", \"name\": \"\", \"rules\": [{\"t\": \"set\", \"p\": \"headers\", \"pt\": \"msg\", \"
to\": \"{}\", \"tot\": \"json\"}, {\"t\": \"set\", \"p\": \"headers.content-
type\", \"pt\": \"msg\", \"to\": \"txt/csv\", \"tot\": \"str\"}, {\"t\": \"set\", \"p\": \"headers.Conten
t-
Disposition\", \"pt\": \"msg\", \"to\": \"contentdisposition\", \"tot\": \"msg\"}], \"action\": \"
\", \"property\": \"\", \"from\": \"\", \"to\": \"\", \"reg\": false, \"x\": 1240, \"y\": 1100, \"wires\": [{\"6
31a26ee148100f7\"}], {\"id\": \"631a26ee148100f7\", \"type\": \"http
response\", \"z\": \"d0b1c77bddc9db6d\", \"name\": \"\", \"statusCode\": \"\", \"headers\": {}, \"x\"
: 1410, \"y\": 1100, \"wires\": []}, {\"id\": \"53d3f8a0de5a1430\", \"type\": \"ui_dropdown\", \"z
\": \"d0b1c77bddc9db6d\", \"name\": \"\", \"label\": \"Arquivo\", \"tooltip\": \"\", \"place\": \"Sel
ecione um arquivo de Radiação
```

```

Solar", "group": "344574d16e88f295", "order": 5, "width": "18", "height": "1", "pass
thru": true, "multiple": false, "options": [{ "label": "", "value": "", "type": "str"
}], "payload": "", "topic": "topic", "topicType": "msg", "className": "", "x": 800, "y"
: 1040, "wires": [[ "88321bba4f2e356c" ]], { "id": "cb53d90c15a8f99e", "type": "func
tion", "z": "d0b1c77bddc9db6d", "name": "Formato", "func": "msg.options = []; \nfor
(var i=0; i<msg.payload.length; i++){\n      obj = {};\n      obj
[msg.payload[i].name.replace(/^.*(\\\\|\\/|\\:)/, '')] =
msg.payload[i].name\n      msg.options.push(obj);\n}\nmsg.payload={};\nreturn
msg;", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 640
, "y": 1040, "wires": [[ "53d3f8a0de5a1430" ]], { "id": "88321bba4f2e356c", "type": "
function", "z": "d0b1c77bddc9db6d", "name": "Salvar", "func": "flow.set(\"Arquivo
SelecionadoRadiacaoSolar\", msg.payload);\nreturn
msg;", "outputs": 1, "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 950
, "y": 1040, "wires": [[]], { "id": "8fc754aa935335ac", "type": "ui_template", "z": "
d0b1c77bddc9db6d", "group": "cfa359d4106e224f", "name": "Template Informações do
Projeto e IFG", "order": 5, "width": "23", "height": "5", "format": "<!DOCTYPE
html>\n\n<style>\n\nna:link {\n      color: #2f9e41;\n      background-color:
transparent;\n      text-decoration: none;\n}\n\n/* width */\n::-webkit-
scrollbar {\n  width: 10px;\n}\n\n/* Track */\n::-webkit-scrollbar-track {\n
background: #f1f1f1; \n}\n\n/* Handle */\n::-webkit-scrollbar-thumb {\n
background: #2f9e41; \n}\n\n/* Handle on hover */\n::-webkit-scrollbar-
thumb:hover {\n  background: #2f9e41; \n}\n\n</style>\n\n<html>\n  <body>\n
<h1>Trabalho de Conclusão de Curso</h1>\n      <p>Orientador: Prof. Dr.
Ghunter Paulo Viajante</p>\n      <p>Aluno: Gabriel Viríssimo Queiroz
Lopes</p>\n      <p></p>\n      <h3>Projeto IoT (Internet of Things -
Internet das Coisas) - Laboratório de Fontes Renováveis</h3>\n      <p>Sensor
de Temperatura.</p>\n      <p>Sensor de Velocidade do Vento.</p>\n
<p>Sensor de Irradiância Solar.</p>\n
</body>\n</html>", "storeOutMessages": true, "fwdInMessages": true, "resendOnRef
resh": true, "templateScope": "local", "className": "", "x": 320, "y": 520, "wires": [
[]], { "id": "d2b220631ad80460", "type": "mqtt
in", "z": "d0b1c77bddc9db6d", "name": "", "topic": "Temperatura-IFG-
Itumbiara", "qos": "2", "datatype": "auto", "broker": "c3c5e6fb9490c966", "nl": fal
se, "rap": true, "rh": 0, "inputs": 0, "x": 490, "y": 600, "wires": [[ "936ae2de2c304d30
" ]], { "id": "936ae2de2c304d30", "type": "function", "z": "d0b1c77bddc9db6d", "nam
e": "ConfigurandoDados", "func": "var agora = new Date();\n\nvar ano =
agora.getFullYear();\nvar mes = agora.getMonth() < 9 ? \"0\" +
(agora.getMonth() + 1) : (agora.getMonth() + 1); // getMonth() inicia com
zero como padrão\nvar dia = agora.getDate() < 10 ? \"0\" + agora.getDate() :
agora.getDate(); \nvar horas = agora.getHours() < 10 ? \"0\" +
agora.getHours() : agora.getHours(); \nvar minutos = agora.getMinutes() < 10

```

```

? \"0\" + agora.getMinutes() : agora.getMinutes();\nvar segundos =
agora.getSeconds() < 10 ? \"0\" + agora.getSeconds() :
agora.getSeconds();\n\nmsg.payload = {\n  \"Temperatura\": msg.payload,\n  \"Dia\": dia,\n  \"Mes\": mes,\n  \"Ano\": ano,\n  \"Horas\": horas,\n  \"Minutos\": minutos,\n  \"Segundos\": segundos,\n  \"Tempo\":
agora.getTime()\n}\n\nreturn
msg;\",\"outputs\":1,\"noerr\":0,\"initialize\":\"\",\"finalize\":\"\",\"libs\":[],\"x\":740
,\"y\":600,\"wires\":[[\"861e75b98a552770\"]]],{\"id\":\"861e75b98a552770\",\"type\":\"f
unction\",\"z\":\"d0b1c77bddc9db6d\",\"name\":\"GeradorNomeDeArquivo\",\"func\":\"var
agora = new Date();\nvar ano = agora.getFullYear();\nvar mes =
agora.getMonth() < 9 ? \"0\" + (agora.getMonth() + 1) : (agora.getMonth() +
1); // getMonth() inicia com zero como padrão\nvar dia = agora.getDate() <
10 ? \"0\" + agora.getDate() : agora.getDate(); \nvar horas = agora.getHours()
< 10 ? \"0\" + agora.getHours() : agora.getHours(); \nvar minutos =
agora.getMinutes() < 10 ? \"0\" + agora.getMinutes() :
agora.getMinutes();\nvar segundos = agora.getSeconds() < 10 ? \"0\" +
agora.getSeconds() : agora.getSeconds();\n\nmsg.fname = \"Temperatura_\"+
dia + \"_\" + mes + \"_\" + ano + \".csv\";\n\nmsg.filename =
\"/home/iffontes/Medidas/Temperatura/\" + msg.fname;\n\nmsg.filecontent =
msg.payload;\n\nmsg.payload
=
{\"pattern\":msg.fname};\n\nnode.status({fill:\"blue\",shape:\"ring\",text:
msg.fname});\n\nreturn
msg;\",\"outputs\":1,\"noerr\":0,\"initialize\":\"\",\"finalize\":\"\",\"libs\":[],\"x\":970
,\"y\":600,\"wires\":[[\"45362ba530a15bdb\"]]],{\"id\":\"45362ba530a15bdb\",\"type\":\"f
s-file-
lister\",\"z\":\"d0b1c77bddc9db6d\",\"name\":\"\",\"start\":\"/home/iffontes/Medidas/Te
mperatura/\",\"pattern\":\"*.*\",\"folders\":\"*\",\"hidden\":true,\"lstype\":\"files\",\"p
ath\":true,\"single\":true,\"depth\":0,\"stat\":true,\"showWarnings\":false,\"x\":1160
,\"y\":600,\"wires\":[[\"af67bb6f7ec2dd43\"]]],{\"id\":\"af67bb6f7ec2dd43\",\"type\":\"s
witch\",\"z\":\"d0b1c77bddc9db6d\",\"name\":\"Arquivo já existe
?\",\"property\":\"$count(msg.payload)\",\"propertyType\":\"jsonata\",\"rules\":[{\"t\":
\"eq\",\"v\":\"0\",\"vt\":\"num\"},{\"t\":\"else\"}],\"checkall\":\"true\",\"repair\":false,\"ou
tputs\":2,\"x\":1330,\"y\":600,\"wires\":[[\"9af139acee908f54\"],[\"99ae7a37a3996ca2
\"]]],{\"id\":\"9af139acee908f54\",\"type\":\"change\",\"z\":\"d0b1c77bddc9db6d\",\"name\":
\"\",\"rules\":[{\"t\":\"set\",\"p\":\"payload\",\"pt\":\"msg\",\"to\":\"filecontent\",\"tot\":\"m
sg\"}],\"action\":\"\",\"property\":\"\",\"from\":\"\",\"to\":\"\",\"reg\":false,\"x\":1520,\"y\":
580,\"wires\":[[\"c82b6717057c9e8c\"]]],{\"id\":\"99ae7a37a3996ca2\",\"type\":\"change
\",\"z\":\"d0b1c77bddc9db6d\",\"name\":\"\",\"rules\":[{\"t\":\"set\",\"p\":\"payload\",\"pt\":\"
msg\",\"to\":\"filecontent\",\"tot\":\"msg\"}],\"action\":\"\",\"property\":\"\",\"from\":\"\",
to\":\"\",\"reg\":false,\"x\":1520,\"y\":620,\"wires\":[[\"bd333e66c8a92f08\"]]],{\"id\":\"
bd333e66c8a92f08\",\"type\":\"csv\",\"z\":\"d0b1c77bddc9db6d\",\"name\":\"\",\"sep\":\"\",

```

```

hdrin":false,"hdrout":"none","multi":"one","ret":"\\n","temp":"Temperatura,
Dia,Mes,Ano,Horas,Minutos,Segundos,Tempo","skip":"0","strings":true,"includ
e_empty_strings":"","include_null_values":"","x":1670,"y":620,"wires":[["99
544d77e1096e16"]]},{"id":"c82b6717057c9e8c","type":"csv","z":"d0b1c77bddc9d
b6d","name":"","sep":",","hdrin":false,"hdrout":"all","multi":"one","ret":"
\\n","temp":"Temperatura,Dia,Mes,Ano,Horas,Minutos,Segundos,Tempo","skip":"
0","strings":true,"include_empty_strings":"","include_null_values":"","x":1
670,"y":580,"wires":[["99544d77e1096e16"]]},{"id":"99544d77e1096e16","type"
:"file","z":"d0b1c77bddc9db6d","name":"","filename":"","appendNewline":fals
e,"createDir":true,"overwriteFile":"false","encoding":"none","x":1790,"y":6
00,"wires":[[]]},{"id":"c2071c2a0e87956e","type":"mqtt
in","z":"d0b1c77bddc9db6d","name":"","topic":"Vento-IFG-
Itumbiara","qos":"2","datatype":"auto","broker":"c3c5e6fb9490c966","nl":fal
se,"rap":true,"rh":0,"inputs":0,"x":470,"y":1260,"wires":[["7330076d8523197
6"]]},{"id":"7330076d85231976","type":"function","z":"d0b1c77bddc9db6d","na
me":"ConfigurandoDados","func":"var agora = new Date();\\n\\nvar ano =
agora.getFullYear();\\nvar mes = agora.getMonth() < 9 ? \"0\" +
(agora.getMonth() + 1) : (agora.getMonth() + 1); // getMonth() inicia com
zero como padrão\\nvar dia = agora.getDate() < 10 ? \"0\" + agora.getDate() :
agora.getDate(); \\nvar horas = agora.getHours() < 10 ? \"0\" +
agora.getHours() : agora.getHours(); \\nvar minutos = agora.getMinutes() < 10
? \"0\" + agora.getMinutes() : agora.getMinutes();\\nvar segundos =
agora.getSeconds() < 10 ? \"0\" + agora.getSeconds() :
agora.getSeconds();\\n\\nmsg.payload = {\\n          \\\"VelocidadeVento\\\":
msg.payload,\\n          \\\"Dia\\\": dia,\\n          \\\"Mes\\\": mes,\\n          \\\"Ano\\\": ano,\\n
          \\\"Horas\\\": horas,\\n          \\\"Minutos\\\": minutos,\\n          \\\"Segundos\\\": segundos,\\n
          \\\"Tempo\\\": agora.getTime()\\n}\\n\\nreturn
msg;","outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":700
,"y":1260,"wires":[["2625b1b1660228e6"]]},{"id":"2625b1b1660228e6","type":"
function","z":"d0b1c77bddc9db6d","name":"GeradorNomeDeArquivo","func":"var
agora = new Date();\\nvar ano = agora.getFullYear();\\nvar mes =
agora.getMonth() < 9 ? \"0\" + (agora.getMonth() + 1) : (agora.getMonth() +
1); // getMonth() inicia com zero como padrão\\nvar dia = agora.getDate() <
10 ? \"0\" + agora.getDate() : agora.getDate(); \\nvar horas = agora.getHours()
< 10 ? \"0\" + agora.getHours() : agora.getHours(); \\nvar minutos =
agora.getMinutes() < 10 ? \"0\" + agora.getMinutes() :
agora.getMinutes();\\nvar segundos = agora.getSeconds() < 10 ? \"0\" +
agora.getSeconds() : agora.getSeconds();\\n\\nmsg.fname =
\\\"VelocidadeVento_\\\"+ dia + \\\"_\\\" + mes + \\\"_\\\" + ano +
\\\".csv\\\";\\n\\nmsg.filename = \\\"/home/iffontes/Medidas/Vento/\\\" +
msg.fname;\\n\\nmsg.filecontent = msg.payload;\\n\\nmsg.payload =

```

```

{"pattern":msg.fname};\n\nnode.status({fill:"blue",shape:"ring",text:
msg.fname});\n\nreturn
msg;,"outputs":1,"noerr":0,"initialize":"","finalize":"","libs":[],"x":930
,"y":1260,"wires":[["c1ca9e599d5003f9"]]},{"id":"c1ca9e599d5003f9","type":
fs-file-
lister","z":"d0b1c77bdbc9db6d","name":"","start":"/home/iffontes/Medidas/Ve
nto/","pattern":"*.*","folders":"*","hidden":true,"lstype":"files","path":t
rue,"single":true,"depth":0,"stat":true,"showWarnings":false,"x":1120,"y":1
260,"wires":[["9abebe3b03975489"]]},{"id":"9abebe3b03975489","type":"switch
","z":"d0b1c77bdbc9db6d","name":"Arquivo já existe
?","property":"$count(msg.payload)","propertyType":"jsonata","rules":[{"t":
"eq","v":"0","vt":"num"},{"t":"else"}],"checkall":"true","repair":false,"ou
tputs":2,"x":1290,"y":1260,"wires":[["3a7be83163739ac8"],["9aeale622fc7728a
"]]},{"id":"3a7be83163739ac8","type":"change","z":"d0b1c77bdbc9db6d","name"
:"","rules":[{"t":"set","p":"payload","pt":"msg","to":"filecontent","tot":
"msg"}],"action":"","property":"","from":"","to":"","reg":false,"x":1480,"y"
:1240,"wires":[["20f2c287656dd33f"]]},{"id":"9aeale622fc7728a","type":"chan
ge","z":"d0b1c77bdbc9db6d","name":"","rules":[{"t":"set","p":"payload","pt"
:"msg","to":"filecontent","tot":"msg"}],"action":"","property":"","from":"","
to":"","reg":false,"x":1480,"y":1280,"wires":[["c6a0e4ea001f42d1"]]},{"id
":"c6a0e4ea001f42d1","type":"csv","z":"d0b1c77bdbc9db6d","name":"","sep":
",","hdrin":false,"hdrout":"none","multi":"one","ret":"\n","temp":"Velocidad
eVento,Dia,Mes,Ano,Horas,Minutos,Segundos,Tempo","skip":"0","strings":true,
"include_empty_strings":"","include_null_values":"","x":1630,"y":1280,"wire
s":[["56e9eb4a314994ca"]]},{"id":"20f2c287656dd33f","type":"csv","z":"d0b1c
77bdbc9db6d","name":"","sep":",","hdrin":false,"hdrout":"all","multi":"one"
,"ret":"\n","temp":"VelocidadeVento,Dia,Mes,Ano,Horas,Minutos,Segundos,Tem
po","skip":"0","strings":true,"include_empty_strings":"","include_null_valu
es":"","x":1630,"y":1240,"wires":[["56e9eb4a314994ca"]]},{"id":"56e9eb4a314
994ca","type":"file","z":"d0b1c77bdbc9db6d","name":"","filename":"","append
Newline":false,"createDir":true,"overwriteFile":"false","encoding":"none","
x":1750,"y":1260,"wires":[[]]},{"id":"cb07a13c0da4468a","type":"mqtt
in","z":"d0b1c77bdbc9db6d","name":"","topic":"Radiacao-IFG-
Itumbiara","qos":"2","datatype":"auto","broker":"c3c5e6fb9490c966","nl":fal
se,"rap":true,"rh":0,"inputs":0,"x":480,"y":920,"wires":[["e7de35b90ba69866
"]]},{"id":"e7de35b90ba69866","type":"function","z":"d0b1c77bdbc9db6d","nam
e":"ConfigurandoDados","func":"var agora = new Date();\n\nvar ano =
agora.getFullYear();\nvar mes = agora.getMonth() < 9 ? \"0\" +
(agora.getMonth() + 1) : (agora.getMonth() + 1); // getMonth() inicia com
zero como padrão\nvar dia = agora.getDate() < 10 ? \"0\" + agora.getDate() :
agora.getDate(); \nvar horas = agora.getHours() < 10 ? \"0\" +

```

```

    agora.getHours() : agora.getHours(); \nvar minutos = agora.getMinutes() < 10
    ? \"0\" + agora.getMinutes() : agora.getMinutes();\nvar segundos =
    agora.getSeconds() < 10 ? \"0\" + agora.getSeconds() :
    agora.getSeconds();\n\nmsg.payload = {\n  \"RadiacaoSolar\": msg.payload,\n
    \"Dia\": dia,\n  \"Mes\": mes,\n  \"Ano\": ano,\n  \"Horas\": horas,\n
    \"Minutos\": minutos,\n  \"Segundos\": segundos,\n  \"Tempo\":
    agora.getTime()\n}\n\nreturn
msg;\", \"outputs\":1, \"noerr\":0, \"initialize\": \"\", \"finalize\": \"\", \"libs\":[], \"x\":720
, \"y\":920, \"wires\":[[\"cab3ddcflae6864c\"]], {\"id\":\"cab3ddcflae6864c\", \"type\":\"f
unction\", \"z\":\"d0b1c77bddc9db6d\", \"name\":\"GeradorNomeDeArquivo\", \"func\":\"var
    agora = new Date();\nvar ano = agora.getFullYear();\nvar mes =
    agora.getMonth() < 9 ? \"0\" + (agora.getMonth() + 1) : (agora.getMonth() +
    1); // getMonth() inicia com zero como padrão\nvar dia = agora.getDate() <
    10 ? \"0\" + agora.getDate() : agora.getDate(); \nvar horas = agora.getHours()
    < 10 ? \"0\" + agora.getHours() : agora.getHours(); \nvar minutos =
    agora.getMinutes() < 10 ? \"0\" + agora.getMinutes() :
    agora.getMinutes();\nvar segundos = agora.getSeconds() < 10 ? \"0\" +
    agora.getSeconds() : agora.getSeconds();\n\nmsg.fname = \"RadiacaoSolar_\"+
    dia + \"_\" + mes + \"_\" + ano + \".csv\";\n\nmsg.filename =
    \"/home/iffontes/Medidas/Radiacao/\" + msg.fname;\n\nmsg.filecontent =
    msg.payload;\n\nmsg.payload
    = {\n\"pattern\":msg.fname};\n\nnode.status({fill:\"blue\",shape:\"ring\",text:
    msg.fname});\n\nreturn
msg;\", \"outputs\":1, \"noerr\":0, \"initialize\": \"\", \"finalize\": \"\", \"libs\":[], \"x\":950
, \"y\":920, \"wires\":[[\"e7e32a93c35b16e9\"]], {\"id\":\"e7e32a93c35b16e9\", \"type\":\"f
s-file-
    lister\", \"z\":\"d0b1c77bddc9db6d\", \"name\":\"\", \"start\":\"/home/iffontes/Medidas/Ra
    diacao/\", \"pattern\":\"*.*\", \"folders\":\"*\", \"hidden\":true, \"lstype\":\"files\", \"path
    \":true, \"single\":true, \"depth\":0, \"stat\":true, \"showWarnings\":false, \"x\":1140, \"y
    \":920, \"wires\":[[\"c21a19e6c125403b\"]], {\"id\":\"c21a19e6c125403b\", \"type\":\"swit
    ch\", \"z\":\"d0b1c77bddc9db6d\", \"name\":\"Arquivo já existe
    ?\", \"property\":\"$count(msg.payload)\", \"propertyType\":\"jsonata\", \"rules\":[{\"t\":
    \"eq\", \"v\":\"0\", \"vt\":\"num\"}, {\"t\":\"else\"}], \"checkall\":\"true\", \"repair\":false, \"ou
    tputs\":2, \"x\":1310, \"y\":920, \"wires\":[[\"d038742427f890ca\"], [\"a38d811b7e09ad0a\"
    ]], {\"id\":\"d038742427f890ca\", \"type\":\"change\", \"z\":\"d0b1c77bddc9db6d\", \"name\":
    \"\", \"rules\":[{\"t\":\"set\", \"p\":\"payload\", \"pt\":\"msg\", \"to\":\"filecontent\", \"tot\":\"m
    sg\"}], \"action\":\"\", \"property\":\"\", \"from\":\"\", \"to\":\"\", \"reg\":false, \"x\":1500, \"y\":
    900, \"wires\":[[\"f744a341480f5843\"]], {\"id\":\"a38d811b7e09ad0a\", \"type\":\"change
    \", \"z\":\"d0b1c77bddc9db6d\", \"name\":\"\", \"rules\":[{\"t\":\"set\", \"p\":\"payload\", \"pt\":
    \"msg\", \"to\":\"filecontent\", \"tot\":\"msg\"}], \"action\":\"\", \"property\":\"\", \"from\":\"\",
    \"to\":\"\", \"reg\":false, \"x\":1500, \"y\":940, \"wires\":[[\"3d69340653183214\"]], {\"id\":\"

```

```

3d69340653183214", "type": "csv", "z": "d0b1c77bddc9db6d", "name": "", "sep": ",", "
hdrin": false, "hdrout": "none", "multi": "one", "ret": "\\n", "temp": "RadiacaoSola
r, Dia, Mes, Ano, Horas, Minutos, Segundos, Tempo", "skip": "0", "strings": true, "incl
ude_empty_strings": "", "include_null_values": "", "x": 1650, "y": 940, "wires": [{"
d06ad6a71ab5cbae"}]}, {"id": "f744a341480f5843", "type": "csv", "z": "d0b1c77bddc
9db6d", "name": "", "sep": ",", "hdrin": false, "hdrout": "all", "multi": "one", "ret"
: "\\n", "temp": "RadiacaoSolar, Dia, Mes, Ano, Horas, Minutos, Segundos, Tempo", "ski
p": "0", "strings": true, "include_empty_strings": "", "include_null_values": "", "
x": 1650, "y": 900, "wires": [{"d06ad6a71ab5cbae"}]}, {"id": "d06ad6a71ab5cbae", "t
ype": "file", "z": "d0b1c77bddc9db6d", "name": "", "filename": "", "appendNewline":
false, "createDir": true, "overwriteFile": "false", "encoding": "none", "x": 1770, "
y": 920, "wires": []}, {"id": "4db332034cb88d5e", "type": "comment", "z": "d0b1c77
bddc9db6d", "name": "Interface -> Laboratório -
Hoje", "info": "", "x": 150, "y": 40, "wires": []}, {"id": "eb9a7e0fad67b40d", "type":
"ui_gauge", "z": "d0b1c77bddc9db6d", "name": "", "group": "e9cdc14bb3de0c65", "ord
er": 1, "width": "8", "height": "6", "gtype": "gage", "title": "Temperatura -
[Celsius]", "label": "°C", "format": "{{value}}", "min": "0", "max": "50", "colors":
["#32a041", "#e6e600", "#ca3838"], "seg1": "40", "seg2": "45", "className": "", "x":
390, "y": 160, "wires": []}, {"id": "c3c5e6fb9490c966", "type": "mqtt-
broker", "name": "", "broker": "192.168.1.186", "port": "1883", "clientId": "NODE-
RED-
ASSINANTE", "autoConnect": true, "usetls": false, "protocolVersion": "4", "keepali
ve": "60", "cleansession": true, "birthTopic": "", "birthQos": "0", "birthPayload":
"", "birthMsg": {}, "closeTopic": "", "closeQos": "0", "closePayload": "", "closeMsg
": {}, "willTopic": "", "willQos": "0", "willPayload": "", "willMsg": {}, "sessionExp
iry": ""}, {"id": "e9cdc14bb3de0c65", "type": "ui_group", "name": "Temperatura", "t
ab": "f5e1dc40e6044532", "order": 2, "disp": true, "width": "28", "collapse": true, "
className": ""}, {"id": "27fd740fbc6ce760", "type": "ui_group", "name": "Velocidade
do
Vento", "tab": "f5e1dc40e6044532", "order": 4, "disp": true, "width": "28", "collaps
e": true, "className": ""}, {"id": "fbf8c256b7b80382", "type": "ui_group", "name": "
Laboratório de Fontes
Renováveis", "tab": "f5e1dc40e6044532", "order": 1, "disp": true, "width": "28", "co
llapse": false, "className": ""}, {"id": "3d6c2c5a88b96f7c", "type": "ui_group", "n
ame": "Radiação
Solar", "tab": "f5e1dc40e6044532", "order": 3, "disp": true, "width": "28", "collaps
e": true, "className": ""}, {"id": "cfa359d4106e224f", "type": "ui_group", "name": "
Laboratório de Fontes
Renováveis", "tab": "f36b7f965744c45d", "order": 1, "disp": true, "width": "28", "co
llapse": false, "className": ""}, {"id": "a34f35553fe7bed7", "type": "ui_group", "n
ame": "Histórica da Velocidade do

```

```
Vento", "tab": "f36b7f965744c45d", "order": 4, "disp": true, "width": "28", "collapse": false, "className": ""}, {"id": "36e31970e049e5e0", "type": "ui_group", "name": "Histórico da Temperatura", "tab": "f36b7f965744c45d", "order": 2, "disp": true, "width": "28", "collapse": false, "className": ""}, {"id": "344574d16e88f295", "type": "ui_group", "name": "Histórico da Radiação Solar", "tab": "f36b7f965744c45d", "order": 3, "disp": true, "width": "28", "collapse": false, "className": ""}, {"id": "f5e1dc40e6044532", "type": "ui_tab", "name": "Laboratório - Hoje", "icon": "today", "order": 1, "disabled": false, "hidden": false}, {"id": "f36b7f965744c45d", "type": "ui_tab", "name": "Laboratório - Histórico", "icon": "history", "order": 2, "disabled": false, "hidden": false}]
```

APÊNDICE 16 – FLUXO LABORATÓRIO - HOJE



APÊNDICE 17 – FLUXO LABORATÓRIO - HISTÓRICO

