Relatório de Projeto

Fábio Emanuel Fiqueli Abreu

Engenharia Informática

jul | 2023

GUARDA
POLI
TÉCNICO

POLI TÉCNICO GUARDA

Escola Superior de Tecnologia e Gestão

REDE WIRELESS PARA LABORATÓRIO DE ENERGIAS RENOVÁVEIS DA ESTG

PROJETO EM CONTEXTO DE ESTÁGIO
PARA OBTENÇÃO DO GRAU DE LICENCIADO EM ENGENHARIA INFORMÁTICA

FÁBIO EMANUEL FIQUELI ABREU
Julho / 2023

Escola Superior de Tecnologia e Gestão

REDE WIRELESS PARA LABORATÓRIO DE ENERGIAS RENOVÁVEIS DA ESTG

PROJETO EM CONTEXTO DE ESTÁGIO
PARA OBTENÇÃO DO GRAU DE LICENCIADO EM ENGENHARIA INFORMÁTICA

Professor Orientador: Filipe José Neto Caetano

Professor Supervisor: Adérito Neto Alcaso

FÁBIO EMANUEL FIQUELI ABREU

Julho / 2023

Agradecimentos

Gostaria de expressar a minha profunda gratidão à minha família, pois sem o vosso apoio incondicional, a minha progressão académica não teria sido possível. Sou imensamente grato por ter sempre o vosso suporte e encorajamento, que têm impulsionado os meus sonhos e objetivos.

Aos meus amigos, tenho uma gratidão especial pela amizade genuína que partilhamos, pelo companheirismo e pelos momentos inesquecíveis que vivemos juntos. A vossa alegria e encorajamento constante tornaram a minha experiência académica ainda mais rica e memorável.

Não posso deixar de expressar um agradecimento especial aos docentes que me acompanharam neste estágio, cujo conhecimento transmitido, tempo dedicado e orientação foram fundamentais para o meu crescimento académico.

Ficha de Identificação

Aluno

Nome: Fábio Emanuel Fiqueli Abreu

Número: 1704154

Licenciatura: Engenharia Informática

Estabelecimento de Ensino

Instituto Politécnico da Guarda (IPG)

Escola Superior de Tecnologia e Gestão (ESTG)

Entidade Acolhedora do Projeto em Contexto de Estágio

Nome: Instituto Politécnico da Guarda (IPG)

Morada: Av. Dr. Francisco Sá Carneiro 50, 6300-559 Guarda

Contacto Telefónico: 271 220 100

Duração do Projeto em Contexto de Estágio

Inicio: 20 de Março de 2023

Fim: 16 Junho de 2023

Supervisor de Estágio

Nome: Adérito Neto Alcaso

Função: Coordenador do CISE no IPG

Docente Orientador de Estágio

Nome: Filipe José Neto Caetano

Grau Académico: Mestre

Resumo

Este relatório descreve um projeto desenvolvido no âmbito de um estágio do curso de Licenciatura em Engenharia Informática na Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

O projeto denominado "*REWLER*" (Rede Wireless Para Laboratório De Energias Renováveis Da ESTG), consiste em um sistema de monitorização que tem como objetivo recolher dados provenientes de duas estações: uma estação meteorológica e uma estação de painéis solares.

A implementação deste sistema envolveu o uso de microcontroladores da família ATMEL, bem como módulos Xbees, a fim de estabelecer uma troca de informações segura e confiável.

Após a receção dos dados por parte do coordenador da rede, estes são guardados num cartão SD como "*backup*". Adicionalmente, o coordenador, ligado a uma rede Wi-Fi, procede a publicação dos dados utilizando o protocolo MQTT para a plataforma Node-RED. No Node-RED, os dados enviados são armazenados numa base de dados MySQL, sendo posteriormente criadas páginas onde o utilizador pode selecionar uma determinada data ou intervalo de datas para visualizar os dados extraídos, apresentados sob a forma de tabelas e gráficos. Com o intuito de facilitar a partilha dos dados é possível fazer o *download* dos dados de cada uma das estações a partir da seleção de uma determinada data.

O objetivo principal deste projeto é desenvolver um sistema de monitorização de baixo custo, de fácil utilização e manutenção, que permita visualizar e comparar os dados recolhidos, de forma a controlar e otimizar o funcionamento das duas estações monitorizadas.

Palavras-Chave: Arduino, MQTT, MySQL, Node-Red, PVT, Sistema de Monotorização, ZigBee.

Abstract

This report describes a project developed within the scope of an internship for the Bachelor's degree in Computer Engineering at the School of Technology and Management of the Polytechnic Institute of Guarda.

The project called "REWLER" (Wireless Network for Renewable Energy Laboratory of ESTG) consists of a monitoring system that aims to collect data from two stations: a weather station and a solar panel station.

The implementation of this system involved the use of microcontrollers from the ATMEL family, as well as Xbee modules, to establish a secure and reliable exchange of information.

After the reception of data by the network coordinator, they are stored on an SD card as a backup. Additionally, the coordinator, connected to a Wi-Fi network, publishes the data using the MQTT protocol to the Node-RED platform. In Node-RED, the sent data is stored in a MySQL database, and subsequently, pages are created where the user can select a specific date or date range to view the extracted data, presented in the form of tables and graphs. To facilitate data sharing, it is possible to download the data from each station based on the selection of a specific date.

The main objective of this project is to develop a low-cost monitoring system that is easy to use and maintain, allowing visualization and comparison of the collected data to control and optimize the operation of the two monitored stations.

Keywords: Arduino, MQTT, MySQL, Node-Red, PVT, Monitoring System, ZigBee.

Índice

Índice de figuras

Índice de Tabelas

Lista de Siglas e Acrónimos

C	Linguagem de programação amplamente utilizada, conhecida pela sua eficiência e flexibilidade
EM	Estação Meteorológica
ES	Estação de Painéis Solares
ESTG	Escola Superior de Tecnologia e Gestão
HTTP	*Hypertext Transfer Protocol*, protocolo de comunicação utilizado para transferir dados na *World Wide Web*
IDE	*Integrated Development Environment*, uma ferramenta que oferece recursos para desenvolver e testar programas
IoT	*Internet of Things*, interconexão de dispositivos físicos ligados à Internet, permitindo comunicação, compartilhamento de dados e automação.
IP	*Internet Protocol*, protocolo responsável pela identificação e localização dos dispositivos conectados numa rede
IPG	Instituto Politécnico da Guarda
JavaScript	Linguagem de programação popular para desenvolvimento web e interações dinâmicas em páginas da WEB
MQTT	*Message Queuing Telemetry Transport*, protocolo de comunicação leve e eficiente para troca de mensagens entre dispositivos conectados em redes de comunicação
Perl	*Practical Extraction and Reporting Language*, é uma linguagem de programação versátil e poderosa, ideal para manipulação de texto e automação de tarefas.
PHP	*Hypertext Preprocessor*, linguagem de programação amplamente utilizada para o desenvolvimento de aplicações web
PVT	Painel termofotovoltaico
REWLER	Rede Wireless para Laboratório de Energias Renováveis da ESTG

SD Card	*Secure Digital Card*, um tipo de dispositivo de armazenamento removível usado em dispositivos eletrônicos
SQL	*Structured Query Language*, uma linguagem de programação usada para gerir bases de dados
URL	*Uniform Resource Locator*, endereço único que identifica a localização de um recurso na web
Wi-Fi	*Wireless Fidelity*, uma tecnologia de comunicação sem fio que permite conexão à internet e troca de dados entre dispositivos
Xbee	Módulo de comunicação sem fios de baixo consumo, utilizado para criar redes de dispositivos interconectados
ZigBee	Protocolo de comunicação sem fios de curto alcance, geralmente utilizado para aplicações de automação residencial e industrial

1. Introdução

O presente relatório descreve de forma detalhada um projeto desenvolvido em contexto de estágio realizado no Laboratório de Energias Renováveis do Instituto Politécnico da Guarda (IPG). Este estágio foi realizado pelo aluno Fábio Emanuel Fiqueli Abreu no contexto da disciplina de projeto de informática, pertencente ao 3º ano do curso de licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão (ESTG) do IPG. Neste relatório, serão apresentados de forma sistemática e rigorosa os elementos essenciais do projeto, incluindo as metodologias adotadas e os resultados obtidos.

1.1. Enquadramento e Motivação

O presente projeto surge como resposta à necessidade de monitorizar uma estação meteorológica e uma estação de painéis solares. Inicialmente, já havia sido desenvolvido um projeto denominado SIMONIC no âmbito de um mestrado, que recolhia os dados provenientes da estação meteorológica, incluindo os dados de um painel termofotovoltaico (PVT). No entanto, devido a alterações implementadas na segurança da rede do IPG, o sistema tornou-se inoperacional.

Com o intuito de proporcionar uma solução mais atualizada e fiável, este projeto apresenta várias diferenças em relação ao projeto mencionado anteriormente. Nesta nova abordagem, foram exploradas estratégias distintas, recolhidos um maior volume de dados e utilizados novos sensores na estação meteorológica. Além disso, foram extraídos dados de 10 PVT. Para o desenvolvimento de um website com Internet das Coisas (IoT), foi adotado o Node-RED, uma plataforma de código aberto atual. Propõe-se ainda a utilização de uma base de dados MySQL como sistema de gestão de base de dados.

A motivação subjacente a este projeto está relacionada com o crescente interesse no campo das energias renováveis, no combate às alterações climáticas e na transição para o digital, mas com uma perspetiva energética mais sustentável.

1.2. Objetivos

O objetivo principal deste projeto é desenvolver um sistema de monitorização de baixo custo, fácil utilização e manutenção, que permita visualizar e comparar os dados recolhidos, de forma a controlar e otimizar a eficiência das duas estações referidas no resumo. As fases para o desenvolvimento do sistema em questão são as seguintes:

- Troca de Informações entre microcontroladores através do uso de módulos Xbee;
- Realização da recolha dos dados provenientes dos sensores;

- Armazenamento dos dados recolhidos num cartão SD;
- Envio dos dados para o Node-RED através do protocolo MQTT;
- Armazenamento dos dados numa base de dados MySQL;
- Criação de tabelas e gráficos para visualizar os dados recolhidos através da escolha de uma data ou intervalo de datas;
- Transferência dos dados recolhidos através da seleção de uma data específica.

Através destas etapas, pretende-se alcançar o objetivo de criar um sistema eficaz que permita monitorizar e otimizar ambas as estações.

1.3.Estrutura do relatório

Este relatório encontra-se estruturado em seis capítulos:

No primeiro capítulo, procede-se à realização de uma introdução ao projeto, fornecendo o enquadramento em que este se insere, juntamente com o propósito e as metas a serem alcançadas.

No segundo capítulo, é feita uma análise abrangente dos sistemas de monitorização existentes, abordando a sua variedade, importância e fornecendo alguns exemplos de sistemas disponíveis no mercado.

O terceiro capítulo descreve detalhadamente a arquitetura utilizada no desenvolvimento do sistema. São justificadas as escolhas efetuadas, tendo em conta o ambiente e as características em que o sistema será aplicado, bem como os resultados desejados.

No quarto capítulo, são abordadas as várias fases do desenvolvimento do sistema, que incluem a configuração dos Xbees, a elaboração do código para a recolha e tratamento dos dados, o envio desses dados através do protocolo MQTT e a criação do website utilizando o Node-RED e uma base de dados MySQL.

O quinto capítulo apresenta os testes que foram realizados e os resultados obtidos.

No sexto capítulo, é feita a apresentação da conclusão do trabalho, onde são resumidos os principais resultados obtidos em relação aos objetivos propostos. Também são destacadas as melhorias que este projeto oferece em relação ao projeto anterior. São discutidas as limitações do sistema e são apresentadas sugestões para possíveis trabalhos futuros nesta área.

2. Estado da Arte

O desenvolvimento de sistemas de monitorização tem sido uma área em constante crescimento no campo da investigação e desenvolvimento, com aplicações abrangentes em diversas áreas de estudo. A implementação de um sistema de monitorização eficiente desempenha um papel de extrema importância, uma vez que permite a obtenção de dados em tempo real, a monitorização do desempenho e a deteção de possíveis falhas ou problemas operacionais.

No âmbito deste projeto, a monitorização da estação meteorológica e da estação de painéis solares desempenha um papel fundamental na garantia do seu adequado funcionamento e desempenho. Através da recolha de dados precisos e atualizados, torna-se possível avaliar a eficiência energética, detetar variações climáticas e otimizar o desempenho das estações.

Ao combinar a monitorização da estação meteorológica e da estação de painéis solares, é possível estabelecer correlações entre os dados obtidos e obter informações mais abrangentes acerca da influência das condições climáticas na produção de energia renovável. Isso contribui para uma gestão mais eficiente dos recursos energéticos e uma melhor compreensão das tendências e padrões climáticos.

2.1. Sistemas de Monitorização

A transição para o digital tem impactado significativamente a sociedade e as empresas, revolucionando a forma como vivemos, comunicamos, nos conectamos e trabalhamos. A digitalização trouxe consigo uma série de benefícios, permitindo que as empresas alcancem novos mercados e expandissem os seus negócios. A automatização dos processos empresariais tem possibilitado uma maior eficiência operacional. Os avanços tecnológicos têm desempenhado um papel fundamental nessa transformação, sendo que os microcontroladores têm sido elementos essenciais nesse processo.

Os microcontroladores, impulsionados pelo constante aprimoramento das Unidades Centrais de Processamento (CPU), têm desempenhado um papel crucial para muitas empresas durante a transição para o ambiente digital. A capacidade de processar grandes volumes de dados em tempo real oferece às organizações a possibilidade de controlar e monitorizar os sistemas de forma eficaz, permitindo-lhes tomar decisões informadas e responder rapidamente às necessidades do mercado.

Esses dispositivos possuem uma vasta gama de funções, que vão desde a recolha e processamento de informações até ao controlo de dispositivos e sistemas automatizados. Com a sua poderosa capacidade de processamento, os microcontroladores possibilitam que as empresas monitorizem em tempo real variáveis importantes, tais como temperatura, pressão, humidade e outras. Essa capacidade de monitorização em tempo

real é essencial em setores como a indústria, logística, saúde e agricultura, onde a tomada de decisões precisa ser rápida e precisa.

Além disso, os microcontroladores são altamente flexíveis e podem ser programados para atender às necessidades específicas de cada empresa.

No passado, os microcontroladores apresentavam limitações consideráveis em termos de capacidade de processamento, memória e conectividade, sendo utilizados principalmente em ambientes industriais para recolha de dados e controle básico. No entanto, nos dias de hoje, os microcontroladores são capazes de processar grandes volumes de dados em tempo real e comunicar com outros dispositivos e sistemas por meio de protocolos de rede avançados. Essa evolução substancial possibilitou o desenvolvimento de sistemas de monitorização mais inteligentes e eficientes. Atualmente, os microcontroladores lidam com tarefas complexas, fornecendo recursos avançados de processamento, armazenamento e conectividade.

Atualmente, essa tecnologia está presente em diversas áreas, como:

Monitorização Ambiental:

Os sistemas de monitorização são amplamente utilizados para acompanhar a qualidade do ar, os níveis de poluição sonora, a temperatura, a humidade e outros parâmetros ambientais. Sensores e dispositivos são estrategicamente colocados em áreas específicas para recolher dados e transmiti-los para um sistema central, permitindo uma análise em tempo real e a identificação de padrões e tendências.

Estes sistemas desempenham um papel fundamental na avaliação contínua das condições ambientais, possibilitando a adoção de medidas adequadas para mitigar impactos negativos e promover uma melhor qualidade de vida. Através da análise detalhada dos dados recolhidos, é possível obter uma compreensão abrangente das variações e tendências ambientais, fornecendo informações valiosas para a implementação de políticas de conservação, medidas de controlo de poluição e otimização da gestão ambiental.

Exemplo:

Figura 1 - Sistema de Monitorização Ambiental

Fonte: https://bettaircities.com/

A plataforma IoT Bettair® [1] ilustrada na Figura 1 constitui uma ferramenta avançada para o mapeamento e monitorização da poluição em larga escala. Por meio de uma rede de dispositivos estáticos altamente precisos, esta plataforma integra microcontroladores da família STM32, o protocolo de comunicação LoRaWAN e uma variedade de sensores, incluindo sensores eletroquímicos e o Contador Ótico de Partículas.

Esta combinação de tecnologias permite à plataforma Bettair® disponibilizar informações valiosas acerca da qualidade do ar, incluindo dados detalhados sobre a concentração de gases, partículas suspensas no ar e níveis de poluição sonora. Tais informações são essenciais para que cidades inteligentes e outras entidades possam adotar medidas concretas para mitigar a poluição e criar ambientes mais saudáveis para os seus habitantes.

Monitorização Energética

Os sistemas de monitorização do consumo de energia em edifícios, instalações industriais ou residenciais desempenham um papel crucial na promoção da eficiência energética. Estes sistemas são projetados para medir e registar minuciosamente o consumo de eletricidade, gás, água e outras fontes de energia, com o objetivo de fornecer informações valiosas sobre a utilização eficiente dos recursos e auxiliar na identificação de áreas de desperdício ou oportunidades de poupança. Através da análise detalhada dos dados recolhidos, é possível detetar

ineficiências e propor medidas corretivas que visem maximizar a eficiência energética e reduzir os custos associados ao consumo descontrolado.

Exemplo:

Figura 2 - Contador Inteligente

Fonte: https://www.edp.pt/particulares/apoio-cliente/contadores/

Os contadores [2] inteligentes, conforme representados na Figura 2, são uma nova geração de medidores que oferecem benefícios significativos em termos de economia de energia, tempo e dinheiro. Eles permitem a comunicação automática e remota com o operador da rede de distribuição, eliminando a necessidade de comunicação de leituras e garantindo uma faturação precisa com base no consumo real. Isso torna o processo de medição e faturação mais eficiente, permitindo aos consumidores um melhor controlo de seus gastos energéticos e contribuindo para a sustentabilidade ambiental.

Monitorização de Tráfego

Os sistemas utilizados em cidades e autoestradas têm um papel crucial na gestão do fluxo de veículos e na melhoria da segurança nas estradas. Estes sistemas incluem a instalação estratégica de sensores e câmaras para monitorizar o tráfego em tempo real, recolhendo dados sobre a densidade do tráfego, a velocidade dos veículos e a identificação de incidentes.

Através destes sistemas avançados de monitorização, é possível obter informações valiosas sobre o tráfego rodoviário, permitindo uma melhor compreensão das condições de circulação e a identificação de potenciais pontos problemáticos. A análise em tempo real dos dados recolhidos possibilita a implementação de medidas

pró-ativas, tais como a gestão do fluxo de veículos, a adaptação dos tempos de sinalização e a comunicação eficaz de informações aos condutores.

Para além disso, a identificação de incidentes, como acidentes de trânsito ou obstruções nas vias, permite uma resposta rápida e eficiente dos serviços de emergência e ajuda a minimizar o impacto negativo na fluidez do tráfego.

Exemplo:

Figura 3 - Controlador inteligente do tráfego

Fonte: https://www.phoenixcontact.com/pt-pt/industrias/aplicacoes/gestao-de-trafego-inteligente

Até ao momento, ainda não existe nenhuma solução comercial disponível especificamente dedicada ao monitoramento de tráfego. Contudo, têm sido consideradas algumas tecnologias para essa funcionalidade, como por exemplo:

Sensores de Tráfego Inteligentes: Dispositivos instalados ao longo da estrada que utilizam tecnologias como indução magnética, radar ou ultrassom para detetar a presença de veículos e recolher dados relacionados, como velocidade e volume de tráfego.

Câmaras de Tráfego: Câmaras de alta resolução instaladas em postes ou estruturas elevadas ao lado das vias. Estas câmaras capturam imagens do tráfego em tempo real e, em alguns casos, utilizam software de reconhecimento de matrículas para identificar veículos e monitorizar o cumprimento de regulamentos de trânsito.

Monitorização na Saúde

Na área da saúde, são utilizados sistemas de monitorização para acompanhar os pacientes em tempo real. Dispositivos médicos portáteis recolhem dados vitais, como frequência cardíaca, pressão arterial e níveis de glicose, transmitindo essas informações para profissionais de saúde que podem monitorizar e intervir, se necessário.

Estes sistemas de monitorização desempenham um papel fundamental na prestação de cuidados de saúde personalizados e na melhoria da qualidade de vida dos pacientes. Através da utilização de dispositivos portáteis, os dados vitais são recolhidos de forma contínua e transmitidos de forma segura aos profissionais de saúde, permitindo-lhes acompanhar de perto o estado de saúde do paciente.

Estes sistemas de monitorização na área da saúde desempenham um papel crucial no acompanhamento contínuo dos pacientes, na prevenção de complicações e na promoção da saúde e do bem-estar.

Exemplo:

Figura 4 - Sistema de Monitorização da Glicose

Fonte: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQnf_i-QGJRPkyD3EhFmzywdH8A3Pp2ivyogw&usqp=CAU

O Sistema de Monitorização da Glicose da FreeStyle [4], ilustrado na Figura 3, é uma tecnologia inovadora que permite às pessoas com diabetes monitorizar os seus níveis de glicose de forma precisa e conveniente, sem a necessidade de picadas nos dedos. Com um pequeno sensor colocado na pele, o dispositivo fornece leituras contínuas e em tempo real, permitindo que os utilizadores tomem decisões informadas sobre o seu estilo de vida e cuidados de saúde. O sistema proporciona liberdade, controlo e tranquilidade às pessoas com diabetes.

Monitorização em Processos Industriais

Na indústria, implementam-se sistemas de monitorização para controlar e otimizar variáveis como temperatura, pressão, nível de líquidos e vibração em máquinas e equipamentos. Estes sistemas permitem detetar precocemente falhas e maximizar a eficiência da operação. Através de sensores e dispositivos adequados, monitoriza-se em tempo real as variáveis essenciais, possibilitando manutenções preventivas, reduzindo paragens inesperadas e otimizando a utilização de recursos. Assim, estes sistemas contribuem para o funcionamento seguro e eficiente dos equipamentos industriais.

Exemplo:

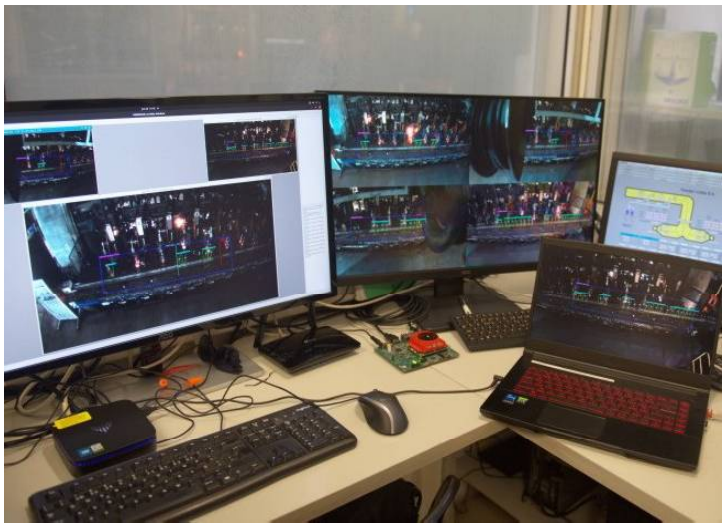


Figura 5 - Sistema Viexpand

Fonte: https://www.jornaldeleiria.pt/noticia/twevo-startup-de-leiria-cria-sistema-inteligente-de-producao-para-a-gallo-vidro

A *startup* TWEvo [5], em parceria com o Instituto de Telecomunicações/Instituto Politécnico de Leiria, desenvolveu o sistema Viexpand, conforme ilustrado na Figura 4. Este sistema foi implementado na Gallo Vidro, uma empresa de fabrico de vidro automático, com o objetivo de supervisionar remotamente a produção de garrafas. O Viexpand utiliza câmaras de alta resolução e inteligência artificial para verificar e medir a distância entre as garrafas, evitando problemas de colagem e bloqueios na linha de produção. Além disso, o sistema melhora o conforto dos

trabalhadores e contribui para a sustentabilidade ao reduzir o desperdício e o consumo de energia.

2.2.Simonic

Conforme mencionado no resumo, o presente projeto tem como objetivo dar continuidade ao projeto anteriormente designado por SIMONIC [6] [7]. O SIMONIC consistia num sistema de monitorização que recolhia dados ambientais através de sensores, tais como velocidade do vento, direção do vento, temperatura ambiente, radiação, assim como dados provenientes de um painel fotovoltaico térmico (PVT), como temperatura e tensão, entre outros. Além disso, o sistema monitorizava o fluxo de água da bomba que passava pelo painel, conforme ilustrado na Figura 5.

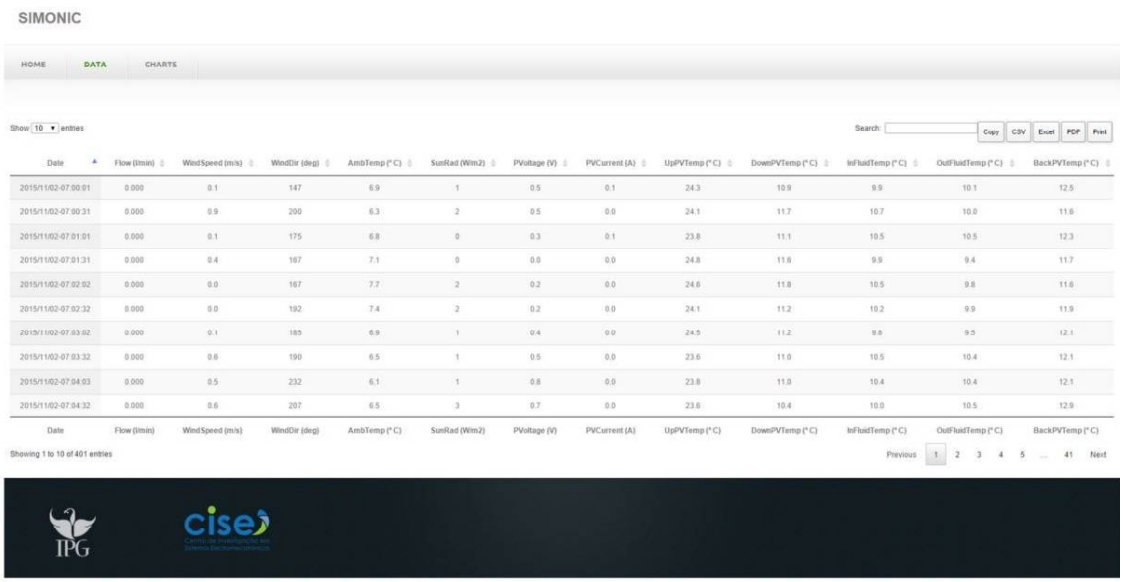


Figura 6 - SIMONIC

Fonte: Saraiva, L. M. G. A. T. (2016). SIMONIC: Sistema de Monitorização e Controlo de Sistema Solar. Relatório de Estágio Profissionalizante. Escola Superior de Tecnologias e Gestão, Instituto Politécnico da Guarda. (Figura 37)

Para a concretização deste projeto, foram utilizadas duas placas com microcontroladores, comunicação de dados sem fios e computação na nuvem, de forma a permitir o acesso remoto em tempo real, permitindo a possibilidade de controlo do sistema de aquisição e a interrupção da aquisição de dados. Este projeto demonstrou a viabilidade da utilização de uma plataforma de baixo custo, dispensando a necessidade de aquisição de sistemas dedicados mais dispendiosos.

Os resultados obtidos neste estudo têm relevância a nível académico, ao comprovar a possibilidade de utilização de uma plataforma acessível e eficiente para a monitorização de dados ambientais e energéticos. Tal contribuição representa uma alternativa viável e economicamente vantajosa na área em questão.

Estes são alguns exemplos de sistemas de monitorização em diferentes áreas. Neste projeto, temos simultaneamente um sistema de monitorização ambiental e energético. A importância destes sistemas reside sempre na possibilidade de aumentar a eficiência da produção, mas também de diagnosticar possíveis falhas ou problemas operacionais, mantendo também a segurança.

3. Arquitetura do Sistema

Comparativamente ao projeto SIMONIC, existem algumas diferenças significativas na arquitetura do novo projeto. Em vez de recolher dados apenas de uma estação, serão consideradas duas estações, o que requer a utilização de 3 placas com microcontroladores em vez de 2. Relativamente ao projeto anterior, mantém-se a recolha dos seguintes dados: a obtenção da temperatura ambiente através do termistor de 10 KΩ, a leitura da radiação solar através do piranómetro e a medição da velocidade e direção do vento através do anemómetro e do sensor de direção do vento. Além dos dados mencionados, será recolhida também a temperatura e humidade através de um sensor da Adafruit, bem como a pressão, luz ambiente, IR e UV através de um sensor SparkFun. Será ainda calculado o valor de Lux. As especificações de cada sensor podem ser consultadas no capítulo 3.3.

Em vez de criar o site através do Coordenador, os dados são enviados para o NodeRed, onde toda a componente do website será desenvolvida. Outra diferença relevante reside na escolha da base de dados. Em vez de utilizar folhas de cálculo Excel, como anteriormente, será utilizado um sistema de gestão de bases de dados MySQL para armazenar as informações. Esta alteração proporciona maior robustez e flexibilidade na gestão dos dados recolhidos. Adicionalmente, será utilizado um módulo RTC (*Real Time Clock*) para obter a hora e a data, em vez de depender da obtenção desses dados através da Internet como era feito anteriormente. Desta forma, mesmo que ocorra uma perda de conexão com a rede Wi-Fi, os dados serão corretamente registados e guardados no cartão SD de forma adequada, evitando a perda de informação.

Estas são as principais diferenças em relação ao projeto anterior, permitindo uma evolução do sistema. A Figura 6 apresenta a arquitetura do sistema atual.

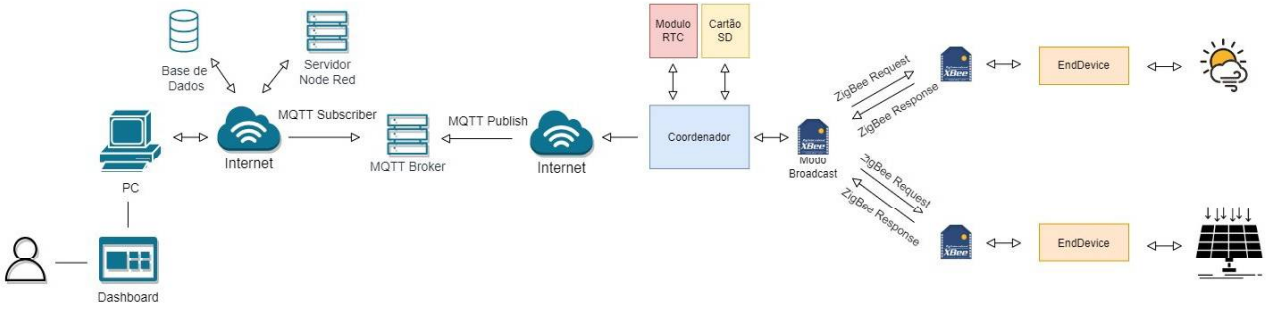


Figura 7 - Arquitetura do Sistema

3.1.Protocolos de Comunicação sem fio

Os protocolos de comunicação sem fios desempenham um papel fundamental na troca de informações entre dispositivos eletrónicos, permitindo a transmissão de dados de forma eficaz e confiável. Neste subcapítulo, serão explorados alguns dos principais protocolos de comunicação sem fios, destacando as suas características e usos específicos. A Tabela 1, apresentada abaixo, oferece uma visão geral de alguns dos protocolos mais conhecidos.

Existem diversos protocolos amplamente conhecidos que oferecem diferentes vantagens e são adequados para vários cenários de aplicação. A escolha do protocolo deve depender dos requisitos específicos do projeto, como alcance de comunicação, consumo de energia, largura de banda necessária e tipos de dispositivos envolvidos. É importante ter em consideração estes aspetos ao selecionar o protocolo mais adequado para cada caso.

Tabela 1 - Protocolos de Comunicação sem fio

Protocolo	Características	Usos Específicos
Wi-Fi (IEEE 802.11)	Conexão rápida e estável, alta velocidade de transferência de dados	*Streaming* de media, transferência de ficheiros, acesso à internet
Bluetooth	Comunicação sem fio de curto alcance, eficiência energética	Dispositivos móveis, fones de ouvido sem fio, colunas portáteis
ZigBee	Baixo consumo de energia, comunicação de curto alcance, troca de dados em malhas (Mesh)	Redes de sensores, automação residencial, controlo de iluminação
LoRaWAN	Comunicação sem fio de longo alcance e baixo consumo de energia	Internet das Coisas (IoT), monitorização ambiental, agricultura inteligente

Neste projeto, serão utilizados três protocolos de comunicação: Zigbee, Wi-Fi e MQTT. O protocolo Zigbee será usado para a troca de mensagens entre os microcontroladores, enquanto o protocolo Wi-Fi será utilizado para ligar o coordenador a uma rede, permitindo a posterior publicação dos dados através do protocolo MQTT.

O protocolo ZigBee foi utilizado tendo em conta que a maioria das placas disponíveis não possuía qualquer tipo de comunicação embutida, enquanto os módulos Xbee estavam disponíveis. O protocolo MQTT foi escolhido justamente

por se tratar de um protocolo que incorpora duas características importantes para este projeto: escalabilidade e confiabilidade. Essas características permitem conectar mais dispositivos e, em caso de trabalhos futuros, manter a confiança de que os dados serão remetidos corretamente. Para mais informações sobre o protocolo MQTT e seu funcionamento, recomenda-se a consulta do capítulo 3.4.

Para um melhor entendimento das diferenças entre esses protocolos, a Figura 7 faz referência aos modelos OSI e TCP/IP, indicando em que camada esses protocolos e outros exemplos se encontram.

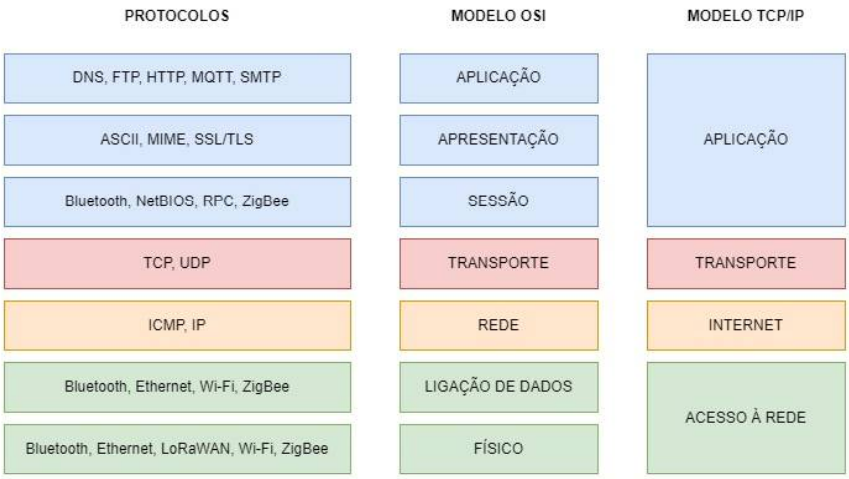


Figura 8 - Modelo OSI vs TCP/IP

Adaptado de: https://paginas.fe.up.pt/~mrs01003/TCP_IP.htm e https://community.cisco.com/legacyfs/online/legacy/4/1/0/136014-proto.jpg

3.2. Microcontoladores

Após selecionar os protocolos de comunicação mais adequados é necessário escolher os microcontroladores que serão utilizados, tendo em conta as especificações dos dispositivos. A Tabela 2, apresentada abaixo para fins comparativos, inclui alguns modelos de microcontroladores de diferentes fabricantes, juntamente com informações sobre as suas características:

Tabela 2 - Especificações de várias placas

Modelo	I2C	Pinos Analógicos	Pinos Digitais	RAM/ SRAM	Memória Flash	Wi-Fi
Arduino						
Yún Rev 2 [8]	Sim	12	20	64MB	16MB	Sim
Mega 2560 Rev3 [9]	Sim	16	54	8KB	256KB	Não
UNO Rev3 [10]	Sim	6	14	2KB	32KB	Não
Leonardo [11]	Sim	12	20	2.5KB	32KB	Não
Due [12]	Sim	12	54	96KB	512KB	Não
Portenta H7 Lite [13]	Sim	8	22	1MB	2MB	Sim
Portenta X8 [14]	Sim	8	22	2GB	16GB	Sim
Espressif Systems						
ESP8266 [15]	Sim	1	17	160KB	16MB	Sim
ESP32 [15]	Sim	18	36	512KB	16MB	Sim
Teensy						
Teensy 4.0 [16]	Sim	14	40	1MB	2MB	Não
Teensy 4.1 [17]	Sim	18	55	1MB	7.9MB	Sim

No contexto deste trabalho, com base no equipamento disponível no laboratório, foram utilizadas duas placas de desenvolvimento Arduino Mega 2560 e uma placa Arduino Yun. O Arduino Yun foi selecionado como coordenador devido à sua

capacidade de conexão Wi-Fi embutida, uma vez que era necessário enviar os dados para o NodeRed por meio dessa rede sem fio.

No que diz respeito aos Arduinos Mega, que foram utilizados como *endDevices* neste caso, tínhamos várias opções disponíveis para escolha. A tabela 3 apresenta as placas de desenvolvimento disponíveis, juntamente com as justificativas para não terem sido selecionadas outras placas:

Tabela 3 - Placas de Desenvolvimento Disponiveis

Placa de Desenvolvimento	**Motivos para não ser selecionada**
Arduino Due	Foi observado que a placa em questão estava enfrentando problemas na comunicação I2C. Tanto as portas SDA0 e SCL0 quanto as portas SDA1 e SCL1 não conseguiam estabelecer a conexão. Vale ressaltar que os sensores utilizam endereços diferentes, excluindo assim a possibilidade de colisão.
Arduino Uno	Não possui a capacidade de processamento necessária.
Arduino Duemilnove	Os pinos SDA e SCL estão localizados nos pinos analógicos 4 e 5, o que impossibilita a recolha de todos os dados desejados, uma vez que a estação meteorológica requer um mínimo de 5 pinos analógicos.
Arduino Mega 2560	Nada a apontar.
Arduino Leonardo	Embora o Leonardo atenda a todas as condições necessárias para integrar o projeto, foram identificadas falhas nas portas de comunicação em duas unidades. Essas falhas podem estar relacionadas ao uso, tornando-se um problema crítico na utilização do projeto.
Arduino 101	Incompatibilidade com bibliotecas da Adafruit usadas no projeto.

Com base nas considerações apresentadas anteriormente, devidamente analisadas e ponderadas, a escolha final recaiu sobre o Arduino Mega.

3.3.Sensores

Os sensores são dispositivos essenciais que recolhem informações do ambiente e as convertem em sinais ou dados mensuráveis. Desempenham um papel fundamental na recolha de dados precisos e fiáveis, sendo projetados para detetar uma vasta gama de informações, como temperatura, pressão, humidade, luz, movimento e muito mais. Cada tipo de sensor é especialmente desenvolvido para cumprir um propósito específico e pode variar em termos de princípio de funcionamento e tecnologia utilizada.

Estes dispositivos operam transformando grandezas físicas em sinais elétricos ou digitais que podem ser interpretados e processados por outros sistemas. Os avanços na tecnologia dos sensores têm sido notáveis nos últimos anos. Temos assistido ao surgimento de sensores cada vez mais sensíveis, compactos e inteligentes.

A tabela 4 a seguir descreve os sensores utilizados no trabalho, juntamente com suas características, gama de medições e tipos.

Conforme mencionado no capítulo 3, alguns dos sensores apresentados na tabela 4 já estavam a ser utilizados no projeto SIMONIC. Adicionalmente, foram adicionados o sensor SI1145 e o sensor SparkFun *Weather Shield*. Estes sensores foram incluídos com o intuito de obter uma visão mais completa das condições ambientais, permitindo recolher dados adicionais relevantes para o projeto.

Tabela 4 - Sensores utilizados no projeto

Nome do Sensor	Características	Tipo
Si1145 [18]	Sensor de luz e proximidade Gama de medição de luz: 0 lux a 40.000 lux Gama de medição de proximidade: até 20 cm	Digital, I2C
PT100	Sensor de temperatura por resistência Gama de medição de temperatura: -200°C a +850°C	Analógico
Termistor de 10 kΩ	Sensor de Temperatura por resistência Gama de medição de temperatura: -20°C a +50°C	Analógico

Anemômetro	Sensor de velocidade do vento Gama de medição de velocidade do vento: 0 m/s a 30 m/s	Analógico
Direção do Vento	Sensor de direção do vento Gama de medição de direção do vento: 0° a 360°	Analógico
Piranómetro	Sensor de radiação solar Gama de medição de radiação solar: 0 a 1500W/m²	Analógico
SparkFun *Weather Shield*		
Si7021 [19]	Sensor de humidade e temperatura Gama de medição de humidade: 0% a 100% HR Gama de medição de temperatura: -40°C a +125°C	Digital, I2C
ALS-PT19 [20]	Sensor de luz Gama de medição de luz: 0 lux a 40.000 lux	Digital, I2C
MPL3115A2 [21]	Sensor de pressão e altitude Gama de medição de pressão: 50 kPa a 110 kPa Gama de medição de altitude: -500 m a +9.000 m	Digital, I2C

3.4. Multiplexer

Os multiplexadores [22], também conhecidos como MUX, são dispositivos eletrónicos utilizados para combinar múltiplos sinais de entrada num único sinal de saída. Desempenham um papel fundamental na área de sistemas digitais, mas também analógicos, sendo amplamente utilizados em circuitos lógicos, processadores, sistemas

de comunicação e muito mais. Neste capítulo, exploraremos os conceitos básicos dos multiplexadores, as suas aplicações e o seu funcionamento interno.

3.4.1. Estrutura e Funcionamento de Multiplexador

Um multiplexador possui várias entradas, uma ou mais entradas de controlo e uma única saída. O número de entradas é determinado pela quantidade de bits necessários para selecionar uma das entradas como saída. Por exemplo, um multiplexador de 4 para 1 requer 2 bits de controlo para selecionar uma das quatro entradas.

Internamente, um multiplexador consiste numa rede de portas lógicas, como portas E (AND) e portas OU (OR). Estas portas são configuradas para que apenas um caminho entre as entradas e a saída seja ativado de cada vez, dependendo do valor dos bits de controlo. Os restantes caminhos são desativados.

O sinal de controlo é utilizado para selecionar qual das entradas será transmitida para a saída. Quando os bits de controlo são alterados, o multiplexador altera a sua configuração interna, direcionando o sinal de uma entrada específica para a saída. A Figura 8 demonstra um diagrama lógico de um multiplexador 4 para 1.

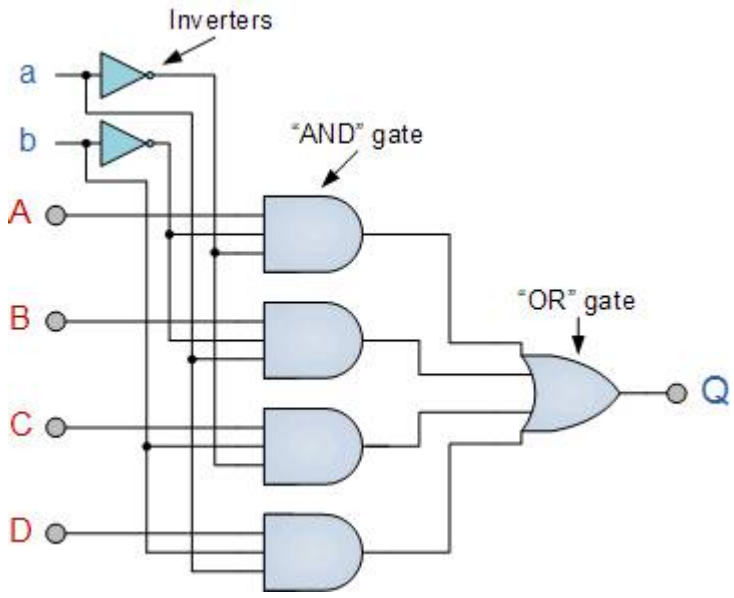


Figura 9 - Diagrama Lógico de MUX digital

Existem diferentes tipos de multiplexadores, variando em termos do número de entradas e bits de controle. De forma geral, um multiplexador n para 1 representa um dispositivo com n entradas (onde n pode ser 2, 4, 8, etc.) e log2(n) bits de controle (1, 2, 3, etc.).

3.5. MQTT

O protocolo MQTT [23] é um protocolo de mensagens leve, especialmente projetado para aplicações IoT, uma vez que requer recursos mínimos e pode ser utilizado em pequenos microcontroladores. Adota um modelo de publicação e subscrição, no qual os dispositivos podem atuar como publicadores, enviando mensagens em tópicos específicos, ou como subscritores, recebendo as mensagens publicadas nos tópicos aos quais estão subscritos.

Um elemento fundamental deste protocolo é o *broker*, que desempenha um papel central no sistema. O *broker* é responsável por receber, filtrar e encaminhar as mensagens para os subscritores, que são os clientes do MQTT. Um *broker* é capaz de lidar com milhões de clientes conectados, tornando-o escalável e adequado para aplicações em grande escala.

Esta abordagem possibilita a criação de um sistema de mensagens distribuído, no qual os dispositivos podem trocar informações de forma assíncrona. Isso facilita a criação de conectividade entre vários dispositivos ou sensores num ambiente IoT. Os clientes MQTT são os dispositivos que interagem com o *broker* para enviar ou receber mensagens.

A Figura 8 demonstra como funciona a arquitetura de publicação/subscrição.

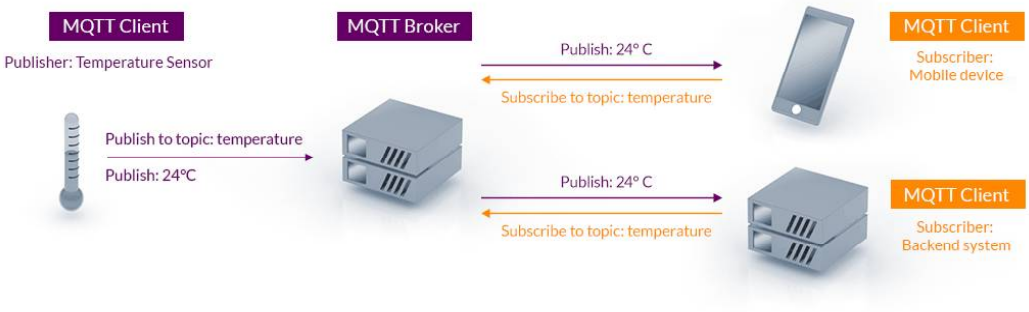


Figura 10 - Arquitetura de Publicação / Subscrição MQTT

Fonte: https://mqtt.org/

Uma das características distintivas do MQTT é o uso de um sistema de tópicos para organizar e encaminhar as mensagens.

No contexto do MQTT, o tópico é fundamental para a troca de informações. Ele funciona como um canal de comunicação que permite a identificação e categorização das mensagens transmitidas. O tópico desempenha um papel essencial ao definir o destino da mensagem, indicando quais dispositivos ou assinantes receberão as informações publicadas.

3.6.HTTP

O protocolo HTTP é um protocolo de comunicação utilizado para transferir informações na internet. É a base de comunicação entre os clientes e os servidores. Ele define a estrutura e o formato das mensagens trocadas entre o cliente e o servidor, permitindo que ambos se comuniquem e partilhem informações.

A Figura 9 ilustra a troca de mensagens entre o cliente e o servidor.

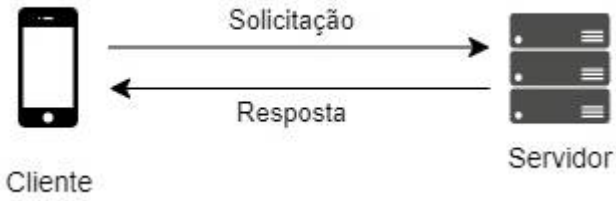


Figura 11 - Ciclo de Solicitação-Resposta

O protocolo HTTP utiliza um modelo de solicitação e resposta, no qual o cliente envia uma solicitação especificando o recurso desejado, como uma página web, e o servidor responde com os dados solicitados.

Quando um cliente deseja visualizar uma imagem online, ele envia uma solicitação utilizando um URL que indica o local no servidor onde a imagem está armazenada.

A Figura 10 ilustra a estrutura de um URL.

Figura 12 - HTTP Request

Fonte: https://www.freecodecamp.org/news/what-is-http/

Sem o HTTP, seria difícil imaginar o funcionamento da internet tal como a conhecemos hoje, uma vez que não existiriam páginas web, URL e hiperligações. Em vez disso, os

clientes precisariam de conhecer o endereço IP exato do servidor que hospeda as informações que desejam aceder.

3.7.ZigBee

O Zigbee é uma tecnologia sem fio desenvolvida para redes de baixo custo e baixo consumo de energia, especialmente projetada para aplicações de máquina a máquina (M2M) e Internet das Coisas (IoT). Baseia-se em padrões abertos e foi criado para atender a requisitos específicos, como baixas taxas de transferência de dados e eficiência energética.

3.7.1. Mesh Networking

O Zigbee é um protocolo que suporta redes mesh, que consistem em redes descentralizadas que utilizam dois arranjos de conexão: topologia de malha completa e topologia de malha parcial. Na topologia de malha completa, todos os nós da rede estão diretamente ligados entre si, enquanto na topologia de malha parcial, alguns nós estão ligados a todos os outros, mas outros estão ligados apenas aos nós com os quais têm maior interação de dados.

O protocolo Zigbee define três tipos de nós: coordenadores (ZC), *routers* (ZR) e dispositivos finais (ZED). O coordenador é responsável por armazenar informações sobre a rede, como chaves de segurança. Os *routers* são nós intermédios que retransmitem dados de outros dispositivos. Os dispositivos finais são dispositivos de baixa potência ou alimentados por bateria, que podem comunicar-se com o coordenador ou um *route*r, mas não são capazes de retransmitir dados de outros dispositivos.

Esta estrutura hierárquica e a capacidade dos *routers* de retransmitir dados permitem a criação de redes Zigbee flexíveis e robustas, com alcance estendido. Cada nó na rede pode atuar como um ponto de acesso para outros dispositivos, aumentando a cobertura e a confiabilidade da comunicação.

A Figura 11 demonstra um exemplo de uma possível rede para uma melhor compreensão da função de cada dispositivo.

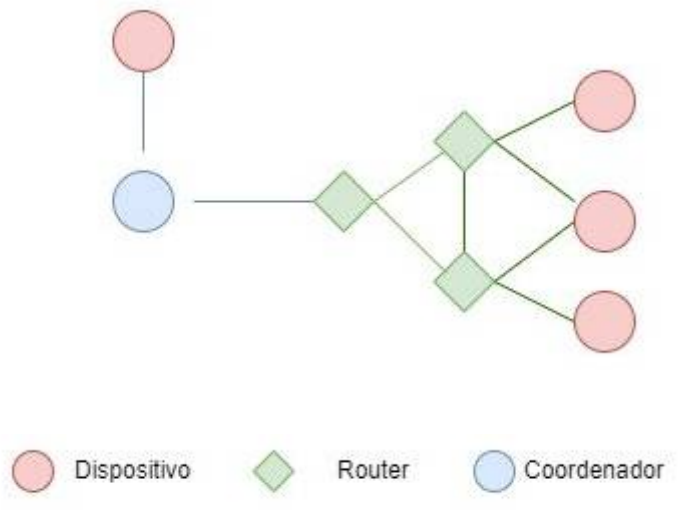


Figura 13 - Dispositivos ZigBee

Esta capacidade de rede mesh do Zigbee permite uma comunicação eficiente e fiável entre os dispositivos numa rede, garantindo uma cobertura abrangente e uma maior tolerância a falhas. Através da utilização de diferentes tipos de nós e da combinação de topologias de malha completa e parcial, o Zigbee oferece soluções adaptáveis e robustas para a comunicação de dispositivos interligados.

3.7.2. Topologias de rede

Em relação ao funcionamento, temos a possibilidade de implementar diferentes topologias de rede, adaptando-as às necessidades específicas do sistema. A Figura 12 representa algumas das topologias mais conhecidas, como a topologia estrela, árvore e malha. Essas topologias oferecem diferentes formas de organização e conexão entre os dispositivos em uma rede.

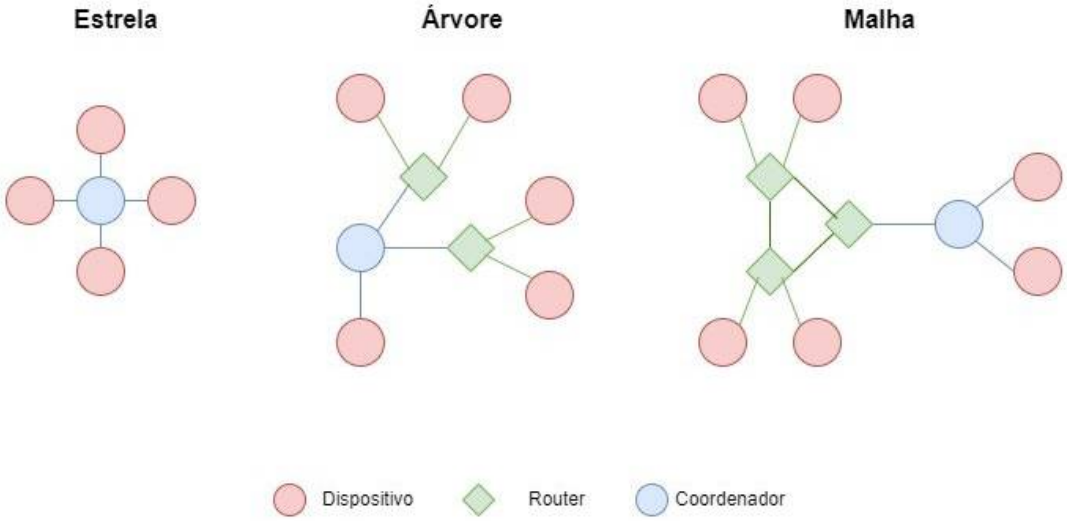


Figura 14 - Tipologia ZigBee

Estrela: Nesta topologia, o controlador é colocado no centro da rede, com todos os dispositivos conectados diretamente a ele.

Malha: Nesta topologia, o controlador é colocado na raiz e todos os dispositivos são conectados a ele, formando uma malha de conexões.

Árvore: Esta é a topologia mais interessante, pois os nós podem estar conectados a outros nós, criando vários caminhos para uma comunicação estável. Se um nó falhar, a rede é reorganizada pelo coordenador.

3.7.3. Vantagens do ZigBee

O Zigbee destaca-se como uma solução tecnológica atrativa para aplicações de IoT e M2M, oferecendo benefícios substanciais, como eficiência energética e escalabilidade da rede por meio da arquitetura em malha. Essas características tornam o Zigbee uma escolha viável para implementações que requerem baixo consumo de energia, ampla cobertura e capacidade de expansão.

4. Desenvolvimento e Implementação da Solução

Este capítulo está dividido em seis subcapítulos distintos. O primeiro subcapítulo é dedicado à descrição do hardware e software utilizados. O segundo subcapítulo aborda a configuração dos módulos Xbee. Os três subcapítulos seguintes são dedicados à aplicação do coordenador e dos dois dispositivos endDevices, sendo que cada um corresponde a uma estação específica. Por último, o sexto subcapítulo é dedicado ao desenvolvimento do website.

4.1. Hardware e Software Utilizado

Hardware

Conforme mencionado no capítulo 3.1, neste projeto foi escolhido utilizar o protocolo ZigBee. Para implementar esse protocolo, decidiu-se utilizar módulos XBee, que incorporam nativamente esse protocolo em seus dispositivos.

Com o objetivo de determinar quais módulos XBee seriam integrados ao projeto, foram realizados testes de comunicação entre diferentes modelos, como XBee Pro S2C, Pro S2, Pro S1 e MaxStream 1 com antena embutida. Durante os testes, observou-se que ao estabelecer a comunicação entre dispositivos de séries diferentes, a fluidez na transmissão dos dados era menor.

Após uma série de testes, verificou-se que o módulo XBee Pro S2C demonstrou um desempenho superior em termos de fluidez na comunicação ao estabelecer conexão com um módulo idêntico. Com base nesses resultados decidiu-se selecionar o XBee Pro S2C como uma opção adequada para o projeto.

No contexto deste projeto, utilizou-se o Arduino Yun como coordenador da rede, devido à sua funcionalidade Wi-Fi incorporada. Os dispositivos *EndDevices* são ambos Arduinos Mega. Para compreender melhor as razões por trás da escolha dessas placas, recomenda-se a consulta do capítulo 3.2.

Além disso, foi adicionado um cartão SD ao Arduino Yun para fins de backup. Os detalhes sobre os sensores utilizados podem ser encontrados no capítulo 3.3 do projeto. Esses componentes foram selecionados visando garantir uma comunicação sem fio eficiente e confiável, além de assegurar a segurança dos dados e a integridade do sistema como um todo.

É importante mencionar que foram utilizados dois multiplexadores 8 para 1 para recolher os dados de tensão dos painéis solares. Essa escolha foi necessária devido

à limitação de pinos analógicos na placa Arduino Mega, que não permitia a conexão direta dos valores de tensão dos painéis. Os multiplexadores foram utilizados para expandir a capacidade de entrada analógica, possibilitando a leitura e o registo adequados dos dados de tensão dos painéis solares.

Software

Para a configuração dos módulos XBee, utilizamos o software XCTU, desenvolvido pelo fabricante Digi, que é especificamente destinado a essa finalidade. O XCTU permite configurar e testar os módulos XBee de maneira fácil e intuitiva.

Quanto ao desenvolvimento do código, optamos por utilizar o Arduino IDE, uma plataforma amplamente conhecida e utilizada. O Arduino IDE oferece uma interface de programação amigável e eficiente para desenvolver o código necessário no projeto. A biblioteca de funções e recursos do Arduino IDE facilita a criação e implementação do código nos dispositivos Arduino utilizados no projeto.

4.2.Configuração dos Modulos XBee

A configuração dos módulos Xbee é realizada através do uso de placas de configuração. A Figura 14 ilustra uma das placas utilizadas para configurar os módulos. O programa XCTU da Digi é utilizado para essa configuração. O XCTU está disponível para as plataformas Linux, MacOS X e Windows, o que proporciona flexibilidade aos utilizadores na escolha do sistema operativo desejado. Ao posicionar o módulo Xbee na placa e conectá-lo a um computador, é possível adicionar um novo módulo de rádio no

programa ou utilizar a função "discover" para identificar e conectar automaticamente o módulo Xbee.

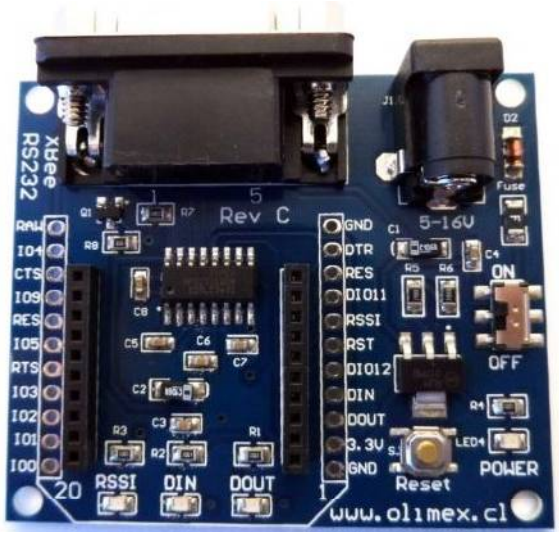


Figura 15 - Placa de configuração

Fonte: https://xbee.cl/wp-content/uploads/2014/10/SAM_1358.jpg

Assim que o módulo estiver conectado, o primeiro passo é selecionar o modo de funcionamento do módulo e escolher a versão do firmware desejada. A utilização da versão mais recente do firmware pode trazer melhorias no desempenho e correções de bugs, por isso é sempre aconselhável utilizar a versão mais recente.

A opção para realizar essa configuração está ilustrada na figura 15, enquanto a figura 16 apresenta os modos de funcionamento disponíveis no módulo Xbee Pro S2C.

Os modos de funcionamento ligados ao hardware são:

802.15.4 TH:

O modo de configuração 802.15.4 TH refere-se à especificação IEEE 802.15.4, que é um padrão de rede sem fios de baixa potência e curto alcance. Os módulos Xbee 802.15.4 TH operam nessa faixa de frequência e são projetados para aplicações simples de comunicação ponto a ponto ou em rede

de malha. Eles suportam topologias de rede em estrela ou malha, onde um dispositivo central atua como coordenador da rede.

DigiMesh 2.4 TH:

O modo de configuração DigiMesh 2.4 TH é uma variante do modo DigiMesh disponível nos módulos Xbee. DigiMesh é uma tecnologia proprietária da Digi International que permite criar redes de malha auto-organizáveis. Os módulos Xbee DigiMesh 2.4 TH operam na faixa de frequência de 2,4 GHz e podem ser configurados para formar uma rede de malha em que cada nó pode atuar como roteador para outros nós, permitindo uma cobertura mais ampla.

ZIGBEE TH Reg:

O modo de configuração ZIGBEE TH Reg refere-se ao suporte ao protocolo Zigbee nos módulos Xbee. O Zigbee é um padrão de comunicação sem fios baseado no IEEE 802.15.4 e é projetado para aplicações de baixo consumo de energia e baixa taxa de dados. O modo ZIGBEE TH Reg significa que os módulos Xbee suportam o perfil Zigbee Home Automation (Zigbee HA), que é usado em aplicações residenciais e domésticas, como automação residencial, controle de iluminação, segurança, entre outros.

Figura 16 - Ícone para configurar

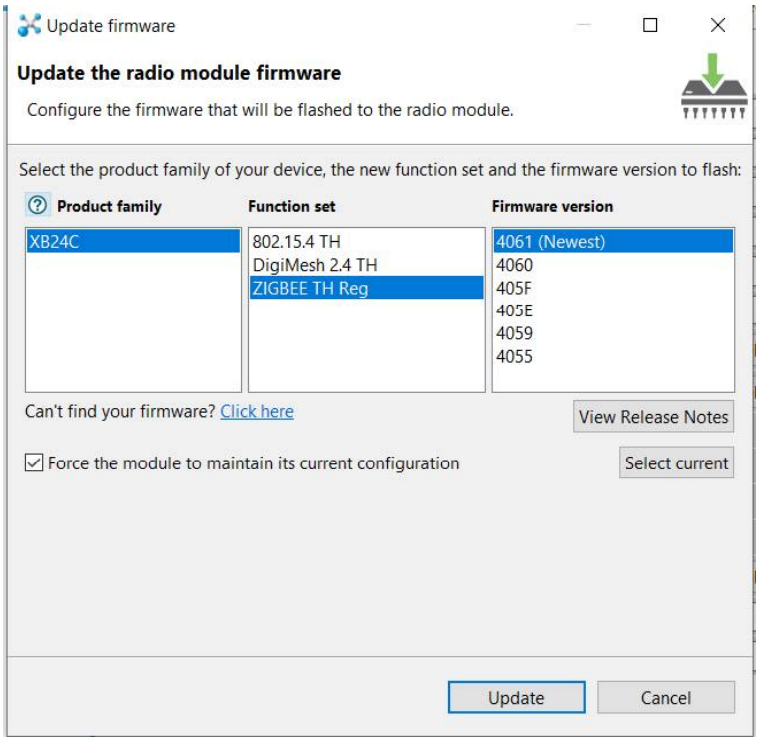


Figura 17 - Seleção do modo de funcionamento e o firmware

No contexto do projeto, foi selecionada a família XB24C no modo ZIGBEE TH Reg, juntamente com o firmware 4061. Essa escolha específica visa atender às necessidades do projeto em questão.

Para iniciar a configuração da rede, é necessário configurar o PAN ID. A configuração do PAN ID permite a criação de uma rede exclusiva, reduzindo possíveis interferências de outras redes sem fio próximas. É essencial para garantir uma comunicação eficiente e confiável entre os dispositivos envolvidos. Com a configuração correta do PAN ID, os módulos Xbee estarão preparados para estabelecer uma conexão sólida e trocar informações de forma consistente.

A opção de configuração do PAN ID pode ser encontrada na figura 17, ilustrada a seguir.

Figura 18 - Configuração do PAN ID

Além da configuração do ID PAN, também foram realizadas as configurações do DH (*Destination Address High*) e DL (*Destination Address Low*), que compõem o endereço do módulo roteador.

O DH (Destination Address High) representa a primeira parte do endereço do módulo roteador, enquanto o DL (Destination Address Low) representa a segunda parte do endereço. Essas configurações são essenciais para estabelecer a comunicação entre os módulos Xbee dentro da rede.

Ao definir corretamente o DH e o DL, os módulos Xbee serão capazes de identificar e encaminhar de forma precisa as mensagens e dados para os destinos desejados dentro da rede estabelecida.

Isso é especialmente importante em redes maiores, onde vários módulos Xbee estão interconectados. A configuração adequada do DH e DL garante que os dados sejam encaminhados corretamente entre os dispositivos, evitando perdas ou erros de comunicação.

O endereço do módulo utilizado pode ser visualizado diretamente no próprio módulo, conforme ilustrado na figura 18, ou através do XCTU, como demonstrado na figura 19.

Figura 19 - Address do Modulo

Fonte: https://www.makerhero.com/blog/tutorial-wireless-arduino-xbee-shield/

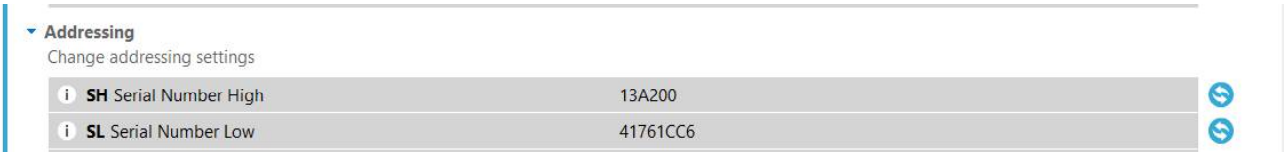


Figura 20 - Address do Modulo no XCTU

Também é necessário configurar o campo "Coordinator Enable" como "enable" ou "disable", dependendo se o módulo Xbee está sendo utilizado como coordenador ou não.

O coordenador tem a responsabilidade de controlar e coordenar a comunicação entre os dispositivos dentro da rede estabelecida. Em redes maiores, é comum ter um ou mais módulos Xbee configurados como coordenadores para facilitar a gestão e a comunicação entre os dispositivos.

Ao selecionar a opção "Test" no programa XCTU e abrir as portas de comunicação dos dois Xbees, é possível estabelecer a comunicação entre os dispositivos e iniciar a troca de informações. Esses testes permitem verificar a conectividade e a funcionalidade dos módulos Xbee, garantindo que estejam configurados corretamente e prontos para operar na rede.

A figura 20 e 21 mostram a seleção da opção "Test" e a abertura da porta de comunicação.

Figura 21 - Seleção da opção para testar a comunicação entre modulos

Figura 22 - Abertura da porta de comunicação

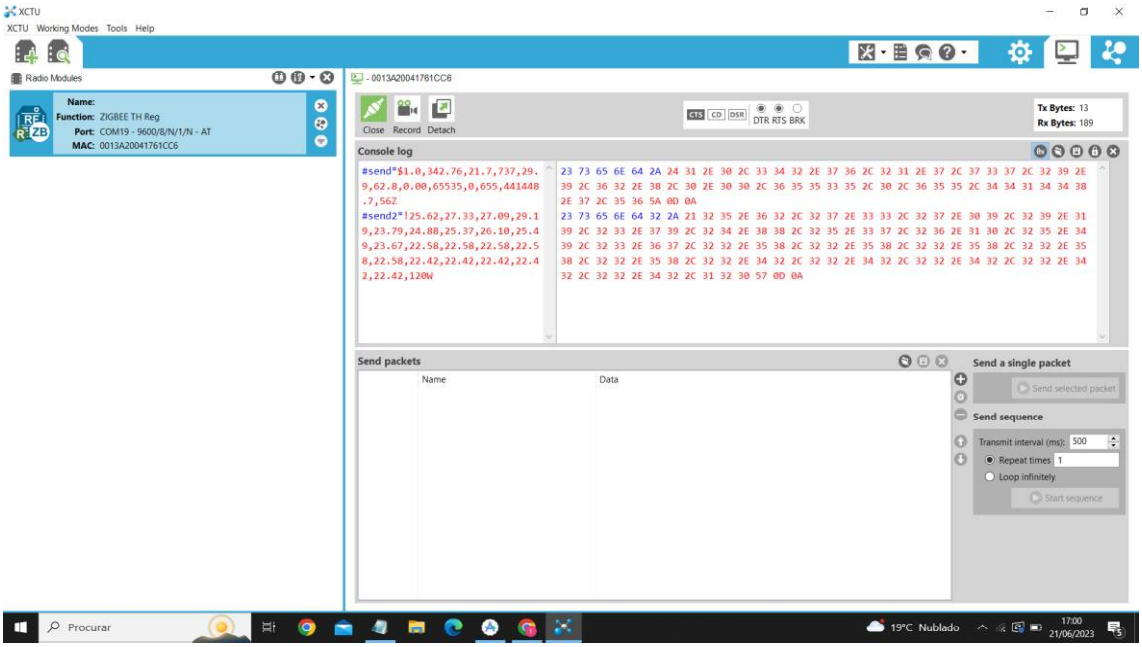


Figura 23 - Exemplo de como é realizado os pedidos

A Figura 22 representa um teste realizado a partir do coordenador da rede, onde um pedido manual foi feito usando as sequências de caracteres "#send*" e "#send2*". Neste teste, a configuração dos dispositivos EndDevices já havia sido concluída, permitindo a receção das respostas com os dados dos sensores.

É relevante mencionar que este sistema opera no modo *Broadcast*, ou seja, a mensagem é enviada para vários destinatários, neste caso, os dispositivos EndDevices. Para evitar colisões de dados, no primeiro pedido, ambos os EndDevices são instruídos a recolher os dados, mas apenas o *EndDevice* da estação meteorológica envia os dados, enquanto o *EndDevice* da estação de painéis solares espera pelo segundo pedido para enviar os dados adquiridos. Isso garante a integridade dos dados, evitando colisões e permitindo a recolha simultânea dos mesmos.

4.3. Aplicação do Coordenador

Neste subcapítulo, será apresentada a configuração e aplicação do coordenador da rede. Para isso, foram criados subcapítulos nos quais é destacado o fluxograma das funções básicas do código para Arduino, tanto do Setup quanto do Loop, a fim de facilitar a compreensão do funcionamento do código desenvolvido. Em seguida, abordamos a comunicação com os *EndDevices*, a obtenção da data e hora por meio do módulo RTC, a recepção e validação das mensagens, o armazenamento no cartão SD e a conexão à rede Wi-Fi, para que posteriormente seja possível enviar as mensagens por meio do protocolo MQTT.

4.3.1. Fluxograma

A Figura 23 e 24 apresentam o fluxograma do setup e do loop, visando facilitar a compreensão de seu funcionamento.

A função setup() é executada somente uma vez quando a placa de desenvolvimento é ligada ou reiniciada. Geralmente, essa função é utilizada para realizar a configuração inicial do programa, incluindo a definição das configurações iniciais dos pinos, a inicialização de bibliotecas e o estabelecimento da comunicação serial, dentre outras tarefas pertinentes.

Por outro lado, a função loop() é executada de forma contínua após a conclusão da função setup(). É nessa função que a lógica principal do programa é implementada. Todo o código inserido dentro da função loop() será executado repetidamente em um loop infinito, a menos que haja uma instrução específica para interromper a execução.

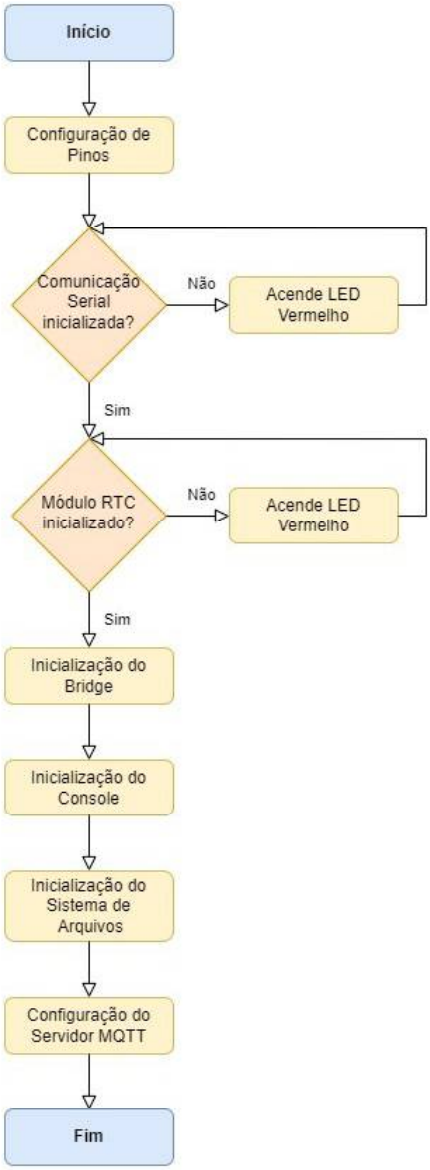


Figura 24 - Fluxograma Setup Coordenador

A biblioteca Bridge é responsável por estabelecer a conexão e comunicação entre o sistema operativo Linux presente na placa Arduino Yun. Permite a interação e partilha de dados entre os dois ambientes do Arduino Yun.

A biblioteca Console é utilizada para iniciar a comunicação série entre a placa e a consola de um computador, permitindo imprimir mensagens na consola durante a execução do programa. Isso facilita o acompanhamento e a depuração do código.

O sistema de ficheiros refere-se à biblioteca FileSystem, responsável pelo gerenciamento dos ficheiros no sistema operativo Linux. Ao utilizar esta função, é possível realizar operações de leitura, escrita, criação e exclusão de ficheiros no ambiente Linux do

Arduino Yun. Isso permite o armazenamento e acesso a dados persistentes no sistema de ficheiros do dispositivo.

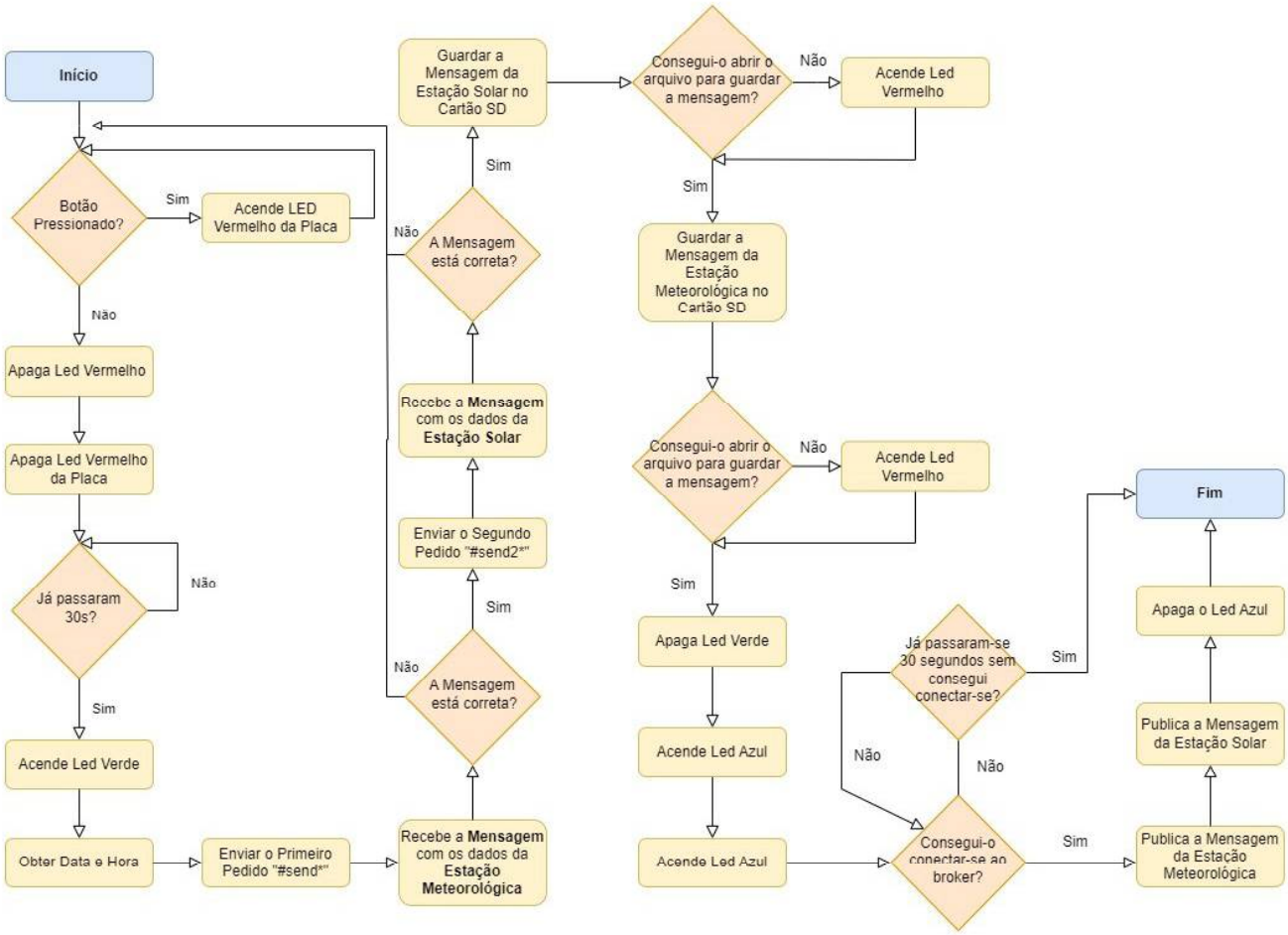


Figura 25 - Fluxograma Loop Coordenador

4.3.2. Módulo RTC

No pseudocódigo representado a seguir, é possível observar como podemos extrair a data, hora e o dia da semana a partir do módulo RTC e colocar os dados no formato desejado:

```
const char* DaysOfTheWeek[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"};

função get_date_and_time(){
```

```
    DateTime now = rtc.now();
    // Obtenção das horas
    Hours = (String(now.hour(), DEC) + ":" + String(now.minute(), DEC) + ":" +
String(now.second(), DEC));
    // Obtenção da data
    Day = (String(now.day(), DEC) + "-" + String(now.month(), DEC) + "-" + String(now.year(),
DEC) + " - " + String(DaysOfTheWeek[now.dayOfTheWeek()]));
    formatted_date = String(now.year(), DEC);
    // Verificações para adicionar zeros à data
    // Verifica se o mês é menor que 10
    se (now.month() < 10) então
      formatted_date += "0"; // Adiciona um zero à string formatada
    fim-se
    formatted_date += String(now.month(), DEC); // Adiciona o mês à string formatada
    // Verifica se o dia é menor que 10
    se (now.day() < 10) então
      formatted_date += "0"; // Adiciona um zero à string formatada
    fim-se
    formatted_date += String(now.day(), DEC); // Adiciona o dia à string formatada
}
```

A variável "Hours" armazena a informação da hora, minutos e segundos no formato HH:MM:SS.

A variável "Day" captura o dia, mês, ano e dia da semana correspondente no formato DD-MM-YYYY-DayOfTheWeek.

Na variável "formatted_date", é necessário verificar se os valores do dia e mês são inferiores a 10. Se forem, adiciona-se um zero antes para garantir a formatação desejada. Ao final desse processo, a variável "formatted_date" é apresentada no formato YYYYMMDD.

4.3.3. Envio dos Pedidos

A Figura 25 apresentada a seguir demonstra como os pedidos de dados são realizados a partir do coordenador da rede. É importante ressaltar que o coordenador funciona em modo *Broadcast*, ou seja, os pedidos são enviados para ambos os *endDevices* ao mesmo tempo.

Conforme mencionado no capítulo 4.2, enviaremos um pedido para que ambos os endDevices recolham os dados simultaneamente. No entanto, apenas a estação Meteorológica enviará os dados no primeiro pedido, a fim de evitar colisão de dados. O *endDevice* da estação de painéis solares aguardará o segundo pedido para enviar os dados.

Antes do envio do segundo pedido, a primeira mensagem será validada para garantir a integridade dos dados e evitar perdas. No próximo subcapítulo, abordaremos esse tema em detalhes e demonstraremos as formas de validação implementadas.

É importante ressaltar que os pedidos são realizados a cada 30 segundos.

Figura 26 - Fluxograma Envio dos Pedidos

4.3.4. Receção e validação das Mensagens

Neste subcapítulo, será descrito como as mensagens são validadas e o que foi feito para tornar isso possível.

As mensagens enviadas pelos endDevices seguem o formato ilustrado na figura 26.

Figura 27 - Formato das Mensagens

Foi adicionado um caractere inicial e final à mensagem como mecanismo de validação que nos permitira receber a mensagem do dispositivo pretendido.

Em seguida, garantimos que os dados não foram alterados comparando o *lenght* na mensagem com o comprimento dos dados recebidos.

Por último, mas não menos importante, verificamos se o *length* não é igual a zero. Caso haja uma falha no envio, os dados podem chegar com tamanho zero, e não podemos aceitar que dados defeituosos sejam armazenados.

4.3.5. Guardar as Mensagens

Neste subcapítulo, é apresentada a função que permite armazenar os dados no cartão SD.

```
função save_info(device_folder, msg2):
    file_path <- string de tamanho 100

    copiar("/mnt/sda1/", file_path) // Copia a string "/mnt/sda1/" para file_path
    concatenar(device_folder, file_path) // Concatena device_folder em file_path
    concatenar("/", file_path) // Concatena "/" em file_path
    concatenar(Day.c_str(), file_path) // Concatena Day convertido em uma string em file_path
    concatenar(".txt", file_path) // Concatena ".txt" em file_path

    data <- FileSystem.open(file_path, FILE_APPEND) // Abre o arquivo para escrita
    se data != null então
      // Arquivo aberto com sucesso
      data.imprimir(Hours + ",") // Escreve a mensagem no arquivo
      data.flush()
      data.imprimir(msg2)
      data.flush()
      data.imprimirLinha("")
      data.flush() // Garante que os dados sejam gravados no arquivo
      data.fechar() // Fecha o arquivo
    senão
      // Erro ao abrir o arquivo
      digitalWrite(ledBicVermelho, HIGH) // Acende o LED vermelho
    fim se
fim função
```

O pseudocódigo em questão demonstra como podemos separar os dados em pastas e, dentro delas, realizar a separação por dia. Essa abordagem é útil para organizar e categorizar informações de maneira mais estruturada.

No anexo A2, é possível verificar o resultado final.

4.3.6. Conectar a rede Wi-Fi

Para realizar o envio dos dados por MQTT, é necessário conectar o Arduino Yun a uma rede Wi-Fi. Esse procedimento pode ser realizado através de uma porta Ethernet ou de uma conexão Wi-Fi.

No caso do Arduino Yun, a configuração da conexão Wi-Fi difere das demais placas. É preciso aceder à rede criada pelo Arduino e aceder ao endereço http://arduino.local ou ao endereço IP (192.168.240.1) para estabelecer a conexão.

A Figura 27 apresenta a interface do endereço arduino.local.

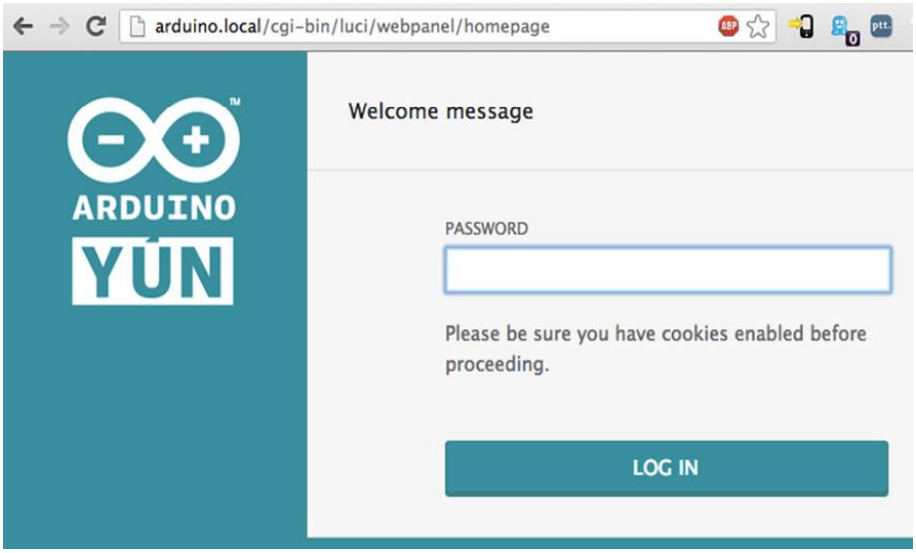


Figura 28 - Interface arduino.local

Fonte: https://labdegaragem.com/m/blogpost?id=6223006%3ABlogPost%3A418919

Ao aceder ao site, será solicitada uma palavra-passe. Por padrão, a palavra-passe é "arduino".

Ao entrar no site, serão exibidas informações relacionadas à rede criada pelo Arduino, como o endereço, máscara, endereço MAC e dados referentes à largura de banda. Além disso, é possível verificar se existe alguma ligação Ethernet.

A Figura 28 demonstra a interface inicial do site.

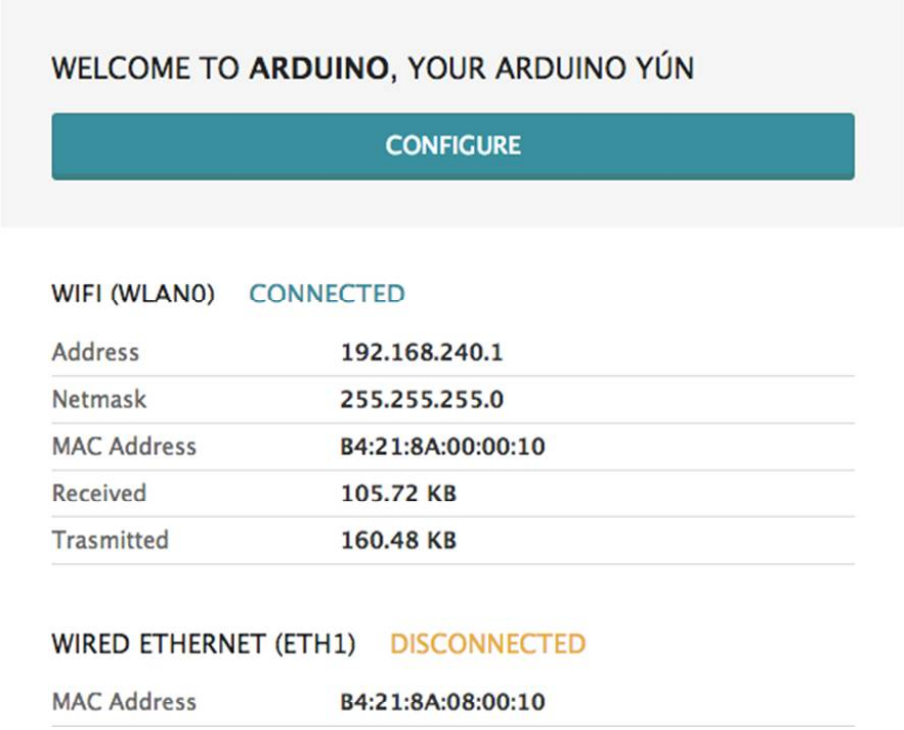


Figura 29 - Interface inicial do site

Fonte: https://labdegaragem.com/m/blogpost?id=6223006%3ABlogPost%3A418919

Ao selecionar a opção "Configurar", seremos direcionados para outra página, representada na Figura 29, onde encontraremos as configurações relacionadas à placa e à rede.

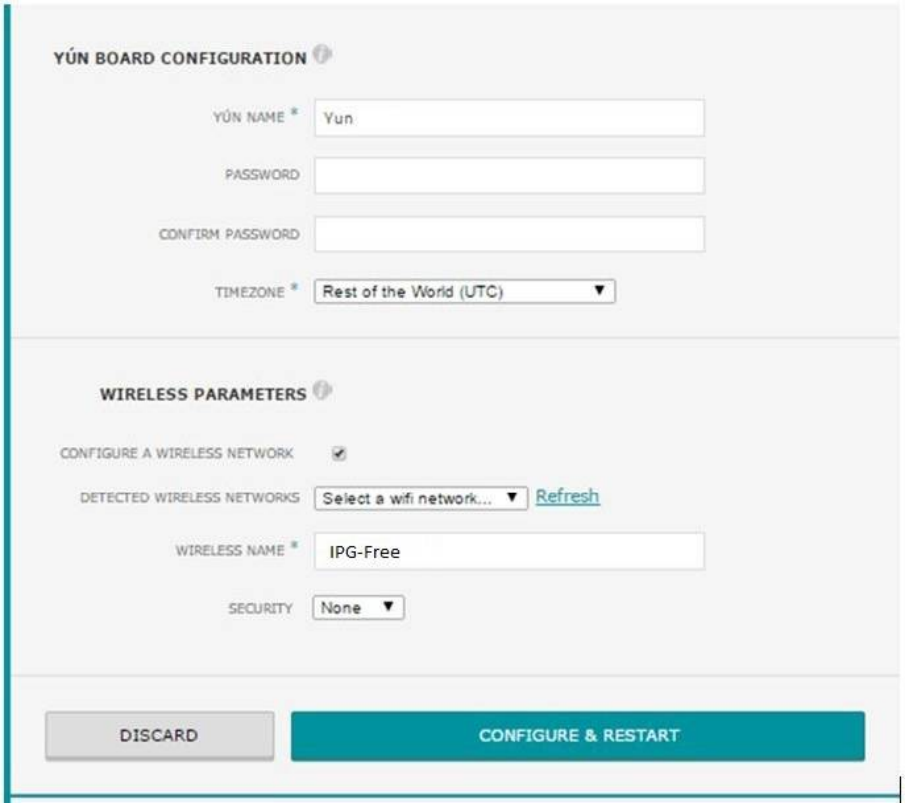


Figura 30 - Interface de configuração

Na configuração da placa, é possível alterar o nome da rede e a senha, além de ajustar o horário de acordo com o fuso horário desejado.

Assim, para configurar a rede Wi-Fi, basta clicar em "Refresh" para que todas as conexões disponíveis sejam listadas. Em seguida, seleciona-se a opção de segurança da rede, como WPA, WPA2, ou deixar em branco caso não possua segurança. Caso haja segurança, será necessário inserir a palavra-passe de acesso à rede. Por fim, basta clicar em "Configurar e Reiniciar" para que a conexão com a rede desejada seja estabelecida.

4.3.7. Conectar ao Broker e Publicar as Mensagens

Neste subcapítulo, são abordados dois aspectos cruciais: a conexão com o *broker* e a publicação da mensagem. O fluxograma que segue, representado na Figura 30, demonstra como foi implementado.

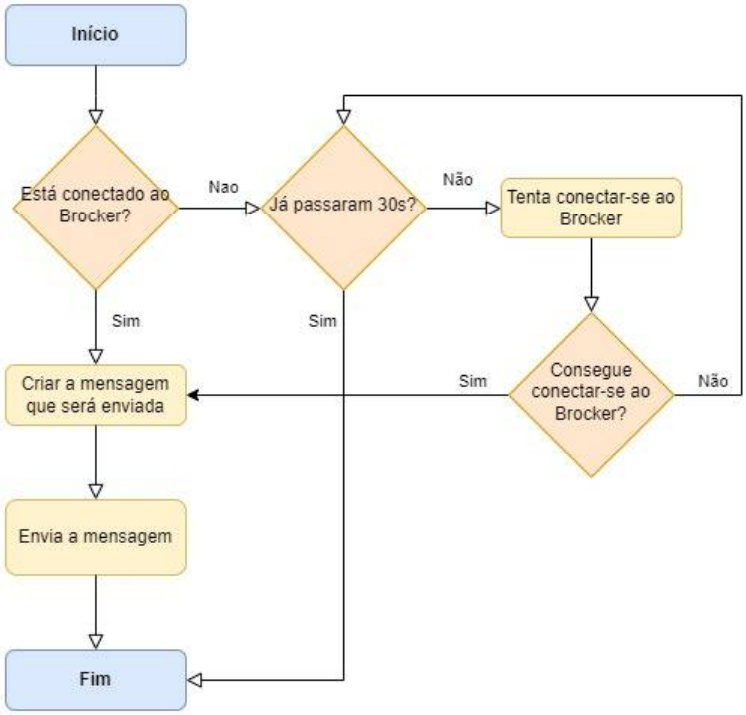


Figura 31 - Fluxograma da publicação da mensagem

A estrutura da mensagem pode ser vista na figura 31.

Figura 32 - Estrutura da mensagem que será publicada

Para que a mensagem fique neste formato, é necessário formatar a data e a hora como uma *string* de caracteres. Em seguida, basta concatenar os dados num único *array* de caracteres e publicar a mensagem, fornecendo o tópico ao qual será enviada.

No fluxograma, a reconexão ao *broker* em caso de falha na conexão tem uma duração de apenas 30 segundos. Isso evita que o programa fique bloqueado num loop contínuo de tentativas e permite que novos pedidos sejam registados no cartão SD.

4.4.Aplicação do EndDevice Estação Meteorológica

Neste subcapítulo, será apresentada a configuração e aplicação do EndDevice referente à estação meteorológica. Para esse fim, foram criados subcapítulos nos quais é destacado o fluxograma das funções básicas do código para Arduino, tanto do setup quanto do Loop, com o objetivo de facilitar a compreensão do funcionamento do código desenvolvido. Em seguida, serão abordados temas como a receção e tratamento dos pedidos por parte do coordenador, a recolha e processamento dos dados e o seu envio de volta ao coordenador.

4.4.1. Fluxograma

A Figura 32 e 33 apresentam o fluxograma do setup e do loop, visando facilitar a compreensão de seu funcionamento.

Figura 33 - Fluxograma Setup End Device - Estação Meteorológica

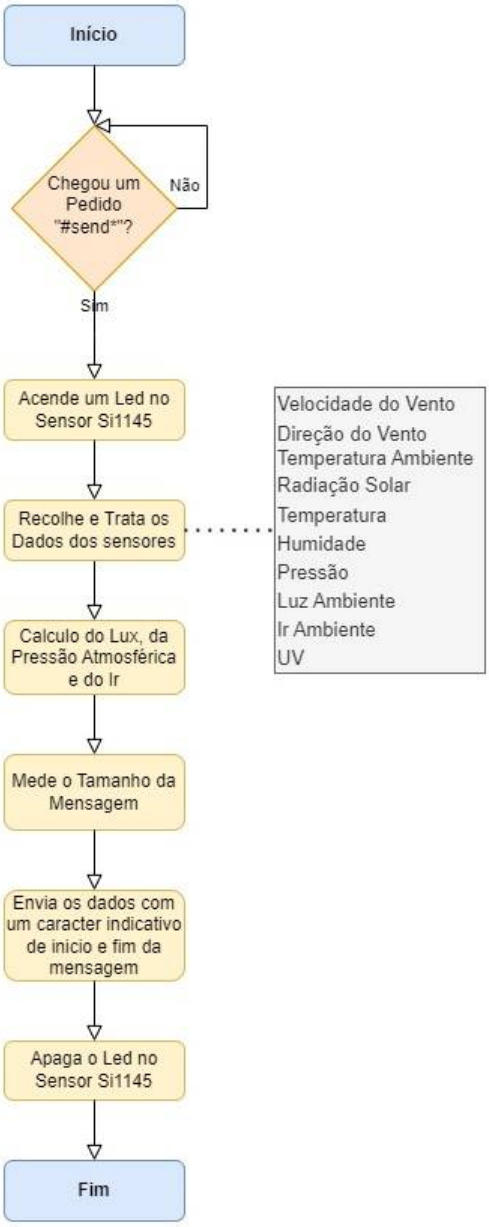


Figura 34 - Fluxograma Loop End Device - Estação Meteorológica

A medição da mensagem, como referido no capítulo 4.3.4, tem como objetivo validar junto ao Coordenador que não ocorreu perda de dados durante o envio. Essa verificação é importante para garantir a integridade e a fiabilidade das informações transmitidas. Ao realizar a medição da mensagem, é possível assegurar que os dados chegaram ao seu destino corretamente, sem qualquer perda ou corrupção ao longo do processo. Isso proporciona uma maior segurança e precisão no sistema de comunicação utilizado.

4.4.2. Receção de Pedido de Envio

Uma abordagem comum para verificar se algum pedido foi enviado para recolher e enviar os dados é a seguinte:

```
// Enquanto houver dados disponíveis na porta Serial
enquanto Serial.available() > 0 faça:
// Lê um caractere da porta Serial
   se Serial.read() == '#' então
   // Se o caractere lido for '#'
      msg_received = Serial.readStringUntil('*')
      // Lê a mensagem até encontrar o caractere '*' e armazena em msg_received
            se msg  received == "send" então
            // Se a mensagem recebida for igual a "send"
            …
```

O pseudocódigo acima verifica se a porta serial recebeu a mensagem "#send*" enviada pelo Coordenador.

4.4.3. Recolha e Tratamento dos Dados

Neste subcapítulo, apresentaremos as fórmulas e as instruções utilizadas para recolher os valores dos sensores e convertê-los em valores reais. Além disso, serão calculados os valores de infravermelho (IR), lux e pressão em bar.

Velocidade do Vento:

Realizou-se a leitura analógica do pino utilizando a função analogRead e, em seguida, foi aplicada a seguinte fórmula para obter um valor representativo da velocidade do vento:

$$y = x \times \frac{5.0}{1023.0} \times \frac{30.0}{4.80}$$

Direção do Vento:

Realizou-se a leitura analógica do pino utilizando a função analogRead e, em seguida, foi aplicada a seguinte fórmula para obter um valor representativo da direção do vento:

$$y = x \times \frac{5.0}{1023.0} \times \frac{360.0}{5.0}$$

Temperatura Zonen (Termistor):

Realizou-se a leitura analógica do pino utilizando a função analogRead e, em seguida, foi aplicada a seguinte fórmula para obter um valor representativo da temperatura do termistor:

$$k = x \times \frac{5.0}{1023.0}$$

$$y = k^6 \times 0.2715 - k^5 \times 4.1324 + k^4 \times 23.641 - k^3 \times 61.447 + k^2 \times 66.787 + k \times 3.4292 - 52.64$$

Radiação Solar:

Realizou-se a leitura analógica do pino utilizando a função analogRead e, em seguida, foi aplicada a seguinte fórmula para obter um valor representativo da Radiação Solar:

$$k = x \times \frac{5.0}{1023.0} \times \frac{1250.0}{4.8}$$

Temperatura Sensor Sparkfun:

Para obter a leitura da temperatura, utilizou-se a seguinte instrução:

```
float temperature = si7021.readTemperature();
```

Humidade Sensor Sparkfun:

Para obter a leitura da humidade, utilizou-se a seguinte instrução:

```
float humidity = si7021.readHumidity();
```

Pressão:

Para obter a leitura da pressão, utilizou-se a seguinte instrução:

```
float pressure = mpl3115a2.readPressure();
```

Para converter de Pascal para Bar, utilizou-se a seguinte fórmula:

$$y = pressure \ / \ 101325.0$$

Luz Ambiente:

Para obter a leitura da luz ambiente, utilizou-se a seguinte instrução:

```
uint32_t ambient_light = si1145.getAlsVisData();
```

IR:

Para obter a leitura do IR ambiente no sensor Si1145, utilize a seguinte instrução:

```
uint8_t amb_ir = si1145.getAlsIrData();
```

Para convertero IR ambiente para IR, utilize a fórmula:

$$y = \text{amb\_ir} \times 0.01$$

UV:

Para obter a leitura do UV no sensor Si1145, utilize a seguinte instrução:

```
uint16_t uv = si1145.getUvIndex();
```

LUX:

Para calcular o valor do Lux, utilizou-se a equação indicada na ficha de aplicação AN523 da Silabs, dedicada a esse propósito.

```
uint16_t uv = si1145.getUvIndex();

float calcLux(float vis, float ir) {
  const uint16_t vis_dark = 256; // valor empírico
  const uint8_t ir_dark = 250; // valor empírico
  const float gainFactor = 1.0;
  const float visCoeff = 5.41; // notas de aplicação AN523
  const float irCoeff = 0.08; // notas de aplicação AN523
```

```
    const float corrFactor = 1.25; // fator de correção empírico

    float lx = ((vis - vis_dark) * visCoeff - (ir - ir_dark) * irCoeff) * gainFactor *
corrFactor;
    return lx;
}

float ambient_light = analogRead(pino_ambient_light);
float amb_ir = si1145.getAlsIrData();
float lux = calcLux(ambient_light, amb_ir);
```

4.4.4. Envio da Mensagem

Após a recolha dos dados, será criada a mensagem final para enviar os dados ao coordenador. A estrutura da mensagem pode ser visualizada na Figura 34. Os dados são separados por vírgulas dentro da mensagem, o que facilita a identificação e organização das informações ao serem transmitidas.

Caracter Inicial	Data	Hora	Velocidade do Vento	Direção do Vento	Temp Zonen	Radiação Solar	Temp Adafruit	Humidade	Pressão	Luz ambiente	IR	UV	LUX	Length	Caracter Final

Figura 35 - Estrutura da Mensagem da EM

Conforme está explicado no capítulo 4.3.4, o uso do caractere inicial e final na mensagem tem como propósito identificar a mensagem e protegê-la contra possíveis interferências de outros dispositivos na rede.

4.5. Aplicação do EndDevice Estação de Painéis Solares

Neste subcapítulo, serão apresentadas a configuração e a aplicação do EndDevice relacionado à estação de painéis solares. Com esse propósito, foram criados subcapítulos nos quais se destaca o fluxograma das funções básicas do código para Arduino, tanto na configuração quanto no loop, a fim de facilitar a compreensão do funcionamento do código desenvolvido. Em seguida, serão abordados tópicos como a receção do pedido enviado pelo coordenador, recolha e tratamento dos dados e envio dos dados para o coordenador da rede.

4.5.1. Fluxograma

A Figura 35 e 36 apresentam o fluxograma do setup e do loop, visando facilitar a compreensão de seu funcionamento.

Figura 36 - Fluxograma Setup End Device - Estação Solar

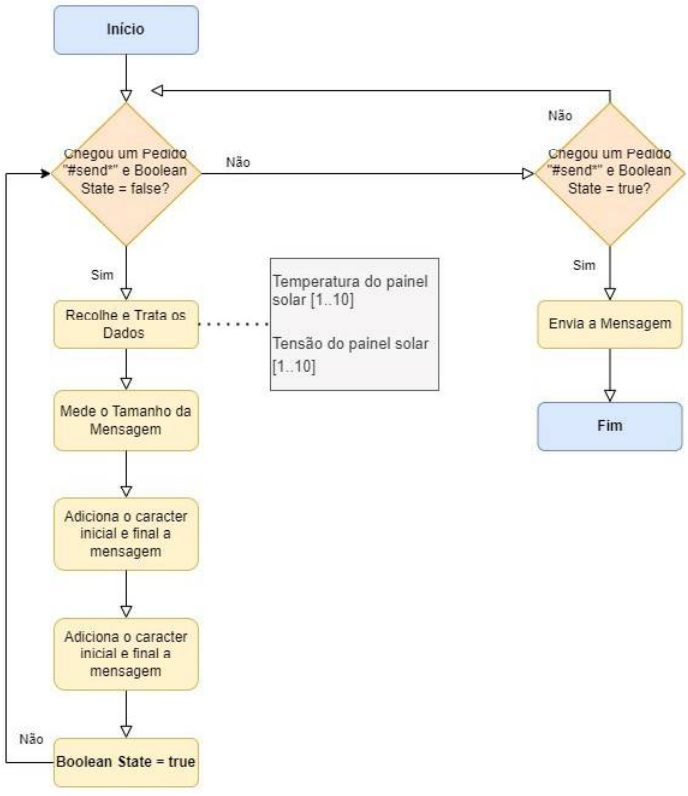


Figura 37 - Fluxograma Loop End Device - Estação Solar

4.5.2. Receção do Pedido de Envio

Conforme mencionado no capítulo 3.4.4, a estação de Painéis Solares só envia a mensagem com os dados quando recebe o segundo pedido. O pseudocódigo abaixo demonstra como essa sequência de pedidos é processada:

```
Enquanto (Serial.available() > 0) {
  // Verifica se o caractere recebido é '#'
  Se (Serial.read() == '#') {
    // Lê a mensagem recebida até encontrar o caractere '*'
    msg_received = Serial.readStringUntil('*');
    // Verifica se a mensagem recebida é "send" e o estado atual é falso (state == false)
    Se (msg_received == "send" && !state) {
      // Executa a função data_collected() para recolher os dados
      data_collected();
      // Executa a função treat_data() para tratar os dados coletados
      treat_data();
      // Limpa a mensagem recebida
      msg_received = "";
      // Define o estado como verdadeiro (state == true)
      state = verdadeiro;
    }

    // Senão, se a mensagem recebida é "send2" e o estado atual é verdadeiro (state == true)
    Senão se (msg_received == "send2" && state) {
      // Executa a função send_data() para enviar os dados tratados
      send_data();
      // Limpa a mensagem recebida
      msg_received = "";
      // Limpa a mensagem final
      msg_final = "";
      // Define o estado como falso (state == false)
      state = falso;
    }
    // Senão, a mensagem não corresponde a "send" nem "send2", então retorna
    Senão {
      Retornar;
    }
  }
}
```

4.5.3. Recolha e Tratamento dos Dados

Neste subcapítulo, apresentaremos as fórmulas e as instruções utilizadas para recolher os valores dos sensores e convertê-los em valores reais.

Temperatura dos Painéis:

Para medir a temperatura dos painéis, realizou-se a leitura analógica do pino utilizando a função analogRead. Posteriormente, aplicou-se a seguinte fórmula para obter um valor representativo da temperatura em cada painel:

$$k = x \times \frac{5.0}{1023.0}$$

$$y = 1.3225 \times k^2 + 18.566 \times k - 27.41$$

Tensão dos Painéis:

Efetuou-se a leitura das tensões através dos multiplexadores, conforme referido no capítulo 4.1. Neste caso, utilizamos dois multiplexadores de 8 para 1 e decidimos dividir as 10 leituras de temperaturas dos painéis solares igualmente entre os dois mux. À medida que obtemos o valor convertido na porta analógica, aplicamos a seguinte fórmula para obter um valor representativo da tensão em cada painel:

$$y = \frac{x * 50}{4.545}$$

4.5.4. Envio da Mensagem

Após a recolha dos dados, será criada a mensagem final para enviar os dados ao coordenador. A estrutura da mensagem pode ser visualizada na Figura 37. Os dados são separados por vírgulas dentro da mensagem, o que facilita a identificação e organização das informações ao serem transmitidas.

Caracter Inicial	Data	Hora	Temp Painel1	Temp Painel2	Temp Painel3	Temp Painel4	Temp Painel5	Temp Painel6	Temp Painel7	Temp Painel8	Temp Painel9	Temp Painel10
Tens Painel1	Tens Painel2	Tens Painel3	Tens Painel4	Tens Painel5	Tens Painel6	Tens Painel7	Tens Painel8	Tens Painel9	Tens Painel10	Length	Caracter Final	

Figura 38 - Estrutura da Mensagem da ES

4.6.Website

Neste subcapítulo, apresentaremos o servidor utilizado para executar a base de dados MySQL. Posteriormente, no Node-Red, explicaremos como é possível recolher os dados publicados por MQTT e formatá-los para inserção na base de dados. Em seguida, veremos como podemos visualizar os últimos dados recebidos, selecionar uma data ou intervalo de datas específicas, e exibir os dados correspondentes a esse período através de tabelas ou gráficos.

Além disso, descreveremos um mecanismo de segurança implementado, que notificará os responsáveis por e-mail caso a temperatura dos painéis esteja acima dos valores recomendados.

Por fim, forneceremos um exemplo de como fazer o *download* dos dados referentes a uma data selecionada, disponibilizando-os em formato txt para que seja mais fácil compartilhar os dados.

4.6.1. XAMPP

O servidor utilizado na aplicação foi o XAMPP, uma solução de desenvolvimento web amplamente adotada. O XAMPP é uma distribuição fácil de instalar e usar, que oferece os seguintes componentes principais:

Apache: O XAMPP inclui o servidor web Apache, reconhecido como um dos servidores web mais populares do mundo. O Apache é conhecido pela sua estabilidade, segurança e flexibilidade, tornando-o adequado para atender a diversas necessidades de hospedagem web.

MySQL: O XAMPP também incorpora a base de dados MySQL, um sistema de gestão de base de dados confiável e amplamente utilizado. O MySQL possibilita o armazenamento, gestão e acesso eficiente a dados, sendo ideal para aplicativos web que requerem a persistência e recuperação de informações.

Desenvolvido principalmente em C++ e fazendo uso de linguagens como Perl e PHP, o XAMPP fornece uma solução completa e poderosa para criar e implantar aplicativos web de forma eficiente. O XAMPP simplifica o processo de configuração do ambiente de desenvolvimento, permitindo que os desenvolvedores foquem em criar aplicações web robustas e escaláveis.

A Figura 38 mostra a interface do XAMPP.

Figura 39 - Interface do XAMPP

4.6.1.1. Guardar os dados publicados na base de dados

Para armazenar os dados publicados, é necessário, primeiramente, configurar o *broke*r MQTT para receber os dados do coordenador. Essa configuração envolve especificar o servidor utilizado no coordenador e o tópico correspondente. Após a configuração adequada, os dados recebidos podem ser separados utilizando a vírgula como delimitador e inseridos num *array*. Em seguida, é possível identificar cada dado no *array* e realizar uma operação de inserção "INSERT INTO" na tabela correspondente. Na Figura 39, encontra-se o grupo de nós no Node RED responsável por essa tarefa, que se dedica à recolha dos dados da estação meteorológica.

Figura 40 - NodeRed - Guardar os Dados publicados na Base de Dados

Código:

Split Msg:

```
var string = msg.payload;
var array = string.split(',');
msg.payload = array;
return msg;
```

Data to be Entered:

```
var array = msg.payload;

var tag = array[0] //Date
var tag0 = array[1] // Hour
var tag1 = parseInt(array[2]) // Wind_Speed
var tag2 = array[3] // Wind_Direction
var tag3 = parseFloat(array[4]) // Temp Zonen
var tag4 = parseInt(array[5]) // Sun
var tag5 = parseFloat(array[6]) // Temperature
var tag6 = parseFloat(array[7]) // Humidity
var tag7 = parseFloat(array[8]) // Pressure
var tag8 = parseInt(array[9]) // Ambient_Light
var tag9 = parseInt(array[10]) // IR
var tag10 = parseInt(array[11]) // UV
var tag11 = parseFloat(array[12]) // LUX

msg.payload =
[tag,tag0, tag1, tag2, tag3, tag4,
 tag5, tag6, tag7, tag8, tag9,
 tag10, tag11]

msg.topic = 'INSERT INTO mega_meteorological_data (Date,Hour,
Wind_Speed, Wind_Direction, Room_Temperature, Sun, Temperature,
Humidity, Pressure, Ambient_Light ,IR ,UV ,LUX)
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?);';

return msg;
```

4.6.1.2. Últimos dados Recebidos

Uma das páginas do website exibe os últimos dados publicados. Para isso, realizamos uma consulta (SELECT) na tabela, buscando todos os dados disponíveis. Em seguida, extraímos o primeiro elemento da coluna correspondente aos dados que desejamos exibir. Repetimos esse processo para os demais dados que queremos mostrar. Na Figura 40 e 41 encontram-se os grupos de nós no Node-RED responsáveis por essa tarefa, um para a estação de painéis solares e outro para a estação meteorológica.

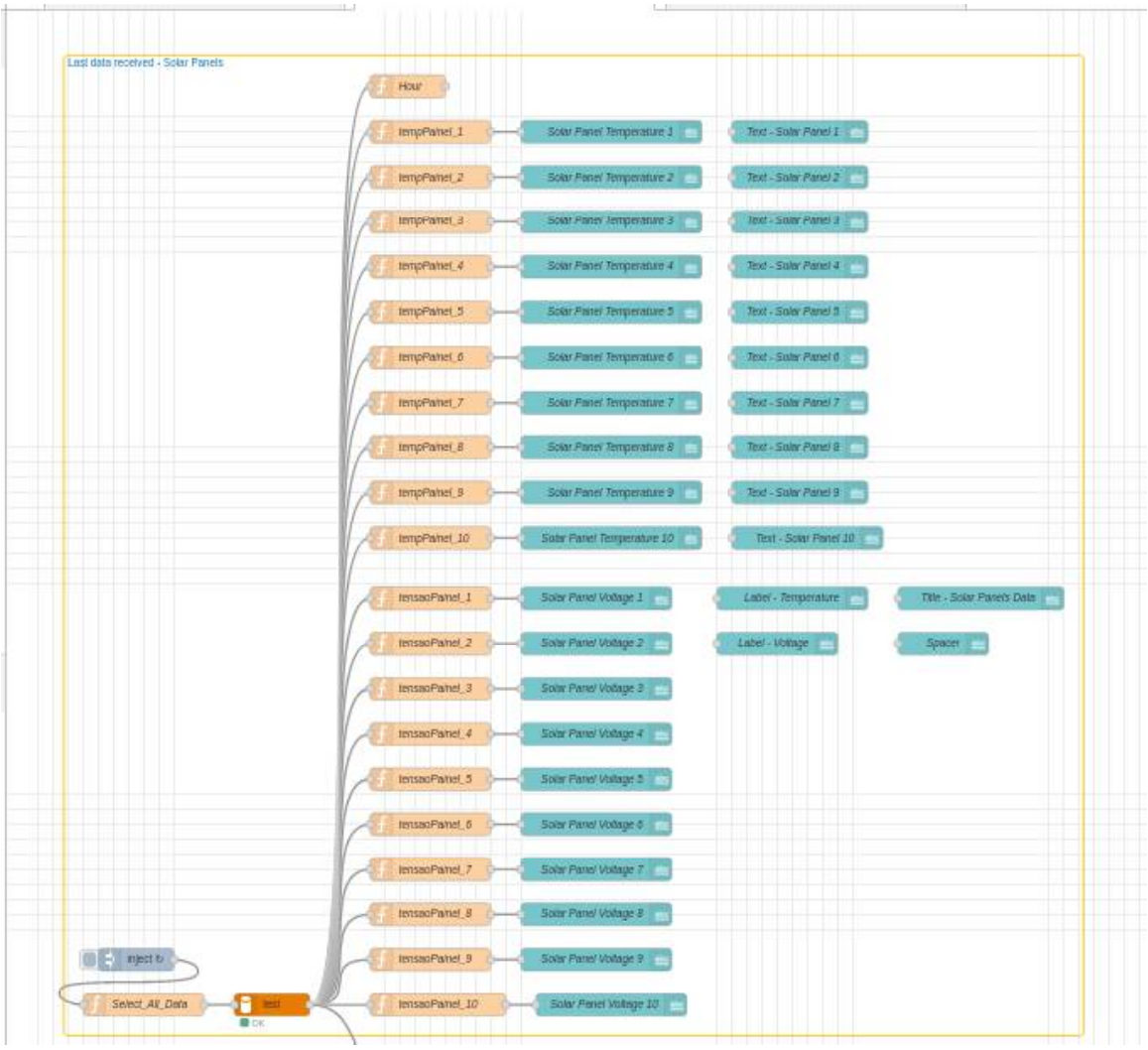


Figura 41 - Últimos dados Recebidos Estação de Painéis Solares

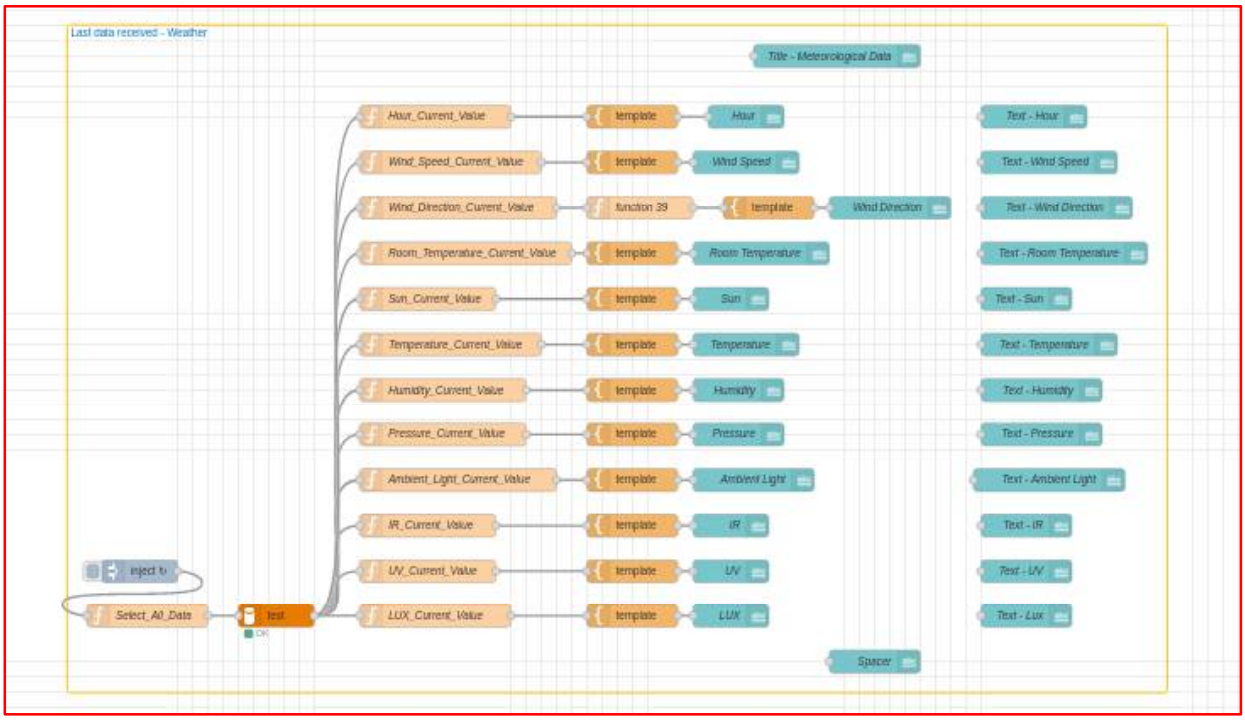


Figura 42 - Últimos dados Recebidos Estação Meteorológica

Código:

Select_All_Data:

```
msg.topic = "SELECT * FROM mega_meteorological_data ORDER BY Id DESC"
```

Wind_Speed_Current_Value:

```
var first_Element = msg.payload[0];
var first_Wind_Speed =  first_Element.Wind_Speed;
msg.payload = first_Wind_Speed;
return msg;
```

Na Figura 42, é possível ver a interface que mostra os últimos dados recebidos, separados por estações.

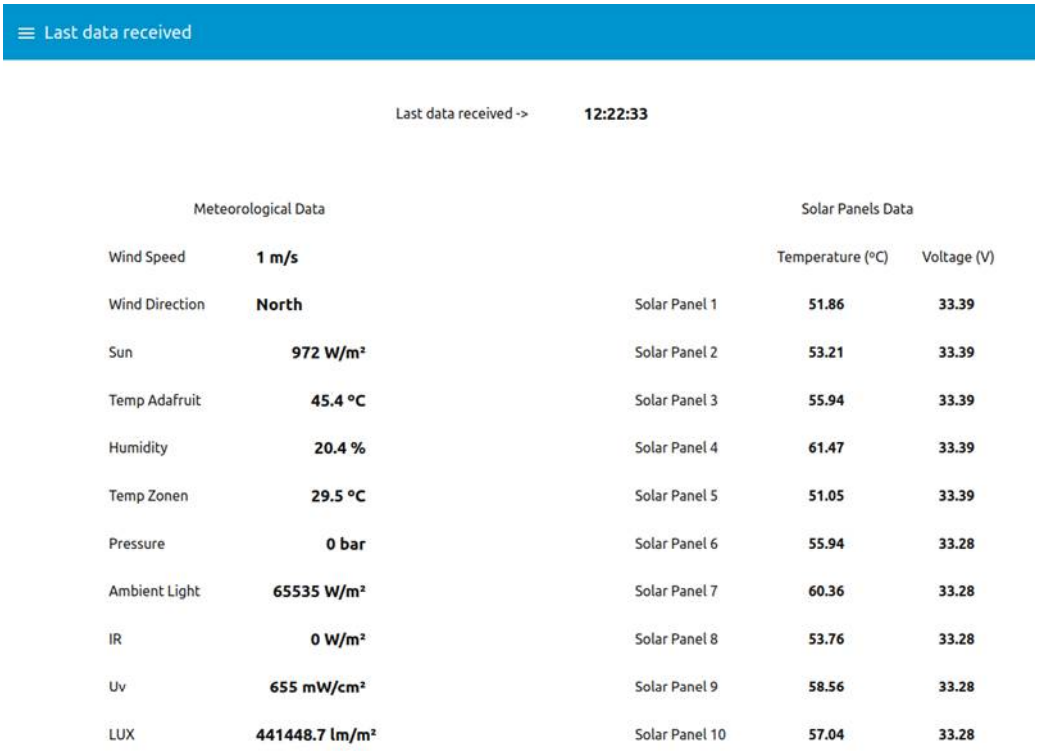


Figura 43 - Interface Últimos dados Recebidos

4.6.1.3. Tabela com os dados

Neste capítulo, demonstrámos como é possível, através da seleção de uma data, mostrar os dados referentes a esse dia numa tabela. O fluxograma representado na Figura 43 demonstra como isso é possível.

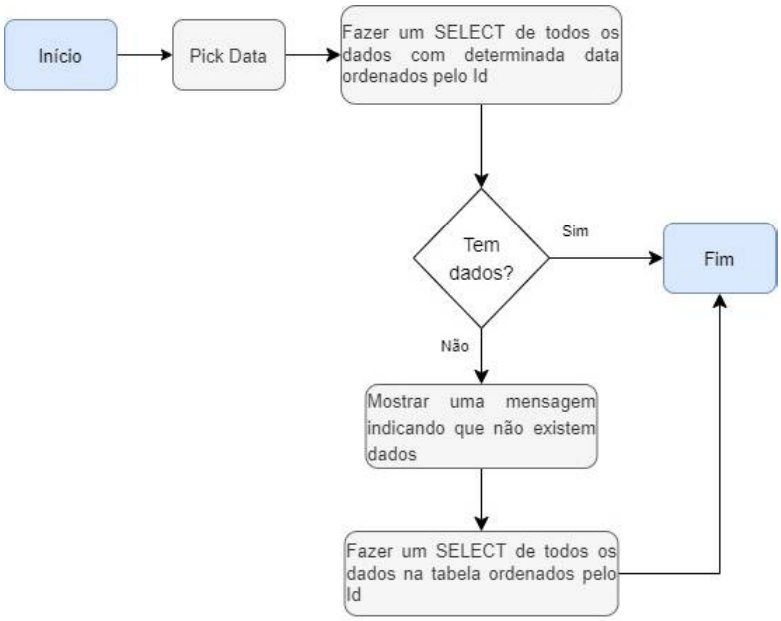


Figura 44 - Fluxograma da Tabela

Na Figura 44 e 45 encontram-se os grupos de nós no Node RED responsáveis por essa tarefa, um para a estação de painéis solares e outro para a estação meteorológica.

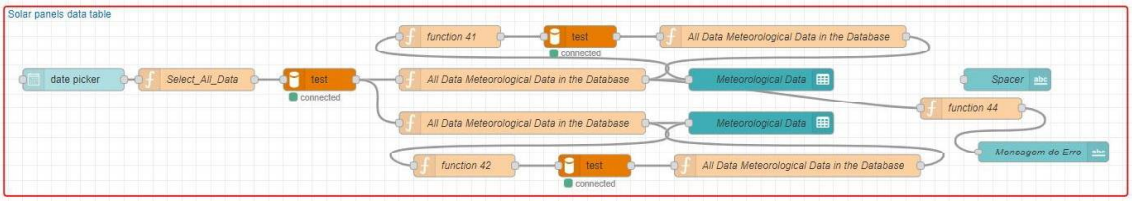


Figura 45 - NodeRed Tabela com os dados da Estação de Paineis Solares

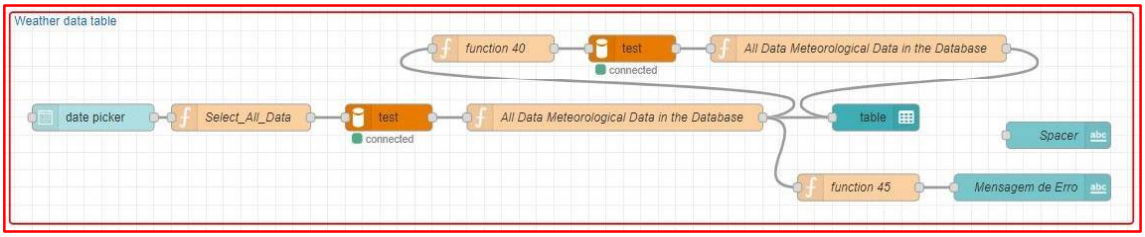


Figura 46 - NodeRed Tabela com os dados da Estação Meteorológica

Código:

Select_All_Data:

```
var data = new Date(msg.payload);
data.setMinutes(data.getMinutes() - data.getTimezoneOffset());

var dataFormatada = data.toISOString().slice(0, 10).replace(/-/g, '');

function extrairData(dataFormatada) {
    const dataFormatada2 = dataFormatada.slice(0, 10);
    return dataFormatada2;
}

var data2 = extrairData(dataFormatada);

msg.topic = "SELECT * FROM mega_meteorological_data WHERE Date =
'" + data2 + "' ORDER BY Hour DESC;";

return msg;
```

All Data Meteorological Data in the Database:

```
var newDataArray = msg.payload.map(item => {
    var date = item.Date;
    var formattedDate = `${date.substr(0, 4)}-${date.substr(4, 2)}-
${date.substr(6, 2)}`;

    return {
        Date: formattedDate,
        Hour: item.Hour,
        W_Speed: item.Wind_Speed,
        W_Direction: item.Wind_Direction,
        Temp_zonen: item.Room_Temperature,
        Sun: item.Sun,
        Temp_adafruit: item.Temperature,
        Hum: item.Humidity,
        Pres: item.Pressure,
        Amb_Light: item.Ambient_Light,
        IR: item.IR,
        UV: item.UV,
```

Na função "Função 45", é verificado se está a ser passado algum dado. Caso contrário, não existem dados referentes àquele dia, e é apresentada uma mensagem no *dashboard*.

A mensagem de erro apresentada pode ser vista na Figura 46.

Function 45:

```
if (msg.payload[0] == null) {
    msg.payload = "Não foram econtrados dados no período específico.";

} else {
    msg.payload = " ";
}
return msg;
```

Figura 47 - Tabela Mensagem de Erro

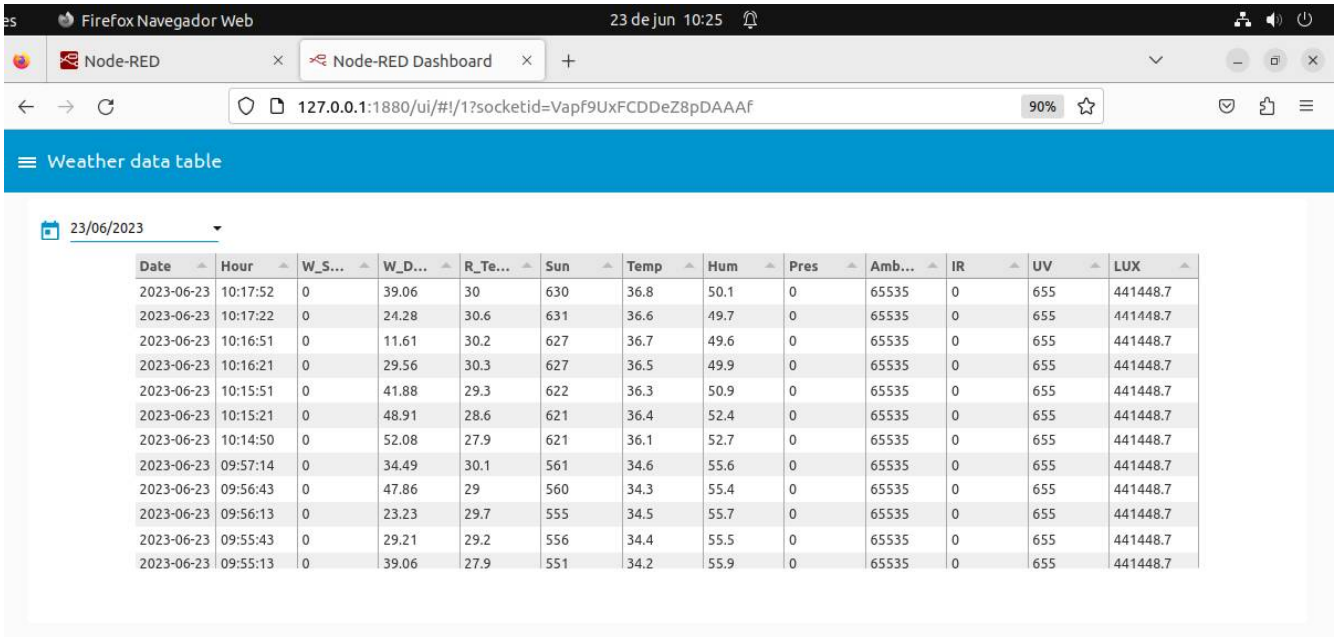
Se não estiverem sendo enviados dados ao nó da tabela, ou seja, se não houver informações no dia em que foi pesquisado, o programa irá retornar todos os dados na base de dados referentes à estação específica e ordená-los por ordem decrescente, do primeiro para o último dado recebido. A Função 40 exemplifica exatamente esse processo.

Function 40:

```
if(msg.payload[0] == null){
    msg.topic = "SELECT * FROM mega_meteorological_data ORDER BY
Id DESC";

}
return msg;
```

A título de exemplo, a Figura 47 demonstra a seleção da data 23/06/2023 e a amostragem dos dados referentes a esse dia na tabela.



Date	Hour	W_S...	W_D...	R_Te...	Sun	Temp	Hum	Pres	Amb...	IR	UV	LUX
2023-06-23	10:17:52	0	39.06	30	630	36.8	50.1	0	65535	0	655	441448.7
2023-06-23	10:17:22	0	24.28	30.6	631	36.6	49.7	0	65535	0	655	441448.7
2023-06-23	10:16:51	0	11.61	30.2	627	36.7	49.6	0	65535	0	655	441448.7
2023-06-23	10:16:21	0	29.56	30.3	627	36.5	49.9	0	65535	0	655	441448.7
2023-06-23	10:15:51	0	41.88	29.3	622	36.3	50.9	0	65535	0	655	441448.7
2023-06-23	10:15:21	0	48.91	28.6	621	36.4	52.4	0	65535	0	655	441448.7
2023-06-23	10:14:50	0	52.08	27.9	621	36.1	52.7	0	65535	0	655	441448.7
2023-06-23	09:57:14	0	34.49	30.1	561	34.6	55.6	0	65535	0	655	441448.7
2023-06-23	09:56:43	0	47.86	29	560	34.3	55.4	0	65535	0	655	441448.7
2023-06-23	09:56:13	0	23.23	29.7	555	34.5	55.7	0	65535	0	655	441448.7
2023-06-23	09:55:43	0	29.21	29.2	556	34.4	55.5	0	65535	0	655	441448.7
2023-06-23	09:55:13	0	39.06	27.9	551	34.2	55.9	0	65535	0	655	441448.7

Figura 48 - Tabela com os dados da Estação Meteorológica

4.6.1.4. Gráficos

Neste capítulo, demonstrámos como é possível, através da seleção de duas datas, mostrar os dados referentes a esse intervalo de datas num gráfico. O fluxograma representado na Figura 48 demonstra como isso é possível.

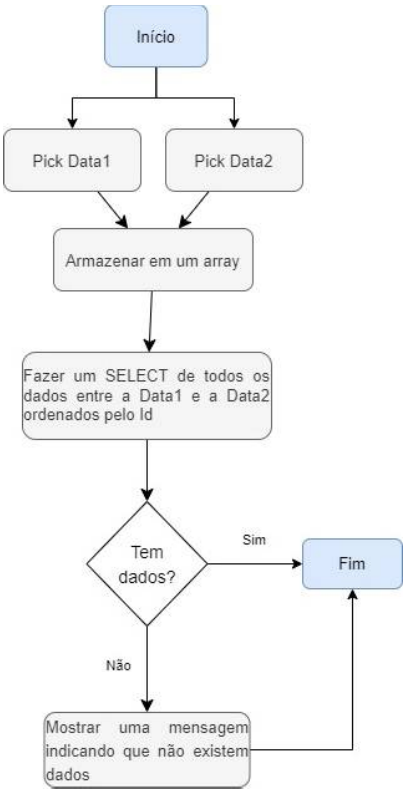


Figura 49 - Fluxograma do Gráfico

Na Figura 49 e 50 encontram-se os grupos de nós no Node RED responsáveis por essa tarefa, um para a estação de painéis solares e outro para a estação meteorológica.

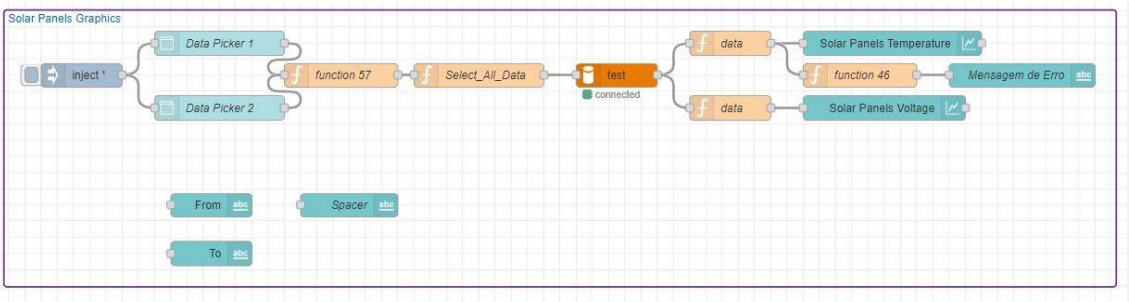


Figura 50 - NodeRed Gráficos Estação de Painéis Solares

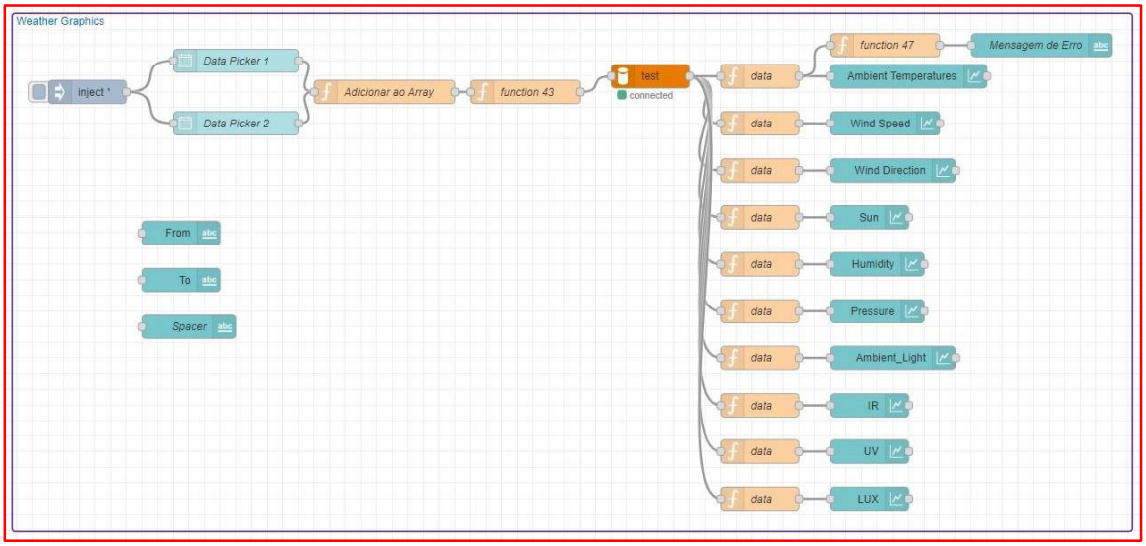


Figura 51 - NodeRed Gráficos Estação Meteorológica

Código:

Adicionar ao Array:

```
var arrayValores = context.get('arrayValores') || [];

if (msg.topic === 'data1' || msg.topic === 'data2') {
    // Verifica se o valor já existe no array
    var index = arrayValores.findIndex(function (date) {
        return date.topic === msg.topic;
    });

    if (index === -1) {
        // Adiciona o valor ao array
        arrayValores.push({ topic: msg.topic, payload: msg.payload });
    } else {
        // Atualiza o valor no array
        arrayValores[index] = { topic: msg.topic, payload: msg.payload };
    }
}
```

```
// Limita o array a duas posições
arrayValores = arrayValores.slice(0, 2);

// Atualiza o contexto com o novo array
context.set('arrayValores', arrayValores);

if (arrayValores.length === 2) {
    var arrayValoresFormatados = arrayValores.map(function (item) {
        var data = new Date(item.payload);
        data.setMinutes(data.getMinutes() - data.getTimezoneOffset());
        return data.toISOString();
    });

    msg.payload = arrayValoresFormatados;
    return msg;
} else {
    return null;
}
```

Function 43:

```
var data = new Date(msg.payload[0]);
data.setMinutes(data.getMinutes() - data.getTimezoneOffset());
var dataFormatada = data.toISOString().slice(0, 10).replace(/-/g, '');

var data2 = new Date(msg.payload[1]);
data2.setMinutes(data2.getMinutes() - data2.getTimezoneOffset());
var dataFormatada2 = data2.toISOString().slice(0, 10).replace(/-/g, '');

msg.topic = "SELECT * FROM mega_meteorological_data WHERE Date >= '" +
dataFormatada + "' AND Date <= '" + dataFormatada2 + "'ORDER BY Id DESC";
return msg;
```

Data:

```
var newDataArray = msg.payload.map(item => {
    var date = item.Date;
    var formattedDate = `${date.substr(6, 2)}-${date.substr(4, 2)}`;

    return {
        Date: formattedDate,
        Hour: item.Hour,
        Temp_adafruit: item.Temperature,
        Temp_zonen: item.Room_Temperature
    };
});

var m = {};
m.labels = newDataArray.map(item => `${item.Hour}`);
//m.labels = newDataArray.map(item => `${item.Date} ${item.Hour}`);
m.series = ['Temp_adafruit', 'Temp_zonen'];
m.data = [
    newDataArray.map(item => item.Temp_adafruit),
    newDataArray.map(item => item.Temp_zonen)
];

msg.payload = [m];
msg.topic = newDataArray;
return msg;
```

Na função 'Function 47', verifica-se se está sendo passado algum dado. Caso contrário, não existem dados no intervalo de datas selecionado, e é apresentada uma mensagem de erro no *dashboard*. A mensagem de erro pode ser vista nas Figuras 51 e 52.

Function 47:

```
if (msg.topic[0] == null) {
    msg.payload = "Não foram econtrados dados no período específico. *A primeira data deve ser menor ou igual à segunda!*";

} else {
    msg.payload = " ";
}
return msg;
```

Figura 52 - Gráficos Mensagem de Erro 1

Figura 53 - Gráficos Mensagem de Erro 2

A título de exemplo, a Figura 53 ilustra o funcionamento do código mencionado anteriormente, considerando, neste caso, a temperatura do sensor Sparkfun e do termistor. Ao selecionar o intervalo de datas de 22/06/2023 a 23/06/2023 são amostrados nos gráficos os dados correspondentes a esse intervalo de dias.

É notável que a temperatura do Sparkfun está mais elevada devido ao ambiente onde o sensor estava colocado. Verificou-se que a caixa onde o sensor estava alojado criava um efeito de estufa, provocando um aumento da temperatura pela falta de circulação de ar. Apesar disso, é possível observar padrões de semelhança entre as duas temperaturas registadas.

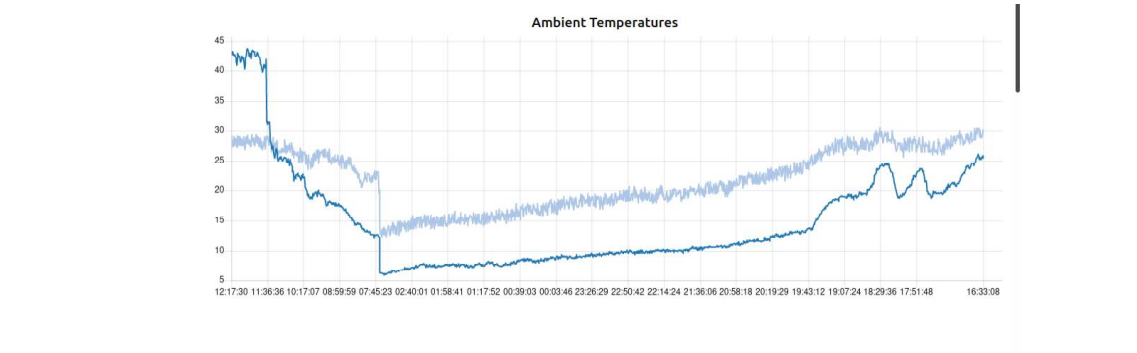


Figura 54 - Gráfico das temperaturas

4.6.1.5. Mecanismo de Segurança

No sistema implementado, caso seja registrada uma temperatura nos painéis igual ou superior a 70 graus, é enviado um e-mail para indicar que um determinado painel ou conjunto de painéis registrou temperaturas acima da recomendada. Essa funcionalidade é importante para monitorar e detectar situações de temperatura elevada, que podem representar um risco ou indicar um mau funcionamento do painel solar. Na Figura 54, está representado o diagrama de estados.

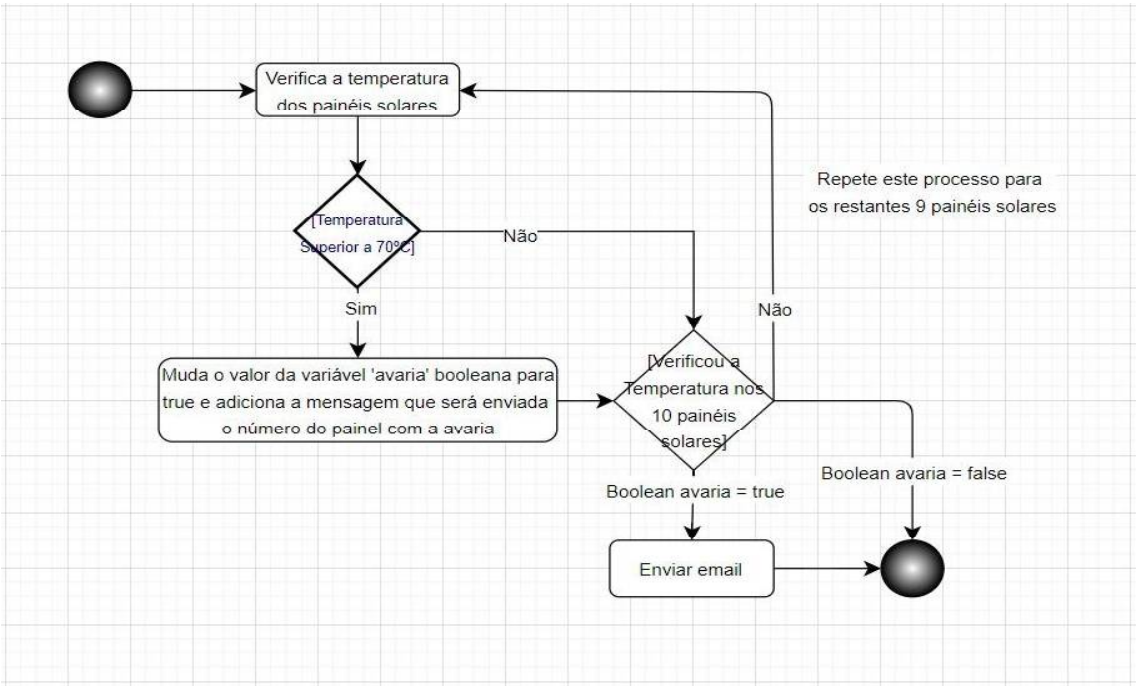


Figura 55 - Diagrama de Estados Envio de Email

A Figura 55 demonstra o grupo de nós no Node RED responsáveis por essa tarefa.

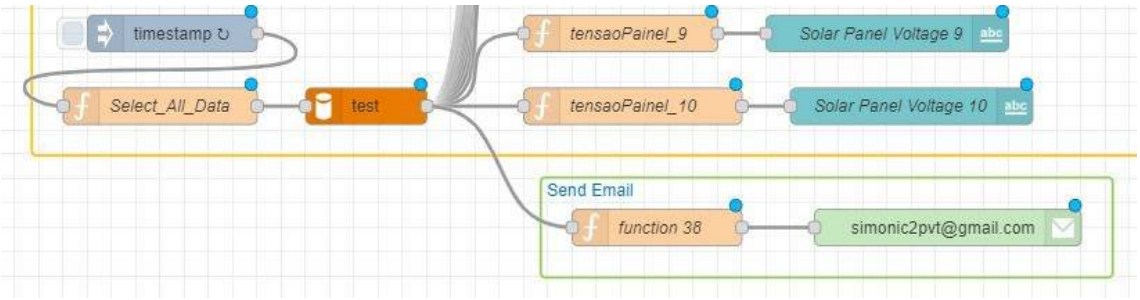


Figura 56 - Envio de um Email

Código:

Function 38:

```
var  first_Element = msg.payload[0];
var paineisComAvarias = [];
var avaria = false;

for (var i = 1; i <= 10; i++) {
   var tempPainel =  first_Element['tempPainel_' + i];
   if (tempPainel >= 70) {
      paineisComAvarias.push(i);
      avaria = true;
   }
}

if (avaria === true) {
   msg.topic = "Temperatura dos Paineis Solares";
   msg.payload = 'Foi registada uma temperatura invulgar nos painéis. Os painéis '
+ paineisComAvarias.join(', ') + ' registaram uma temperatura superior a 70 graus.';
   return msg;

} else {
   return null;
}
```

Email:

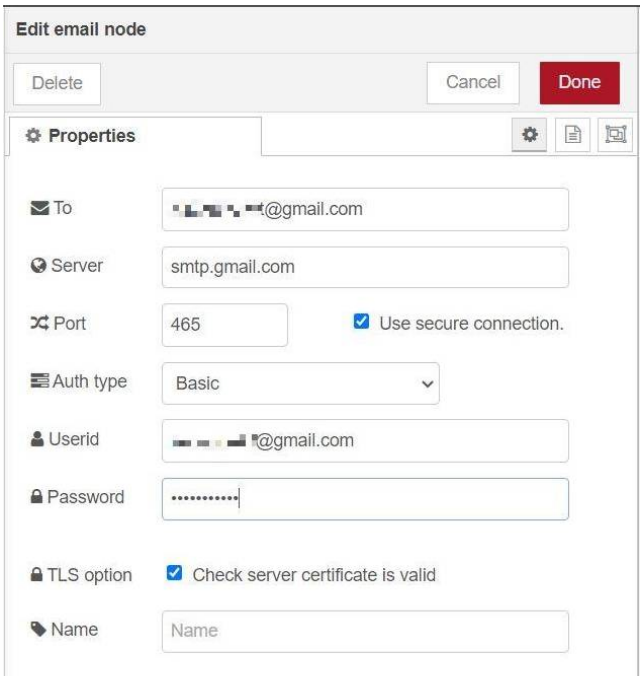


Figura 57 - Configuração do node Email

Para garantir o recebimento das mensagens por e-mail, é necessário realizar algumas configurações específicas. Primeiro é preciso ter uma conta de e-mail configurada e ativa, que será utilizada para enviar e receber as mensagens.

Além disso, é importante realizar uma dupla validação para garantir a segurança do processo. A dupla validação é um recurso de autenticação adicional que requer a confirmação de um código ou senha enviado para um dispositivo confiável, como um *smartphone*, durante o processo de login.

Outra medida de segurança importante é a utilização de uma senha específica para a aplicação que irá enviar as mensagens por e-mail. Essa senha é diferente da senha principal da conta de e-mail e é utilizada exclusivamente para autenticar a aplicação no processo de envio de mensagens.

4.6.1.6. Exportar os dados

Para facilitar a partilha dos dados para outros fins, o código e as imagens abaixo ilustram como é possível permitir que o utilizador selecione uma data, clique num botão e faça o download dos dados correspondentes a essa data num ficheiro de texto, com os dados separados por vírgulas. A Figura 57 demonstra o grupo de nós no NodeRed responsável por essa tarefa.

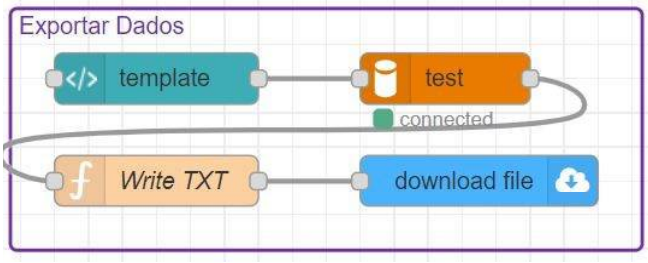


Figura 58 - NodeRed Extrair Dados

Na Figura 58 é demonstrada a interface correspondente no *dashboard*.

Figura 59 - Botão e DatePicker para selecionar os dados a serem baixados

Para desenvolver está funcionalidade foi necessário fazer *download* do pacote "@prescient-devices/node-red-contrib-downloadfile".

Código:

Template:

```
<div style="display: flex;">
   <md-button class="md-raised" ng-click="selectData()" style="flex:
0.15;">Exportar Dados</md-button>

   <md-datepicker ng-model="selectedDate" md-placeholder="data" md-hide-
icons="all"
      md-open-on-focus="true" md-current-view="year" style="margin-left: 15px;
margin-top: 15px;"></md-datepicker>
</div>

<script>
   (function(scope) {
   scope.selectData = function() {
      if (!scope.selectedDate) {
         return;
      }

      var data = new Date(scope.selectedDate);
      data.setMinutes(data.getMinutes() - data.getTimezoneOffset());

      var dataFormatada = data.toISOString().slice(0, 10).replace(/-/g, '');

      function extrairData(dataFormatada) {
         const dataFormatada2 = dataFormatada.slice(0, 10);
         return dataFormatada2;
      }

      var data2 = extrairData(dataFormatada);

      scope.send({topic: "SELECT * FROM mega_meteorological_data WHERE
Date = '" + data2 + "' ORDER BY Hour DESC;"});
   }
})(scope);
</script>
```

Write TXT:

```
var newDataArray = msg.payload.map(item => {
   var date = item.Date;
   var formattedDate = `${date.substr(0, 4)}-${date.substr(4, 2)}-${date.substr(6, 2)}`;

   return `${formattedDate},${item.Hour},${item.Wind_Speed},${item.Wind_Direction},${item.Room_Temperature},${item.Sun},${item.Temperature},${item.Humidity},${item.Pressure},${item.Ambient_Light},${item.IR},${item.UV},${item.LUX}`;
});

msg.payload = newDataArray.join('\n');
```

5. Testes e Resultados

O foco dos testes realizados no sistema foi a análise dos dados adquiridos através dos gráficos, com o objetivo de verificar se os dados registados tanto no cartão como na base de dados eram iguais. Ou seja, pretendia-se avaliar se não houve perda de dados e confirmar se os dados adquiridos têm coerência, bem como verificar se os dados registados no cartão também estão na base de dados principal.

A título de exemplo, as figuras 49, 50 e 51 demonstram os dados registados da Estação Meteorológica no dia 06/07/2023, armazenados no cartão SD, na base de dados e visualizados na tabela correspondente à Estação Meteorológica no painel de controlo (*dashboard*).

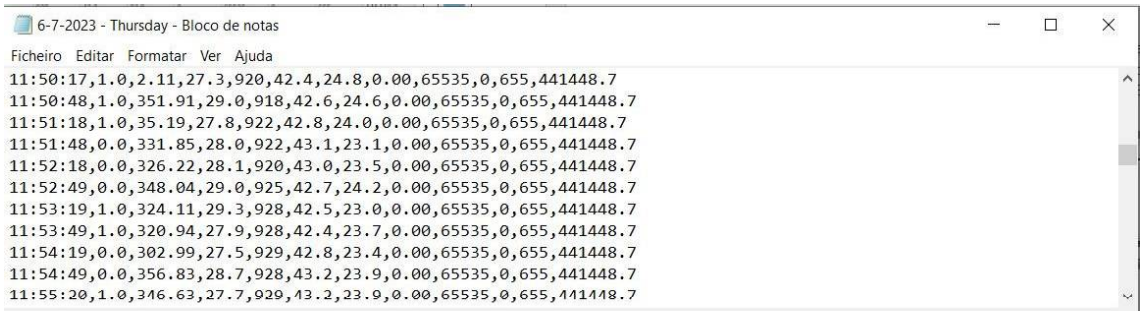


Figura 60 - Dados no Cartão SD

Figura 61 - Dados no Base de Dados

Date	Hour	W_S...	W_Di...	Tem...	Sun	Tem...	Hum	Pres	Amb...	IR	UV	LUX
2023-07-06	11:55:20	1	346.63	27.7	929	43.2	23.9	0	65535	0	655	441448.7
2023-07-06	11:54:49	0	356.83	28.7	928	43.2	23.9	0	65535	0	655	441448.7
2023-07-06	11:54:19	0	302.99	27.5	929	42.8	23.4	0	65535	0	655	441448.7
2023-07-06	11:53:49	1	320.94	27.9	928	42.4	23.7	0	65535	0	655	441448.7
2023-07-06	11:53:19	1	324.11	29.3	928	42.5	23	0	65535	0	655	441448.7
2023-07-06	11:52:49	0	348.04	29	925	42.7	24.2	0	65535	0	655	441448.7
2023-07-06	11:52:18	0	326.22	28.1	920	43	23.5	0	65535	0	655	441448.7
2023-07-06	11:51:48	0	331.85	28	922	43.1	23.1	0	65535	0	655	441448.7
2023-07-06	11:51:18	1	35.19	27.8	922	42.8	24	0	65535	0	655	441448.7
2023-07-06	11:50:48	1	351.91	29	918	42.6	24.6	0	65535	0	655	441448.7
2023-07-06	11:50:17	1	2.11	27.3	920	42.4	24.8	0	65535	0	655	441448.7

Figura 62 - Dashboard NodeRed Dados na Tabela Meteorológica

As informações fornecidas revelam que os registros armazenados no cartão no dia 06/07/2023 são consistentes com os dados presentes na base de dados.

A título de ilustração adicional, a figura seguinte apresenta as leituras de temperatura dos painéis solares entre os dias 05/07/2023 e 06/07/2023.

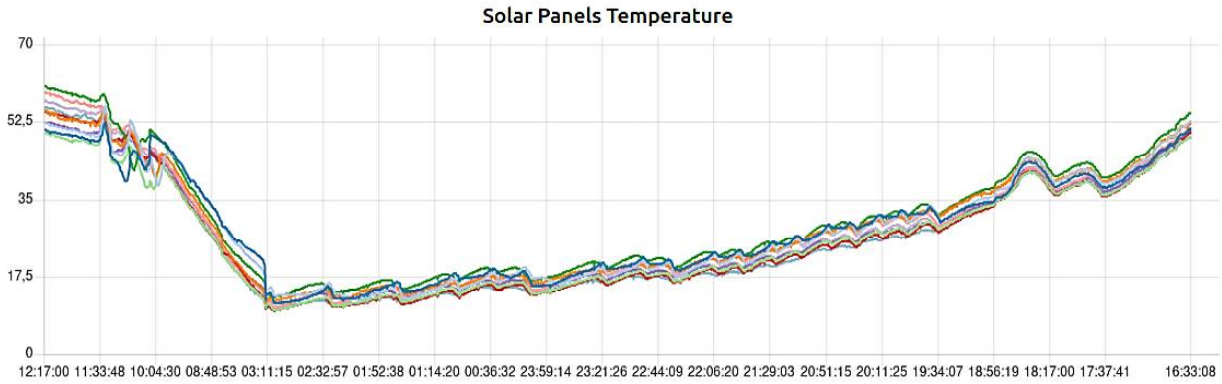


Figura 63 - Gráfico das Temperaturas dos Painéis

Ao analisar os dados, observamos uma consistência notável, indicando que quando a temperatura de um painel aumenta, as temperaturas dos restantes painéis também aumentam. Além disso, é possível constatar uma variação de temperatura de aproximadamente 1 a 2 graus entre os painéis. Podemos também verificar que, durante a noite, a temperatura dos painéis solares diminui, enquanto ao amanhecer ela começa a aumentar gradualmente, o que é esperado devido ao surgimento da luz solar.

Para fins de exemplificação adicional, a figura seguinte exibe as medições da radiação solar ao longo do dia 06/07/2023.

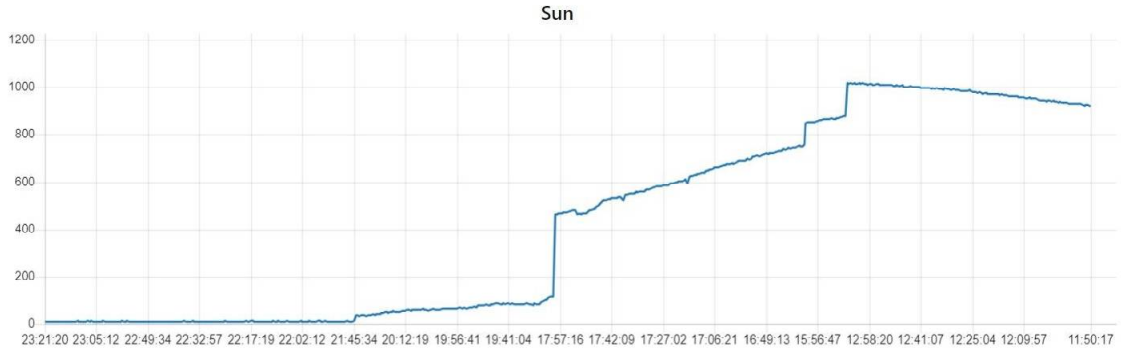


Figura 64 - Gráfico Radiação Solar

Ao analisarmos o gráfico, podemos observar que a radiação solar diminui ao longo do dia, chegando a praticamente zero. Essa tendência está diretamente relacionada à exposição solar, onde um índice de sol mais elevado resulta em uma maior radiação.

6. Conclusões e Recomendações

Neste relatório, apresenta-se a solução desenvolvida para realizar a monitorização remota de duas estações, tendo em consideração as características do ambiente e procurando minimizar os custos de desenvolvimento.

A solução criada permite ao utilizador visualizar os dados recolhidos de forma organizada, através de tabelas e gráficos, e também possibilita a sua transferência. Foram implementadas melhorias em relação ao projeto SIMONIC, destacando-se a redução do tempo de recolha e apresentação dos dados, mesmo com um maior volume de informações. Enquanto o projeto SIMONIC demorava 12 segundos para enviar os dados para a folha de cálculo, realizamos um teste que demonstrou que o processo de recolha e envio dos dados para uma folha de cálculo, através do Node-RED, levou apenas 2 segundos, conforme evidenciado na tabela 70 anexo 5.

Além disso, no projeto atual, optou-se por utilizar uma base de dados MySQL em vez de uma folha de cálculo, como no SIMONIC. Essa mudança permite uma gestão mais eficiente de grandes quantidades de dados acumulados ao longo do tempo.

Ao trabalhar com microcontroladores, é importante considerar as limitações de recursos, como a capacidade de memória e processamento. Por essa razão, é crucial otimizar o código, reduzindo a quantidade de variáveis globais, utilizando tipos de dados mais eficientes em termos de ocupação de espaço e simplificando o código para evitar repetições desnecessárias. Estas práticas possibilitam a escalabilidade do sistema, isto é, a capacidade de adicionar mais dispositivos à rede e recolher mais dados.

No que diz respeito às recomendações, sugerimos algumas ações a serem consideradas. Em caso de perda de ligação à rede Wi-Fi, recomendamos que os dados sejam guardados num ficheiro temporário e, assim que a ligação for restabelecida, esses dados devem ser enviados para o Node-RED pela ordem de chegada, evitando, assim, a perda de informação. Também é aconselhável validar as mensagens enviadas para o Node-RED através de *tokens*, verificando se a mensagem está correta no momento em que chega ao Node-RED. Caso a mensagem não esteja correta, deve-se enviar um pedido ao coordenador para reenviar a mensagem. Uma outra questão importante é validar a qualidade dos dados, o que é crucial para a integridade do sistema, evitando que informações inválidas ou fora dos parâmetros dos sensores sejam inseridas na base de dados. Ao assegurar que apenas dados confiáveis sejam processados pelo Node-RED, contribui-se significativamente para o bom funcionamento do sistema, permitindo uma análise mais precisa e eficiente.

Estas recomendações resultam da experiência adquirida durante o desenvolvimento do projeto. A sua implementação tem como objetivo melhorar a eficiência e a confiabilidade do sistema, garantindo uma recolha e tratamento adequados dos dados.

Em suma, o projeto cumpre os objetivos estabelecidos e apresenta avanços significativos em relação à versão anteriormente desenvolvida.

Referências

[1] Bettaircities, "IoT bettair® Platform," Bettaircities, [Online]. Available: https://bettaircities.com/. [Acesso em Julho 2023].

[2] EDP, "Contadores," EDP, [Online]. Available: https://www.edp.pt/particulares/apoio-cliente/contadores/. [Acesso em Julho 2023].

[3] "GESTÃO INTELIGENTE DE TRÁFEGO VAI EVITAR 205 MILHÕES DE TONELADAS DE CO2 ATÉ 2027, PREVÊ ESTUDO," [Online]. Available: https://smart-cities.pt/noticias/sistemas-gestao-inteligente0706-trafego-mobilidade-urbana-co2-estudo-juniper-research/. [Acesso em Julho 2023].

[4] Freestyle, "Sistema de Monitorização da Glicose," Freestyle, [Online]. Available: https://www.freestyle.abbott/pt-pt/home.html. [Acesso em julho 2023].

[5] J. d. leiria, "TWEvo, startup de Leiria, cria sistema inteligente de produção para a Gallo Vidro," Jornal de leiria, 29 Junho 2023. [Online]. Available: https://www.jornaldeleiria.pt/noticia/twevo-startup-de-leiria-cria-sistema-inteligente-de-producao-para-a-gallo-vidro. [Acesso em Julho 2023].

[6] Luis Saraiva*, Adérito Alcaso, Paulo Vieira, Carlos Figueiredo Ramos, and Antonio Marques, "Development of a cloud-based system for remote monitoring of a PVT panel," *ICEUBI 2015,* 2016.

[7] Luis Saraiva*, Adérito Alcaso, Paulo Vieira, Carlos Figueiredo Ramos, and Antonio Marques, "SIMONIC: Sistema de Monitorização e Controlo de Sistema Solar," Relatório de Estágio Profissionalizante. Escola Superior de Tecnologias e Gestão, Instituto Politécnico da Guarda., 2016.

[8] A. Store, "Arduino Yún Rev 2," [Online]. Available: https://store.arduino.cc/products/arduino-yun-rev-2. [Acesso em Julho 2023].

[9] A. Store, "Arduino Mega 2560 Rev3," Arduino Store, [Online]. Available: https://store.arduino.cc/products/arduino-mega-2560-rev3. [Acesso em Julho 2023].

[10] A. Store, "Arduino Uno Rev3," Arduino Store, [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3. [Acesso em Julho 2023].

[11] A. Store, "Arduino Leonardo with Headers," Arduino Store, [Online]. Available: https://store.arduino.cc/products/arduino-leonardo-with-headers?queryID=66308ffa9c7ea5e5178786ec5517563f. [Acesso em Julho 2023].

[12] A. Store, "Arduino Due," Arduino Store, [Online]. Available: https://store.arduino.cc/products/arduino-due. [Acesso em Julho 2023].

[13] A. Store, "Portenta H7 Lite," Arduino Store, [Online]. Available: https://store.arduino.cc/products/portenta-h7-lite. [Acesso em Julho 2023].

[14] A. Store, "Arduino® Portenta X8," Arduino Store, [Online]. Available: https://docs.arduino.cc/resources/datasheets/ABX00049-datasheet.pdf. [Acesso em Julho 2023].

[15] Fernandok, "Introdução ao ESP32," Fernandok, [Online]. Available: https://www.fernandok.com/2017/11/introducao-ao-esp32.html. [Acesso em Julho 2023].

[16] PJRC, "Teensy®4.0 Development Board," PJRC, [Online]. Available: https://www.pjrc.com/store/teensy40.html. [Acesso em Julho 2023].

[17] PJRC, "Teensy®4.1 Development Board," PJRC, [Online]. Available: https://www.pjrc.com/store/teensy41.html. [Acesso em Julho 2023].

[18] Silabs, "OVERLAY CONSIDERATIONS FOR THE Si114X SENSOR," Silabs, [Online]. Available: https://www.silabs.com/documents/public/application-notes/AN523.pdf. [Acesso em Julho 2023].

[19] Elfadistrelec, "Adafruit Si7021 Temperature + Humidity Sensor," Elfadistrelec, [Online]. Available: https://www.elfadistrelec.no/Web/Downloads/_t/ds/Adafruit_3251_eng_tds.pdf. [Acesso em Julho 2023].

[20] Sparkfun, "Ambient Light Sensor," Sparkfun, [Online]. Available: https://cdn.sparkfun.com/assets/b/e/c/3/d/M_DS.pdf. [Acesso em Julho 2023].

[21] "MPL3115A2 - I2C precision pressure sensor with altimetry," [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MPL3115A2.pdf. [Acesso em Julho 2023].

[22] E. tutorials, "The Multiplexer," Electronics tutorials, [Online]. Available: https://www.electronics-tutorials.ws/combination/comb_2.html. [Acesso em Julho 2023].

[23] MQTT, "MQTT: The Standard for IoT Messaging," MQTT, [Online]. Available: https://mqtt.org/. [Acesso em Julho 2023].

[24] JN, "Combate à poluição atmosférica mortal com sensores e satélites.," 2022 Novembro 16. [Online]. Available: https://www.jn.pt/inovacao/combate-a-poluicao-atmosferica-mortal-com-sensores-e-satelites-15358173.html/. [Acesso em Julho 2023].

[25] Adafruit, "PROXIMITY/UV/AMBIENT LIGHT SENSOR IC WITH I2C INTERFACE," Adafruit, [Online]. Available: https://cdn-shop.adafruit.com/datasheets/Si1145-46-47.pdf. [Acesso em Julho 2023].

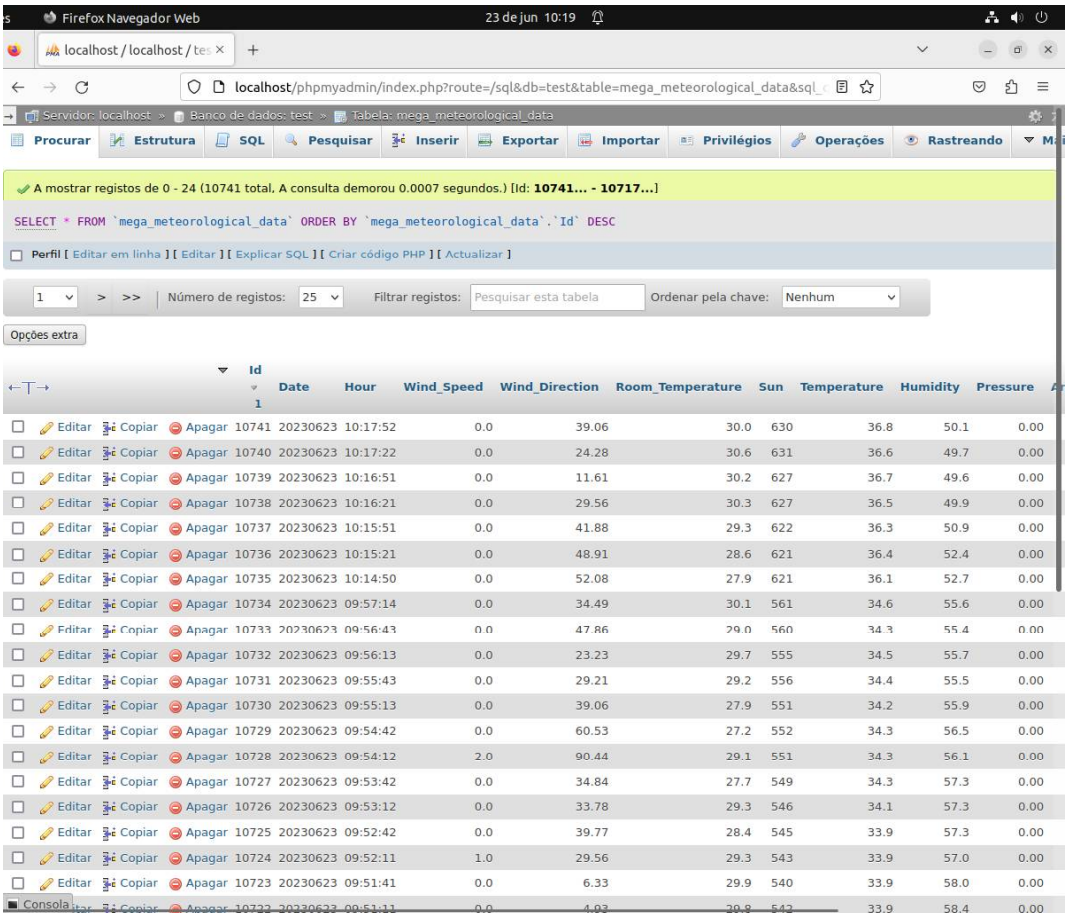
[26] "What is MQTT and Why it is Important for the Internet of Things," [Online]. Available: https://blog.akenza.io/what-is-mqtt?utm_term=&utm_campaign=Dynamic+Search+Campaign+-+EU&utm_source=adwords&utm_medium=ppc&hsa_acc=7184580909&hsa_cam=19248438136&hsa_grp=144426835677&hsa_ad=641301872798&hsa_src=g&hsa_tgt=dsa-19959388920&hsa_kw=&hsa_mt=&hsa. [Acesso em Julho 2023].

[27] Freecodecamp, "What is HTTP? Protocol Overview for Beginners," Freecodecamp, [Online]. Available: https://www.freecodecamp.org/news/what-is-http/. [Acesso em Julho 2023].

[28] Dicasled, "O que é o protocolo ZigBee?," Dicasled, [Online]. Available: https://www.dicasled.pt/o-que-e-o-protocolo-zigbee/. [Acesso em Julho 2023].

Anexos

A1 - Base de Dados - MySQL



Id	Date	Hour	Wind_Speed	Wind_Direction	Room_Temperature	Sun	Temperature	Humidity	Pressure
10741	20230623	10:17:52	0.0	39.06	30.0	630	36.8	50.1	0.00
10740	20230623	10:17:22	0.0	24.28	30.6	631	36.6	49.7	0.00
10739	20230623	10:16:51	0.0	11.61	30.2	627	36.7	49.6	0.00
10738	20230623	10:16:21	0.0	29.56	30.3	627	36.5	49.9	0.00
10737	20230623	10:15:51	0.0	41.88	29.3	622	36.3	50.9	0.00
10736	20230623	10:15:21	0.0	48.91	28.6	621	36.4	52.4	0.00
10735	20230623	10:14:50	0.0	52.08	27.9	621	36.1	52.7	0.00
10734	20230623	09:57:14	0.0	34.49	30.1	561	34.6	55.6	0.00
10733	20230623	09:56:43	0.0	47.86	29.0	560	34.3	55.4	0.00
10732	20230623	09:56:13	0.0	23.23	29.7	555	34.5	55.7	0.00
10731	20230623	09:55:43	0.0	29.21	29.2	556	34.4	55.5	0.00
10730	20230623	09:55:13	0.0	39.06	27.9	551	34.2	55.9	0.00
10729	20230623	09:54:42	0.0	60.53	27.2	552	34.3	56.5	0.00
10728	20230623	09:54:12	2.0	90.44	29.1	551	34.3	56.1	0.00
10727	20230623	09:53:42	0.0	34.84	27.7	549	34.3	57.3	0.00
10726	20230623	09:53:12	0.0	33.78	29.3	546	34.1	57.3	0.00
10725	20230623	09:52:42	0.0	39.77	28.4	545	33.9	57.3	0.00
10724	20230623	09:52:11	1.0	29.56	29.3	543	33.9	57.0	0.00
10723	20230623	09:51:41	0.0	6.33	29.9	540	33.9	58.0	0.00
10722	20230623	09:51:11	0.0	4.93	29.8	543	33.9	58.4	0.00

Figura 65 - Tabela Estação Metereológica - MySQL

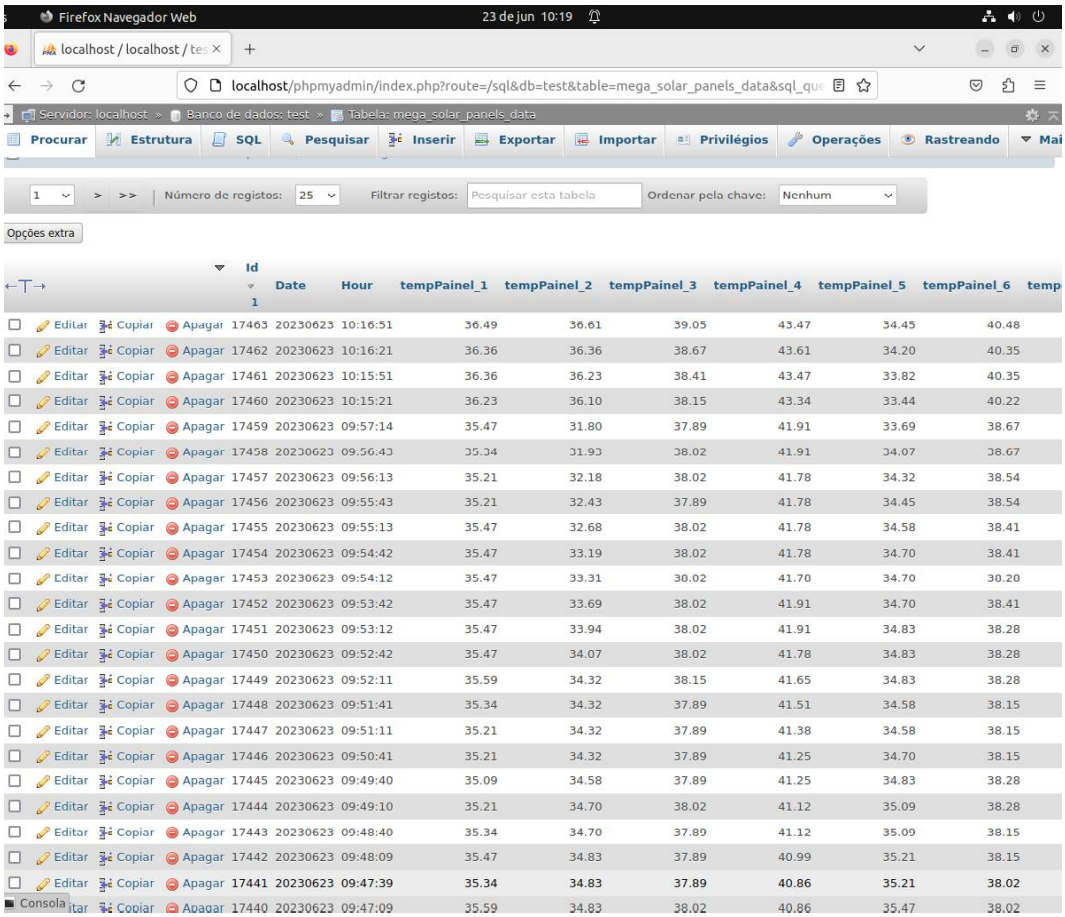


Figura 66 - Tabela Estação Solar - MySQL

A2 - Base de Dados - Backup

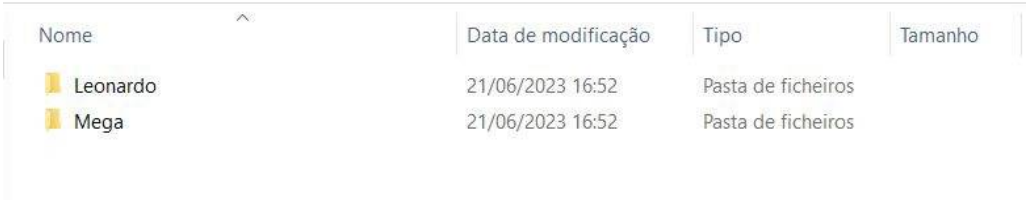
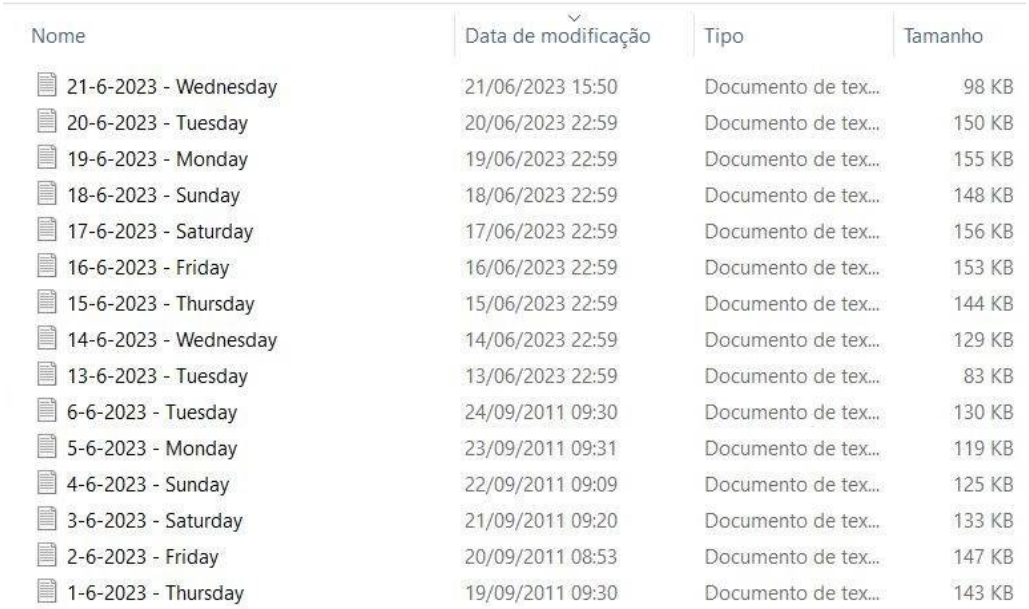


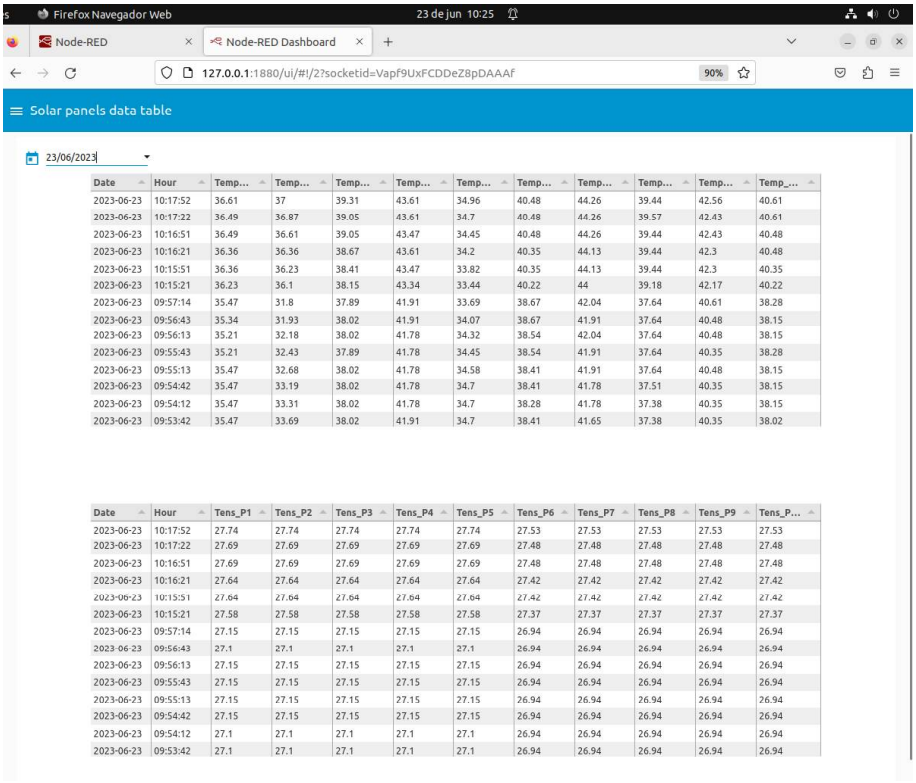
Figura 67 - Organização dos dados por pastas



Nome	Data de modificação	Tipo	Tamanho
21-6-2023 - Wednesday	21/06/2023 15:50	Documento de tex...	98 KB
20-6-2023 - Tuesday	20/06/2023 22:59	Documento de tex...	150 KB
19-6-2023 - Monday	19/06/2023 22:59	Documento de tex...	155 KB
18-6-2023 - Sunday	18/06/2023 22:59	Documento de tex...	148 KB
17-6-2023 - Saturday	17/06/2023 22:59	Documento de tex...	156 KB
16-6-2023 - Friday	16/06/2023 22:59	Documento de tex...	153 KB
15-6-2023 - Thursday	15/06/2023 22:59	Documento de tex...	144 KB
14-6-2023 - Wednesday	14/06/2023 22:59	Documento de tex...	129 KB
13-6-2023 - Tuesday	13/06/2023 22:59	Documento de tex...	83 KB
6-6-2023 - Tuesday	24/09/2011 09:30	Documento de tex...	130 KB
5-6-2023 - Monday	23/09/2011 09:31	Documento de tex...	119 KB
4-6-2023 - Sunday	22/09/2011 09:09	Documento de tex...	125 KB
3-6-2023 - Saturday	21/09/2011 09:20	Documento de tex...	133 KB
2-6-2023 - Friday	20/09/2011 08:53	Documento de tex...	147 KB
1-6-2023 - Thursday	19/09/2011 09:30	Documento de tex...	143 KB

Figura 68 - Organização dos ficheiros

A3 - Interface do Website



Date	Hour	Temp...	Temp...	Temp...	Temp...	Temp...	Temp...	Temp...	Temp...	Temp...	Temp_...
2023-06-23	10:17:52	36.61	37	39.31	43.61	34.96	40.48	44.26	39.44	42.56	40.61
2023-06-23	10:17:22	36.49	36.87	39.05	43.61	34.7	40.48	44.26	39.57	42.43	40.61
2023-06-23	10:16:51	36.49	36.61	39.05	43.47	34.45	40.48	44.26	39.44	42.43	40.48
2023-06-23	10:16:21	36.36	36.36	38.67	43.61	34.2	40.35	44.13	39.44	42.3	40.48
2023-06-23	10:15:51	36.36	36.23	38.41	43.47	33.82	40.35	44.13	39.44	42.3	40.35
2023-06-23	10:15:21	36.23	36.1	38.15	43.34	33.44	40.22	44	39.18	42.17	40.22
2023-06-23	09:57:14	35.47	31.8	37.89	41.91	33.69	38.67	42.04	37.64	40.61	38.28
2023-06-23	09:56:43	35.34	31.93	38.02	41.91	34.07	38.67	41.91	37.64	40.48	38.15
2023-06-23	09:56:13	35.21	32.18	38.02	41.78	34.32	38.54	42.04	37.64	40.48	38.15
2023-06-23	09:55:43	35.21	32.43	37.89	41.78	34.45	38.54	41.91	37.64	40.35	38.28
2023-06-23	09:55:13	35.47	32.68	38.02	41.78	34.58	38.41	41.91	37.64	40.48	38.15
2023-06-23	09:54:42	35.47	33.19	38.02	41.78	34.7	38.41	41.78	37.51	40.35	38.15
2023-06-23	09:54:12	35.47	33.31	38.02	41.78	34.7	38.28	41.78	37.38	40.35	38.15
2023-06-23	09:53:42	35.47	33.69	38.02	41.91	34.7	38.41	41.65	37.38	40.35	38.02

Date	Hour	Tens_P1	Tens_P2	Tens_P3	Tens_P4	Tens_P5	Tens_P6	Tens_P7	Tens_P8	Tens_P9	Tens_P...
2023-06-23	10:17:52	27.74	27.74	27.74	27.74	27.74	27.53	27.53	27.53	27.53	27.53
2023-06-23	10:17:22	27.69	27.69	27.69	27.69	27.69	27.48	27.48	27.48	27.48	27.48
2023-06-23	10:16:51	27.69	27.69	27.69	27.69	27.69	27.48	27.48	27.48	27.48	27.48
2023-06-23	10:16:21	27.64	27.64	27.64	27.64	27.64	27.42	27.42	27.42	27.42	27.42
2023-06-23	10:15:51	27.64	27.64	27.64	27.64	27.64	27.42	27.42	27.42	27.42	27.42
2023-06-23	10:15:21	27.58	27.58	27.58	27.58	27.58	27.37	27.37	27.37	27.37	27.37
2023-06-23	09:57:14	27.15	27.15	27.15	27.15	27.15	26.94	26.94	26.94	26.94	26.94
2023-06-23	09:56:43	27.1	27.1	27.1	27.1	27.1	26.94	26.94	26.94	26.94	26.94
2023-06-23	09:56:13	27.15	27.15	27.15	27.15	27.15	26.94	26.94	26.94	26.94	26.94
2023-06-23	09:55:43	27.15	27.15	27.15	27.15	27.15	26.94	26.94	26.94	26.94	26.94
2023-06-23	09:55:13	27.15	27.15	27.15	27.15	27.15	26.94	26.94	26.94	26.94	26.94
2023-06-23	09:54:42	27.15	27.15	27.15	27.15	27.15	26.94	26.94	26.94	26.94	26.94
2023-06-23	09:54:12	27.1	27.1	27.1	27.1	27.1	26.94	26.94	26.94	26.94	26.94
2023-06-23	09:53:42	27.1	27.1	27.1	27.1	27.1	26.94	26.94	26.94	26.94	26.94

Figura 69 - Tabela com os dados da Estação Solar

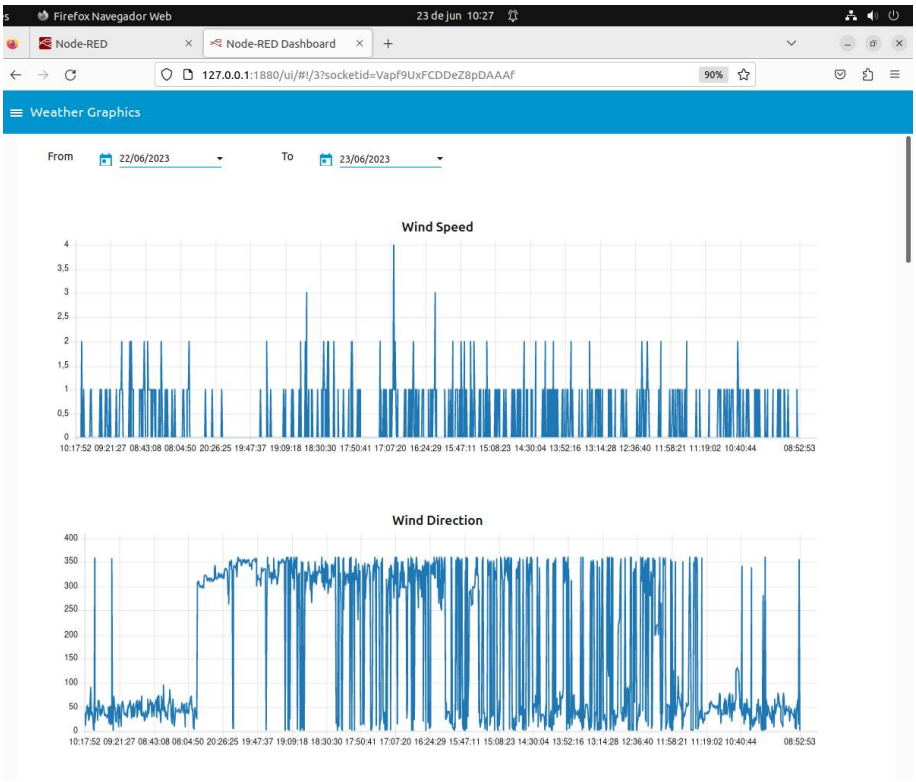


Figura 70 - Gráfico da velocidade e da direção do vento

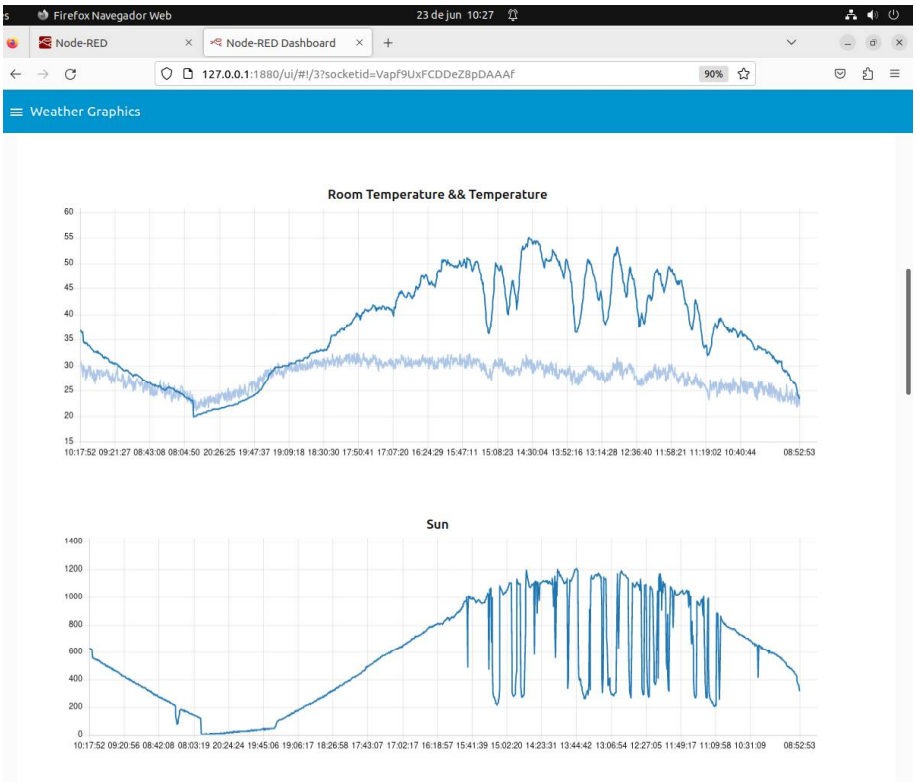


Figura 71 - Gráfico das temperaturas e da radiação solar

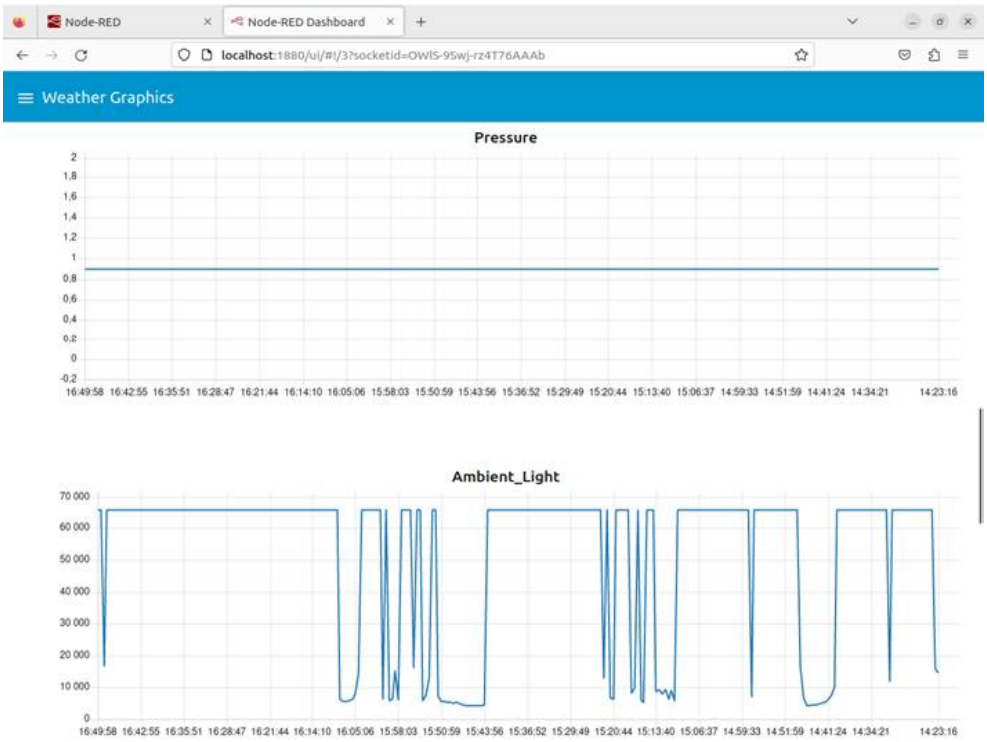


Figura 72 - Gráfico da pressão atmosférica e da luz ambiente

Figura 73 - Gráfico da humidade e do lux

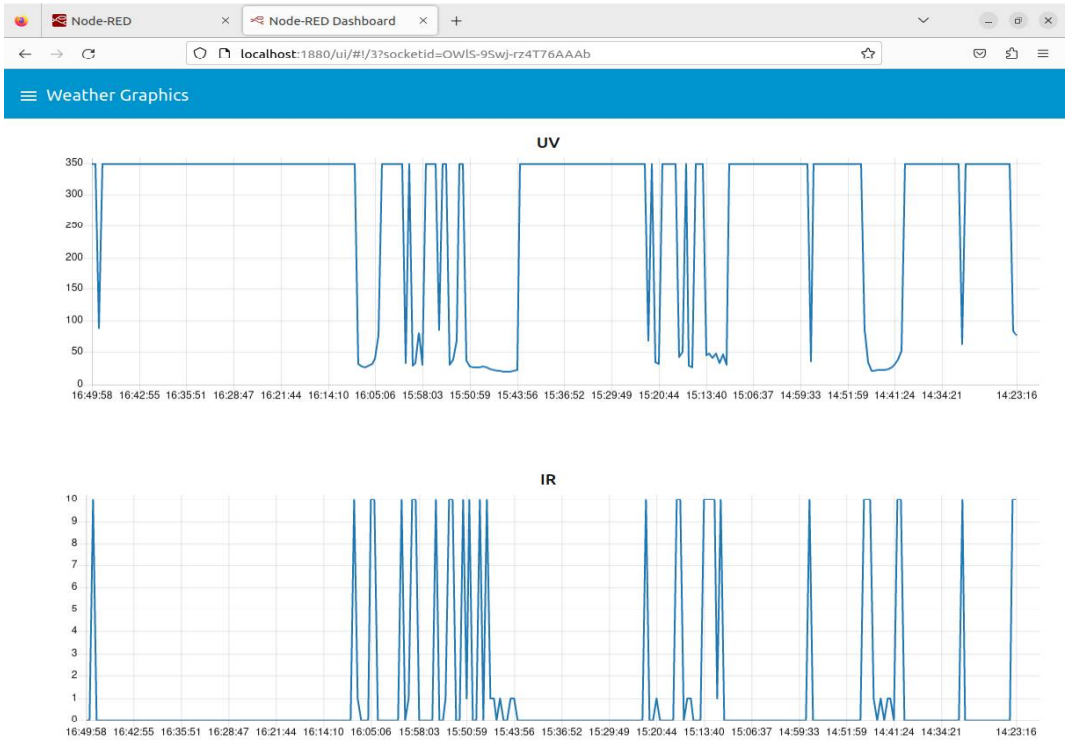


Figura 74 - Gráfico do uv e do ir

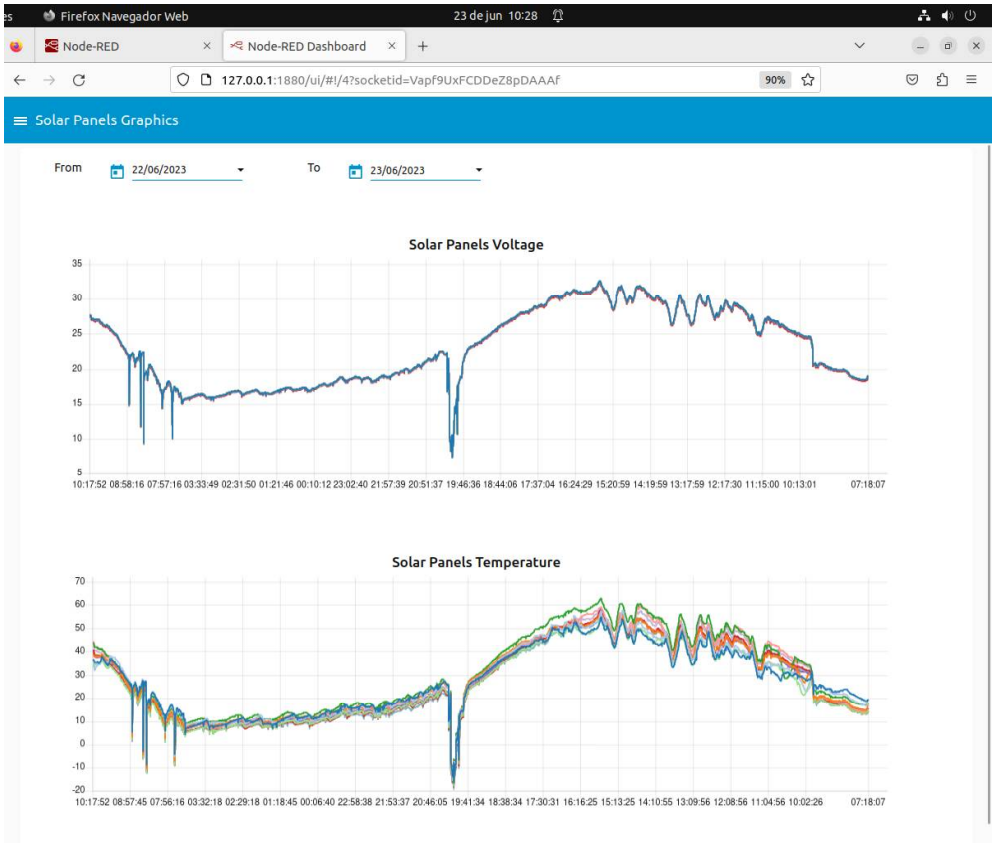


Figura 75 - Gráfico da temperatura e da tensão dos painéis solares

A4 - Fotografias dos Circuitos - EndDevices e Coordenador

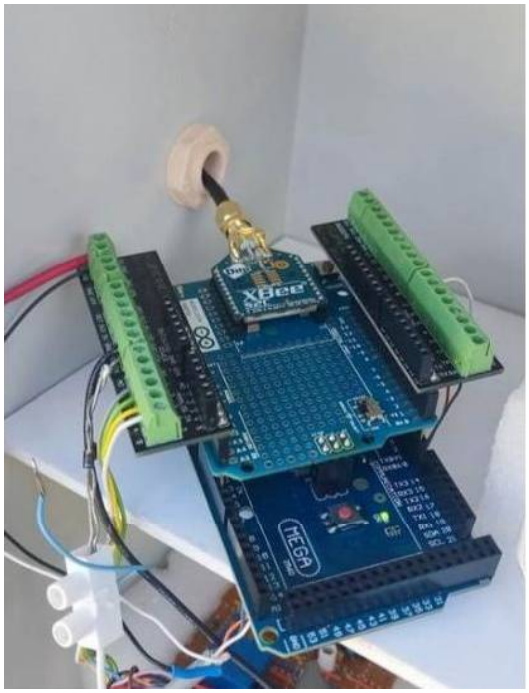


Figura 76 - Circuito EndDevice - Estação Metereológica

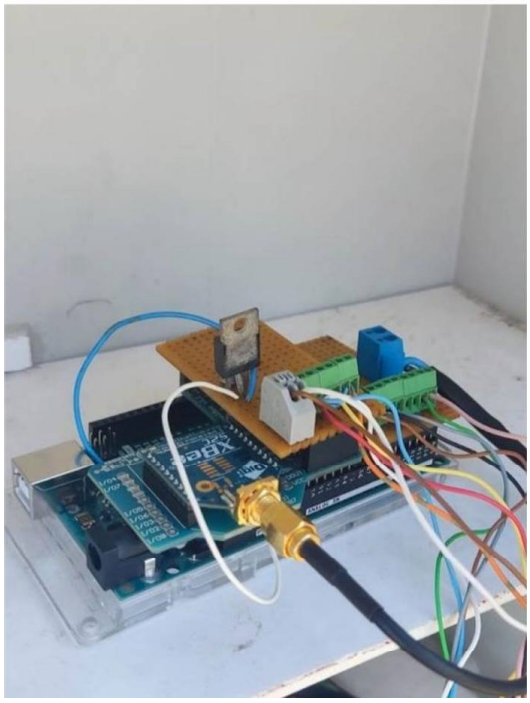


Figura 77 - Circuito EndDevice - Estação Solar

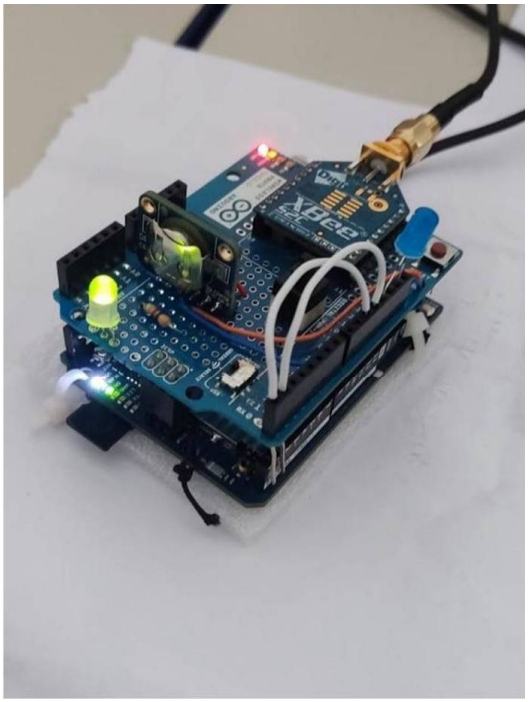


Figura 78 - Circuito Coordenador

A5 - Tempo de envio dos dados para uma folha de cálculo

	A	B	C	D	E	F
1	Carimbo de data/hora	Date	Hour	Wind_Speed	Wind_Direction	Zonen_Temperature
2	2023/06/23 13:51:04	20230623	13:51:2	0	38.71	32.3
3	2023/06/23 13:51:35	20230623	13:51:33	0	4.22	31.6
4	2023/06/23 13:52:05	20230623	13:52:3	0	15.13	31.9
5	2023/06/23 13:52:35	20230623	13:52:33	0	41.88	32.2
6	2023/06/23 13:53:05	20230623	13:53:3	2	94.66	31
7	2023/06/23 13:53:36	20230623	13:53:34	1	122.82	31.3
8	2023/06/23 13:54:06	20230623	13:54:4	3	133.72	29.6
9	2023/06/23 13:54:36	20230623	13:54:34	1	78.83	29.5
10	2023/06/23 13:55:06	20230623	13:55:4	1	68.27	30.2

Figura 79 - Folha de Calculo

A6 - Código Coordenador

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // Cria uma instância de
comunicação serial utilizando os pinos 10 (RX) e 11 (TX)

#include <Wire.h>
#include <Bridge.h>
#include <YunClient.h>
#include <PubSubClient.h>
#include <FileIO.h>
#include "RTClib.h"
RTC_DS3231 rtc; // Cria uma instância do módulo de tempo real
RTC_DS3231

YunClient client;
PubSubClient mqttClient(client); // Cria uma instância do
cliente MQTT utilizando o cliente Yun

// Configurações do broker MQTT
const char* mqttServer = "test.mosquitto.org";
const int mqttPort = 1883;

const char* DaysOfTheWeek[] = {"Sunday", "Monday", "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday"};

uint32_t last_data_send = 0;
const uint16_t time_interval = 29000;

String msg;
String msg_EM;
String msg_ES;
String msg_lab;

String Day;
String Hours;
String formatted_date;

volatile bool msg_EM_ok = false;
volatile bool msg_ES_ok = false;
volatile bool msg_lab_ok = false;

const int buttonPin = 4;
const int ledYun = 13;
const int ledBicVermelho = 5;
const int ledBicVerde = 6;
const int ledPin_NodeRed = 12;
volatile bool state = false;
int buttonState = 0;
bool buttonPressed = false;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledYun, OUTPUT);
  pinMode(ledBicVermelho, OUTPUT);
  pinMode(ledBicVerde, OUTPUT);
```

```
  pinMode(ledPin_NodeRed, OUTPUT);

  Wire.begin();

  mySerial.begin(9600); // Inicia a comunicação serial com a
velocidade de 9600 bps
  delay(400);
  while(!mySerial){
    digitalWrite(ledBicVermelho, HIGH); // Acende o LED vermelho
se a comunicação serial não for estabelecida

  }

  while(!rtc.begin()){
    //Serial.println("Error: RTC_DS3231 MODULE");
    digitalWrite(ledBicVermelho, HIGH); // Acende o LED vermelho
se o módulo RTC não for inicializado corretamente
  }

  // Para atualizar data e hora em caso de perda de energia
  //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  //rtc.adjust(DateTime(2023, 5, 17, 15, 06, 00));

  Bridge.begin();
  Console.begin();
  FileSystem.begin();

  // Configuração do servidor MQTT
  mqttClient.setServer(mqttServer, mqttPort);
}

void loop() {
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH && !buttonPressed) {
    buttonPressed = true;
    state = !state; // Alterna o estado da variável "state"
  }

  if (buttonState == LOW) {
    buttonPressed = false;
  }

  if (state) {
    digitalWrite(ledYun, HIGH);  // Acende o LED conectado ao
pino 13 se "state" for verdadeiro

  } else {
    digitalWrite(ledYun, LOW); // Apaga o LED conectado ao pino
13 se "state" for falso
    if((millis()-last_data_send >= time_interval) &&
(!msg_EM_ok)){
      digitalWrite(ledBicVermelho, LOW); // Apaga o LED vermelho
      digitalWrite(ledBicVerde, HIGH); // Acende o LED verde
      get_date_and_time();
      mySerial.println("#send*"); // Envia um comando pela
comunicação serial
      mySerial.flush();
      msg_EM = extracted_msg('$','Z'); // Extrai uma mensagem da
comunicação serial
```

```
      last_data_send = millis();
    }

    if(msg_EM_ok){
      delay(100);
      mySerial.println("#send2*");
      mySerial.flush();
      msg_ES = extracted_msg('!','W');

      if(msg_EM_ok && msg_ES_ok){
        delay(100);
        save_info("EstacaoSolar", msg_ES);
        delay(100);
        save_info("EstacaoMeteorologica", msg_EM);

        digitalWrite(ledBicVerde, LOW);
        digitalWrite(ledPin_NodeRed, HIGH);
        delay(100);
          NodeRed_Publish("mega",msg_ES);
          delay(100);
          NodeRed_Publish("leonardo",msg_EM);
        digitalWrite(ledPin_NodeRed, LOW);
      }
      msg_EM_ok = false;
      msg_ES_ok = false;

    }

  }
}


String extracted_msg(char initial_character, char
ending_character) {

  String msg2;
  msg = mySerial.readString(); // Lê os dados disponíveis na
comunicação serial
  uint8_t start_index = msg.indexOf(initial_character),
end_index = msg.indexOf(ending_character); // Encontra os
índices dos caracteres inicial e final na mensagem lida
  if (start_index != -1 && end_index != -1) {
    msg2 = msg.substring(start_index + 1, end_index); // Extrai
a parte da mensagem entre os caracteres inicial e final
    msg2.trim();
    uint8_t posicaoUltimaVirgula = (msg2.lastIndexOf(",")+1);
    uint8_t Lenght2 = msg2.length();
    uint8_t Size =
(msg2.substring(posicaoUltimaVirgula,Lenght2)).toInt(); //
Extrai o tamanho da mensagem
    msg2 = msg2.substring(0,(posicaoUltimaVirgula-1));
    if ((posicaoUltimaVirgula == Size) && (Size > 0)) { //
comparar o lenght enviado na mensagem com o lenght neste preciso
momento (extrair a virgula).
      if(initial_character  == '!'){
        msg_ES_ok = true;
      } else {
        msg_EM_ok = true;
      }
```

```
    }
  }
  return msg2;
}

void get_date_and_time(){
  DateTime now = rtc.now();
  Hours = (String(now.hour(), DEC) + ":" + String(now.minute(),
DEC) + ":" + String(now.second(), DEC));
  Day = (String(now.day(), DEC) + "-" + String(now.month(), DEC)
+ "-" + String(now.year(), DEC) + " - " +
String(DaysOfTheWeek[now.dayOfTheWeek()]));

  formatted_date = String(now.year(), DEC);
  if (now.month() < 10) {
    formatted_date += "0";
  }
  formatted_date += String(now.month(), DEC);
  if (now.day() < 10) {
    formatted_date += "0";
  }
  formatted_date += String(now.day(), DEC);
}

void save_info(char *device_folder, String msg2) {
  char file_path[100];

  strcpy(file_path, "/mnt/sda1/");
  strcat(file_path, device_folder);
  strcat(file_path, "/");
  strcat(file_path, Day.c_str());
  strcat(file_path, ".txt");

  File data = FileSystem.open(file_path, FILE_APPEND); // Abre o
arquivo para escrita
  if (data) {
    // Arquivo aberto com sucesso
    data.print(Hours + ","); // Escreve a mensagem no arquivo
    data.flush();
    data.print(msg2);
    data.flush();
    data.println("");
    data.flush(); // Garante que os dados sejam gravados no
arquivo
    data.close(); // Fecha o arquivo
  } else {
    // Erro ao abrir o arquivo
    digitalWrite(ledBicVermelho, HIGH); // Acende o LED vermelho
  }
}

void NodeRed_Publish(String topic, String msg) {
  if (!mqttClient.connected()) { // Conecta-se ao broker MQTT
    reconnect();
  }

  char message[200];
  const char* formatted_date_cstr = formatted_date.c_str();
  const char* hours_cstr = Hours.c_str();
  const char* msg_cstr = msg.c_str();
```

```
  snprintf(message, sizeof(message), "%s,%s,%s",
formatted_date_cstr, hours_cstr, msg_cstr);
  mqttClient.publish(topic.c_str(), message);
  message[0] = '\0';
}


void reconnect() {
  uint32_t last_data_send_NodeRed = millis();
  const uint16_t time_interval_NodeRed = 30000;
  while (!mqttClient.connected() && (millis() -
last_data_send_NodeRed < time_interval_NodeRed)) {
    if (mqttClient.connect("arduinoClient")) {
      break;
    }
  }
}
```

A7 - Código Estação Metereológica

```
String msg_received;
String msg_final;
String data_array[11];

#include "Wire.h"
#include "Adafruit_Sensor.h"
#include "Adafruit_Si7021.h"
Adafruit_Si7021 si7021 = Adafruit_Si7021();

#include <SparkFunMPL3115A2.h>
#include <SI1145_WE.h>
SI1145_WE si1145 = SI1145_WE();
MPL3115A2 mpl3115a2;

#define LED_SI1145 7

void setup() {
  Serial.begin(9600);
  while(!Serial);

  Wire.begin();
  si7021.begin();

  mpl3115a2.begin();
  mpl3115a2.setModeBarometer();
  mpl3115a2.setOversampleRate(7);
  mpl3115a2.enableEventFlags();

  si1145.init();
  si1145.setI2CAddress(0x59); // mudar de 0x60 para 0x59
si1145.enableMeasurements(PSALSUV_TYPE, AUTO);
  //si1145.setAlsVisAdcGain(0);
  si1145.selectPsDiode(SMALL_DIODE); // SMALL_DIODE - MAIS
SENSIVEL
  si1145.selectIrDiode(SMALL_DIODE);
  si1145.setAlsIrAdcGain(SMALL_DIODE);

}

void loop() {
  while(Serial.available()>0){
    if(Serial.read() == '#'){
      msg_received = Serial.readStringUntil('*');
      if(msg_received == "send"){
        digitalWrite(LED_SI1145,HIGH);
        treat_data();
        send_data();
        msg_received="";
        msg_final = "";
      }
    }
  }
}
```

```
void treat_data(){
    char buffer[10];

    // Wind speed
    float caxi1 = round(analogRead(4)*
(5.0/1023.0)*(30.0/4.80));

    dtostrf(caxi1, 3, 1, buffer);
    data_array[0] = buffer;

   // wind direction
    float caxi2 = analogRead(1) *(5.0/1023.0)*(360.0/5.0);
    //uint8_t interval = round(caxi2 /22.5);
    dtostrf(caxi2, 5, 2, buffer);
    data_array[1] = buffer;

    // room temperature
    float k1 = analogRead(2) *(5.0/1023.0);
    float caxi3 = pow(k1,6)*0.2715 - pow(k1,5)*4.1324 +
pow(k1,4)*23.641 - pow(k1,3)*61.447 + pow(k1,2)*66.787 +
k1*3.4292-52.64;

    dtostrf(caxi3, 3, 1, buffer);
    data_array[2] = buffer;


    // SUN
    uint16_t caxi4 = round(analogRead(3)*
(5.0/1023.0)*(1250.0/4.8));

    itoa(caxi4, buffer, 2);
    data_array[3] = caxi4;

    // Si7021 temperature and humidity
    float temperature = si7021.readTemperature();
    float humidity = si7021.readHumidity();
    temperature =isnan(temperature)?0.0:temperature;
    humidity =isnan(humidity)?0.0:humidity;

    dtostrf(temperature, 3, 1, buffer);
    data_array[4] = buffer;

    dtostrf(humidity, 3, 1, buffer);
    data_array[5] = buffer;

    // MPL3115A2 pressure
    float pressure = mpl3115a2.readPressure();
    float atmospheric_pressure = (pressure/ 101325.0);

    dtostrf(atmospheric_pressure, 3, 2, buffer);
    data_array[6] = buffer;

    // SI1145 ambient light, IR and UV index
    uint32_t ambient_light = si1145.getAlsVisData();
    uint8_t amb_ir = si1145.getAlsIrData();
    uint16_t uv = si1145.getUvIndex();
    float lux = calcLux(ambient_light,amb_ir);
    uint16_t ir = amb_ir * 0.01;
```

```
    itoa(ambient_light, buffer, 2);
    data_array[7] = ambient_light;

    itoa(ir, buffer, 2);
    data_array[8] = buffer;

    itoa(uv, buffer, 2);
    data_array[9] = uv;

    lux = lux < 0.0 ? 0.0 : lux;

    dtostrf(lux, 6, 1, buffer);
    data_array[10] = buffer;

    for(int i=0; i< 11; i++){
      msg_final.concat(data_array[i]);
      msg_final += ",";
    }

    msg_final.concat(msg_final.length());

}

void send_data(){
  digitalWrite(LED_SI1145,LOW);
  Serial.println("$"+msg_final+"Z");
  Serial.flush();
}


float calcLux(uint32_t vis, uint16_t ir){
  const uint16_t vis_dark = 256; // empirical value
  const uint8_t ir_dark = 250; // empirical value
  const float gainFactor = 1.0;
  const float visCoeff = 5.41; // application notes AN523
  const float irCoeff = 0.08; // application notes AN523
  const float corrFactor = 1.25; // my empirical correction
factor

  // According to application notes AN523:
  float lx = ((vis - vis_dark) * visCoeff - (ir - ir_dark) *
irCoeff) * gainFactor * corrFactor;
  return lx;
}
```

A8 - Código Estação Solar

```
bool state = false; // Verifique se o pedido de recolha de dados
foi enviado e aguarde até que o segundo pedido seja recebido.
String msg_received; // Variavel que verifica se a mensagem
envia é aquela que pretendemos
String msg_final; // A mensagem final contém os dados
recolhidos, separados por vírgulas.

const uint8_t pins_used[] ={1,2,3,4,5,8,9,10,14,15}; // All the
pins being used are stored in this array.
float data_extracted[20]; // Nesta matriz, são colocados os
dados, ainda por tratar.

// Definição dos pinos de controle de cada multiplexer
const uint8_t s0 = 2, s1 = 3, s2 = 4, s3 = 5, s4 = 6, s5 = 7;

// Definição dos pinos de leitura analógicos de cada multiplexer
const uint8_t analogPin_Mult1  = 12, analogPin_Mult2 = 13;


void setup() {
  // Inicialização da porta serial
  Serial.begin(9600);
  while(!Serial);

  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);
  pinMode(s3, OUTPUT);
  pinMode(s4, OUTPUT);
  pinMode(s5, OUTPUT);

}

void loop() {
  while(Serial.available()>0){
    if(Serial.read() == '#'){
      msg_received = Serial.readStringUntil('*');
      if(msg_received == "send" && !state){
        data_collected();
        treat_data();
        msg_received="";
        state = true;
      }else if(msg_received == "send2" && state){
        send_data();
        msg_received="";
        msg_final="";
        state = false;
      } else {
        return;
      }
    }
  }
}
```

```
void data_collected(){

  for(uint8_t i=0; i<10; i++){
    data_extracted[i] = analogRead(pins_used[i]);
  }

  for(uint8_t i=10; i<15; i++){
    for (int port = 0; port < 5; port++) {
      // Seleciona a porta do multiplexer usando os pinos de
controle
      selectPort_Mult1(port);
      // Lê o valor analógico da porta selecionada
      data_extracted[i] = analogRead(analogPin_Mult1);
    }
  }
  for(uint8_t i=15; i<20; i++){
    for (int port2 = 0; port2 < 5; port2++) {
      selectPort_Mult2(port2);
      data_extracted[i] = analogRead(analogPin_Mult2);
    }
  }
}

void selectPort_Mult1(int port) {
  // Converte o número da porta em sinais binários para os pinos
de controle
  digitalWrite(s0, bitRead(port, 0));
  digitalWrite(s1, bitRead(port, 1));
  digitalWrite(s2, bitRead(port, 2));
}

void selectPort_Mult2(int port2) {
  digitalWrite(s3, bitRead(port2, 0));
  digitalWrite(s4, bitRead(port2, 1));
  digitalWrite(s5, bitRead(port2, 2));
}

void treat_data(){
  for(uint8_t i=0; i<20; i++){
    float x = data_extracted[i] * (5.0 / 1023.0);
    float y = (i<10) ? (1.3225 * pow(x,2)) + ((18.566*x)-27.41)
: (x * 50) / 4.545;
    msg_final.concat(y);
    msg_final += ",";
    data_extracted[i] = 0.0;
  }
  msg_final.concat(msg_final.length());
}

void send_data(){
  Serial.println("!"+msg_final+"W");
  Serial.flush();
}
```

A9 - Código NodeRed

https://drive.google.com/file/d/1mEXiSN9mDY7ihqn4vh83z63zSyyjfOzc/view?usp=sharing