

**UNIVERSIDADE FEDERAL DE SANTA MARIA  
CENTRO DE TECNOLOGIA  
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

**INTERFACE SUPERVISÓRIA PARA SISTEMAS DE  
RASTREAMENTO SOLAR - TRACKER**

**BRUNO GABRIEL FLORES SAMPAIO**

**Santa Maria, RS, Brasil**

**2023**

## **RESUMO**

## **ABSTRACT**

## LISTA DE FIGURAS

Figura 1 : Mapa do potencial de geração solar fotovoltaica em termos do rendimento energético anual para todo o Brasil (medido em kWh/kWp.ano ) .....	11
Figura 2 : Pirâmide de um sistema SCADA .....	15
Figura 3 : Exemplo de MODEM de comunicação. ....	16
Figura 4 : Logo da linguagem de programação Python. ....	17
Figura 5 : Primeiro CLP da Modcon - MODICON 084 .....	19
Figura 6 : Logo da ModBus Organization. ....	19
Figura 7 : Modelo de separação dos campos de dados no protocolo Modbus. ....	20
Figura 8 : Pacotes Modbus. ....	21
Figura 9 : Modelo mestre-escravo <i>modbus</i> . ....	21
Figura 10 : Modelo de requisição e resposta Modbus. ....	22
Figura 11 : Modelos de PDUs do modelo <i>Modbus</i> . ....	22
Figura 12 : Registradores Modbus. ....	23
Figura 13 : Modelo de hierarquia de objetos em interfaces orientadas a objetos. ....	25
Figura 14 : Display utilizado como IHM. ....	28
Figura 15 : Hierarquia do modelo de organização MVC .....	30
Figura 16 : Modelo de requisição de login. ....	31
Figura 17 : Fluxograma do processo de controle do nível de acesso. ....	32
Figura 18 : Fluxograma das responsabilidades do Modelo. ....	34
Figura 19 : Fluxograma relacional entre o Modelo e o gerenciador do servidor remoto. ....	35

## **LISTA DE TABELAS**

Tabela 1: Relação dos diferentes tipos de geração elétrica no Brasil. ....	9
--	---

## SUMÁRIO

<b>1. INTRODUÇÃO .....</b>	<b>7</b>
1.1 Aspectos gerais .....	7
1.2 Antecedente do problema .....	10
1.3 Descrição do problema .....	12
1.4 Objetivo .....	12
1.5 Objetivos específicos .....	12
<b>2. REVISÃO BIBLIOGRÁFICA .....</b>	<b>14</b>
2.1 O que é um sistema SCADA .....	14
2.2 Protocolos de comunicação .....	16
<b>3. MATERIAIS E MÉTODOS .....</b>	<b>17</b>
3.1 Especificações de <i>Software</i> - Python 3.10 .....	17
3.2 Comunicação de dados – <i>Modbus</i> .....	18
3.2.1 Estrutura dos pacotes <i>Modbus</i> .....	20
3.2.2 Modelo de comunicação mestre-escravo .....	21
3.2.3 Código de função .....	22
3.2.4 Tipos de registradores .....	23
3.3 Interface gráfica - <i>Kivy</i> .....	24
3.4 Banco de dados - <i>SQLite3</i> .....	26
3.5 Decisão .....	26
3.6 Especificações de <i>Hardware</i> .....	27
3.6.1 Raspberry Pi 3 .....	27
3.6.2 IHM .....	28
<b>4. DESENVOLVIMENTO .....</b>	<b>29</b>
4.1 <i>Software</i> de controle .....	29
4.1.1 Organização do <i>Software</i> de interface .....	29
4.1.2 Processos internos .....	30
4.1.3 Modelo .....	33
4.1.4 Tela gráfica .....	37
<b>5. INTERFACE DE CONTROLE .....</b>	<b>38</b>
5.1 Tela de Login .....	38
5.2 Tela de início .....	41
5.3 Tela de mapas .....	41
5.4 Tela de conexão serial .....	41
5.5 Tela de sensoriamento .....	41
5.6 Tela de diagnosticos .....	42
<b>6. RESULTADOS .....</b>	<b>43</b>

6.1 Comunicação com o sistema .....	43
<b>7. DISCUSSÃO .....</b>	<b>44</b>
7.1 Interpretação dos resultados .....	44
7.2 Implicações teóricas da pesquisa .....	44
7.3 Confiança estimada da conclusão .....	44
7.4 Restrições de projeto .....	44
7.5 Recomendações para pesquisas futuras .....	44
<b>CONCLUSÃO .....</b>	<b>45</b>
<b>BIBLIOGRAFIAS .....</b>	<b>46</b>
7.6 Introdução .....	46
7.7 Revisão bibliográfica .....	47
<b>ANEXO .....</b>	<b>51</b>
<b>APÊNDICE .....</b>	<b>52</b>

# 1. INTRODUÇÃO

## 1.1 Aspectos gerais

Precedendo a era heliocentrista, baseados nos fortes argumentos propostos por Aristóteles (384 a.C. a 322 a.C.) e Ptolomeu (90 d.C. – 168 d.C.) de que a Terra era o centro do universo, adotou-se a visão geocentrista do mundo. Não é descabível pensar na terra ocupando o centro do universo e todos os astros girando em seu torno quando leis fundamentais da natureza ainda eram desconhecidas e as teorias eram obtidas com bases experimentais e através da pura observação, somados a recursos muito limitados.

No entanto, rompendo tais ideologias e indo contra a verdade adotada na época, Nicolau Copérnico (1473-1543), no ano de 1543, já em seu leito de morte, publicou uma obra “Da revolução de esferas celestes”, tradução livre para o português de “*De Revolutionibus Orbium Coelestium*”, um livro que continha todos os anos de pesquisa de Copérnico acerca do movimento do sol e os demais corpos celestes conhecidos na época com forte embasamento matemático. Além disso, Copérnico pela primeira vez apresentou publicamente a ideia de que o planeta Terra realiza 3 tipos de movimentos, sendo eles: O movimento de rotação em torno do seu próprio eixo (Rotação diária), o movimento de translação, que realiza ao redor do sol (Volta anual) e o movimento de precessão que realiza em torno do seu eixo eclíptico (inclinação anual de seu eixo). Tal livro gerou uma verdadeira revolução na maneira como o mundo era visto na época e foi o responsável por posteriormente derrubar por terra as teorias Geocentristas vigentes na época (COPÉRNICO, 2003).

Servindo de inspiração para outros grandes nomes da física, como Johannes Kepler (1571 - 1630) que em seu livro “Astronomia Nova” publicado em 1610, apresentava as suas duas primeiras leis acerca do movimento dos planetas ao redor do sol (TOSSATO; MARICONDA, 2010) e Galileu Galilei (1564 – 1642) com seu folheto “Mensageiro Sideral”, tradução livre de “*Sidereus Nuncius*”, publicado em 1610 que apresentou observações feitas do espaço através de telescópios desenvolvidos pelo próprio Galileu, trazendo novos fatos acerca do que viria ser descoberto como a Via Láctea e o universo, trazendo consigo fortes argumentos que confrontavam o geocentrismo.

Esses cientistas foram muito importantes nos seus campos de pesquisa e dedicaram suas vidas, arriscando-se para fazer ciência em um tempo que isso não era permitido, pois



iniciaram uma busca por conhecimento nos céus, estudando os astros que nos rodeiam e trazendo informações a cerca deles, que formam a base de tudo que se é aprendido e aceito hoje.

Com as descobertas do comportamento dos astros, pode-se então compreender suas características de movimento, traçar trajetórias, calcular suas posições e fazer previsões de posição de cada um no céu de maneira precisa. Quando focamos no Sol como nosso astro de estudo, tais feitos se tornam ainda mais importantes uma vez que o sol é uma grande fonte de energia, principalmente nos dias de hoje, ao qual somos capazes de aproveitar sua energia, não apenas no âmbito da agricultura ou arquitetura, mas também como uma fonte geradora de energia elétrica inacabável.

Com as descobertas feitas por Alexandre Edmond Becquerel em 1839 quando descobriu o efeito fotovoltaico, efeito que transforma a energia dos raios solares em energia elétrica e Willoughby Smith que mais tarde, em 1873 descobriu a fotocondutividade do selênio, que originalmente era um isolante, mas se comportava como um condutor na presença de raios solares e não apenas conduzia eletricidade como também era capaz de gera-la, teve-se em 1883 a criação da primeira célula fotovoltaica por Charles Fritts e em 1958, Russell Ohl patenteou o primeiro sistema fotovoltaico, o mais próximo do que temos hoje (RICHARDSON, 2018), um sistema capaz de produzir energia elétrica através dos raios solares que poderia ser facilmente instalado em qualquer lugar.

Não se sabia na época que tais descobertas e invenções iriam mudar a forma como a geração de energia passou a ser feita ao redor do mundo, tornando a geração solar uma das principais e mais cobiçadas fontes geradoras de energia renováveis no planeta. Atualmente há um grande número de incentivos em escala global para essa prática de geração, um reflexo dos incentivos mencionados, foi a produção primária de energia solar por painéis fotovoltaicos aumentando 395% entre 2003 e 2013, frente a 56% das outras fontes renováveis. Somente o crescimento da geração de energia eólica superou a energia solar nesse período (SILVA, 2015).

No Brasil, em 2021 a marca de geração solar está para atingir 8GW de potência (sieBRASIL, sd), representando apenas 4% da geração de energia no país, como mostrado na Tabela 1, um valor baixo se comparado ao potencial de geração que esse tipo de tecnologia pode nos oferecer. Atualmente, a maior usina de geração solar do Brasil está localizada na cidade de São Gonçalo do Gurguéia, no Piauí. Contando com mais de 2.2 milhões de painéis solares em uma região semiárida do Brasil podendo chegar a gerar 2,2TW por ano de energia (GREEN POWER, 2021) sendo também considerada a maior Usina de geração solar da

América do sul. A Usina de São Gonçalo por sua vez, conta com um sistema de rastreamento do sol com um grau de liberdade, sendo capaz de seguir o sol no seu movimento de azimute e zenite, variando sua inclinação com o solo devido sua posição geográfica favorecida.

**Tabela 1: Relação dos diferentes tipos de geração elétrica no Brasil.**

Tipo de geração	Unidade	Produção	Porcentagem
Hidrelétrica	MW	109294,05	60,89%
Eólica	MW	17146,13	9,55%
Solar	MW	7922,22	4,41%
Térmica	MW	43152,37	24,04%
Nuclear	MW	1990,00	1,11%
Total	MW	179504,77	100,00%
Valores referentes a 01 set 2021			

**Fonte 1: adaptação de sieBRASIL, sd.**

Ao contrário dos painéis móveis, os painéis solares fixos, que possuem uma limitação de irradiação que se agrava na medida em que se sai das regiões próximas à linha do equador. Devido aos movimentos do sol ao longo do dia (rotação), somados aos movimentos do sol ao longo do ano (translação), os painéis fixos perdem grande parte da irradiação que poderiam receber no período do dia. Estima-se que 40% da energia solar é desperdiçada para configurações de painéis fixos em regiões afastadas da linha do equador quando comparados com painéis não fixos com 2 graus de liberdade e até 30% com apenas um grau de liberdade (VALLDOREIX GREENPOWER, 2015).

Visando um notável crescimento no uso de sistemas de rastreamento solar na geração de energia, esses sistemas têm se tornado uma escolha popular devido ao seu potencial de aumentar significativamente a eficiência e o rendimento das instalações solares. No entanto, à medida que as instalações solares com *Trackers* se tornam mais complexas e numerosas, surge a necessidade de um sistema supervisor confiável para gerenciar e controlar eficientemente esses sistemas, principalmente em redes complexas com numerosos sistemas *Tracker* pendurados na mesma rede.

Um sistema supervisor desempenha um papel crucial no controle e monitoramento desses sistemas, permitindo um posicionamento preciso dos painéis solares e a otimização do rendimento energético. O sistema supervisor coleta dados dos sensores, como a posição do sol, a intensidade da luz solar e a produção de energia, e os exibe de forma clara e intuitiva em uma interface gráfica. Com base nessas informações em tempo real, os operadores podem tomar decisões informadas para maximizar a eficiência da geração de energia solar.

Além disso, um sistema supervisório oferece recursos avançados de controle, permitindo que os operadores realizem ajustes e modifiquem as estratégias de movimento, incluindo a programação de perfis de movimento, a adaptação a condições climáticas variáveis ou o alinhamento com padrões específicos de geração de carga. Com a automação proporcionada pelo sistema supervisório, os ajustes e o controle dos sistemas de rastreamento podem ser realizados de forma eficiente e precisa, eliminando a necessidade de intervenção manual frequente e como consequência, reduzindo a necessidade de manutenções corretivas.

Sistemas supervisórios oferecem recursos avançados de detecção e diagnóstico de falhas. Ele pode monitorar constantemente o desempenho dos *Trackers*, detectando problemas, como mau funcionamento de sensores ou falhas mecânicas, e fornecendo alertas e notificações em tempo real e colocando a planta em estados de emergência caso preciso. Isso permite uma rápida intervenção e manutenção corretiva, minimizando o tempo de inatividade e garantindo a operação contínua e confiável do sistema.

Com um sistema supervisório, as instalações solares podem alcançar todo o seu potencial, contribuindo para a transição para fontes de energia mais limpas e sustentáveis.

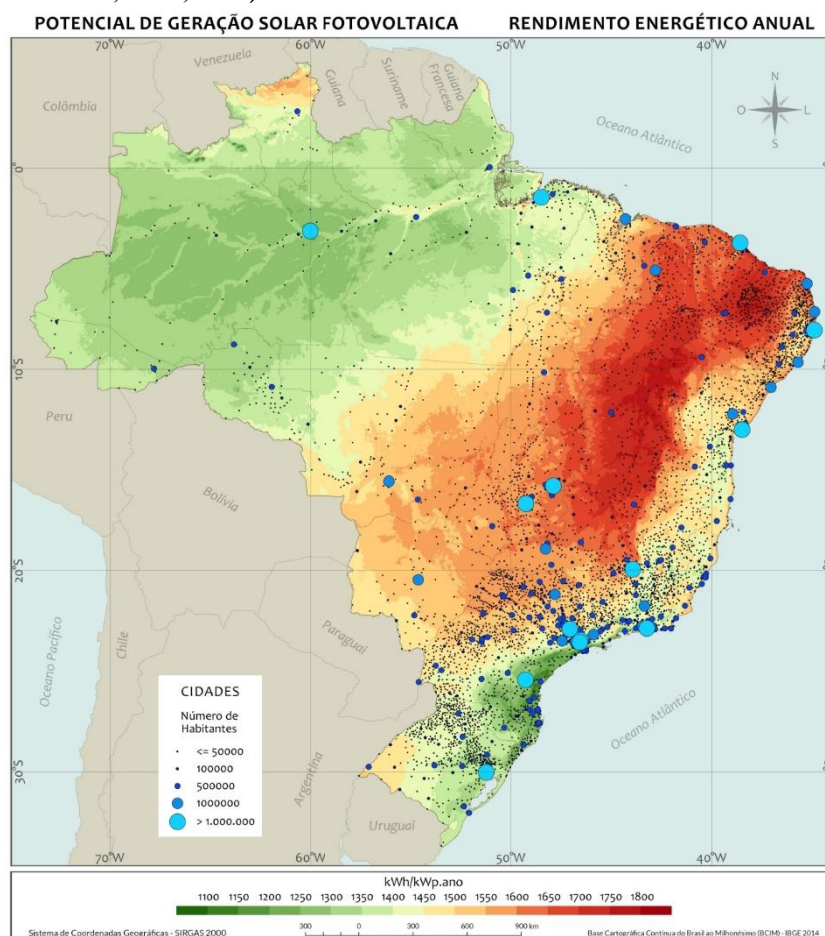
Com essa perspectiva de crescimento no uso de sistema de rastreamento e o consequente aumento da complexidade dos sistemas dos seus gerenciamentos, propõem-se o estudo e desenvolvimento de um sistema de controle SCADA (*Supervisory Control And Data Acquisition*) para sistemas de rastreamento solar *Tracker*. Um sistema SCADA deve possuir a capacidade de supervisionar, controlar e fazer a aquisição dos dados do sistema, que englobam valores de geração de energia, valores de sensoramento e movimentação dos motores. Além disso, os sistemas de controle e supervisão devem contar com uma camada que garanta a segurança e integridade do sistema.

## 1.2 Antecedente do problema

O Brasil é um país de grande área territorial, possuindo muitos climas e características próprias em cada região, possuindo predominantemente um clima tropical, semiárido e subtropical. Por possuir áreas que cortam desde a linha do equador até regiões subtropicais, a geração de energia solar torna-se um problema para as regiões mais afastadas da linha do equador, necessitando de sistemas mais inteligentes e capazes de gerar mais energia com menos irradiação solar. Na Figura 1 podemos ver os níveis de irradiação anuais médios para

irradiação solar normal no território brasileiro. Destacam-se duas regiões no mapa, sendo a primeira região, a região de clima semiárido em vermelho, localizados aproximadamente em 10°S 45°W, que correspondem a zonas ideais para a implementação de usinas de painéis solares fixos ou de um grau de liberdade como a de São Gonçalo e a segunda região localizada aproximadamente a 30°S 54°W, de clima subtropical correspondente ao estado do Rio Grande do Sul que será o ponto estudado.

**Fonte 2: (PEREIRA, E. B., 2017)**



**Figura 1: Mapa do potencial de geração solar fotovoltaica em termos do rendimento energético anual para todo o Brasil (medido em kWh/kWp.ano )**

Devido a grandes extensões territoriais e estando situado em grande parte na região subtropical, o Brasil possui grande potencial de geração de energia solar fazendo uso de sistemas de rastreamento. Com essa perspectiva de crescimento, o uso de sistemas SCADA para monitoramento e supervisão desses sistemas, se torna um ponto importante que deve ser considerado no desenvolvimento dessas tecnologias.

### 1.3 Descrição do problema

Devido ao complexo sistema de rastreamento, que permite o uso de motores de alta potência, inversor de frequência e uma gama de sensores para posicionamento, detecção de condições climáticas adversas, sistemas de controle em diferentes malhas e processamentos de dados, como valores de geração de energia dos painéis solares e consumo de energia por parte do sistema, o sistema *Tracker* se torna um sistema complexo para ser gerenciado. Dessa forma, a fim de viabilizar sistemas de rastreamento *Trackers* mais seguros e que permitam a fácil manutenção do sistema sem a necessidade de um serviço especializado, é necessário que se implemente um sistema SCADA que seja capaz de realizar esse gerenciamento e permita que um operador ou usuário final, esteja no controle do sistema sem que seja necessário um conhecimento técnico especializado.

### 1.4 Objetivo

Tendo em vista a necessidade de se implementar um sistema SCADA em sistemas de geração solar com rastreamento como forma de diminuir a complexidade do sistema, como descrito em tópicos anteriores e estimados os valores de produtividade de seguidores solares de dois graus de liberdade (2 eixos), propõem-se a criação de um sistema SCADA que seja capaz de supervisionar, realizar o controle e gerenciar o sistema como um todo.

O objetivo do presente trabalho não é o desenvolvimento do sistema de rastreamento solar, apenas o desenvolvimento de um *software* que seja capaz de supervisioná-lo. O sistema SCADA desenvolvido deverá ser capaz de se comunicar com o sistema de rastreamento através de um protocolo de comunicação a nível industrial, de forma a garantir a integridade e segurança na troca de informação por parte dos sistemas.

### 1.5 Objetivos específicos

Dada a criação de um serviço SCADA personalizado, os seguintes objetivos específicos que o sistema SCADA terá, podem ser destacados:

- I. Controle de acesso e níveis de permissão ao sistema (*login*);
- II. Capacidade de ser integrado com sistemas *Tracker* que usem sensores e atuadores utilizando o protocolo industrial *Modbus RTU*;

- III. Realizar a supervisão da geração e consumo do sistema;
- IV. Realizar o sensoramento da posição dos motores.

Dessa forma, espera-se desenvolver um *software* SCADA utilizando o protocolo *Modbus RTU* utilizando a linguagem de programação *Python* com o *framework* de desenvolvimento de aplicações *Kivy* para personalização da aplicação. Esse sistema integrará com um sistema *Tracker* previamente desenvolvido ou então um sistema simulado, que possua os parâmetros de comunicação padronizados pelo protocolo *Modbus RTU*.

O aplicativo deverá contar com as funcionalidades de um sistema SCADA e possuir controles de acesso, níveis de controle e as demais funcionalidades de um sistema SCADA.

## 2. REVISÃO BIBLIOGRÁFICA

Os sistemas SCADA (*Supervisory Control and Data Acquisition*) desempenham um papel essencial na automação e monitoramento de processos dentro da indústria. Esses sistemas desempenham o papel de coletar, controlar e analisar dados de uma planta em tempo real, permitindo a supervisão e o controle efetivo de sistemas complexos. Um sistema SCADA é uma solução tecnológica que integra *hardware* e *software* para fornecer controle e monitoramento centralizado de um ou mais processos, permitindo que dados sejam trocados entre os sistemas, além realizar as tarefas de coletar dados em tempo real de dispositivos e sensores distribuídos em uma planta, instalação ou sistema, processa-los e apresenta-los de forma visual e compreensível para os operadores e tomadores de decisão, facilitando o trabalho de supervisão de sistemas complexos.

Nesse capítulo serão discutidos aspectos físicos de um sistema SCADA tais quais suas arquiteturas de *hardware*, comunicação entre sensores e atuadores, sistemas de proteção contra ataques, surtos dos sistemas, funcionalidades e o interfaceamento entre as operações e o operador do sistema, visando uma aplicação voltada um sistema de geração solar.

### 2.1 O que é um sistema SCADA

De acordo com JUNIOR, 2019, um sistema de supervisão e controle consiste na atuação do homem nos processos de produção através e mecanismos confiáveis e que garantam um bom desempenho das ações, a segurança no controle de ambientes de difícil acesso humano e a minimização de falhas, que garante assim a otimização da produção.

Esses sistemas são usados em plantas e processos que possuam medições e controle de atuadores a longas distancias, disponibilizando em uma única central, todas as informações desse processo, para que um ser humano, através desses dados, possa tomar ações de forma remota, reduzindo tempo nas operações.

São características importantes de um sistema SCADA as ações de: Tele comando, tele medição e tele supervisão, que possibilitam que a planta seja monitorada de forma remota, a aquisição dos dados de sensores em tempo real e a datação desses valores em bancos de dados para análises de processo.

Na Figura 2 é mostrado um esquema em formato de pirâmide apresentado por SANTOS, (sd), onde são relacionados a hierarquia entre os dispositivos de campo ou de instrumentação, os dispositivos de controle e dispositivos de supervisão remota.

Fonte 3: SANTOS, sd



**Figura 2: Pirâmide de um sistema SCADA**

Essa relação entre os dispositivos mostra a importância do emprego de um sistema SCADA, uma vez que ele consegue unificar uma grande variedade de informações de sensoriamento em uma central de operações.

Uma das etapas de grande importância em um sistema SCADA está nos dispositivos que estabelecem a comunicação entre os dispositivos de campo e a central de operação. Os dispositivos que são responsáveis pela comunicação entre os sistemas são os Processadores de Fronteira (FEP do inglês *Front End Point*) e os MODENS (Moduladores e Demoduladores).

Os dispositivos responsáveis pela transmissão dos dados trafegados pelas comunicações do sistema são os Modulares, que conseguem condicionar os sinais provenientes dos sensores em sinais elétricos tratados para envia-los à central de comando. Por outro lado, os Demoduladores recebem esses sinais e os condicionam novamente para que os sistemas SCADA possa os utilizar.

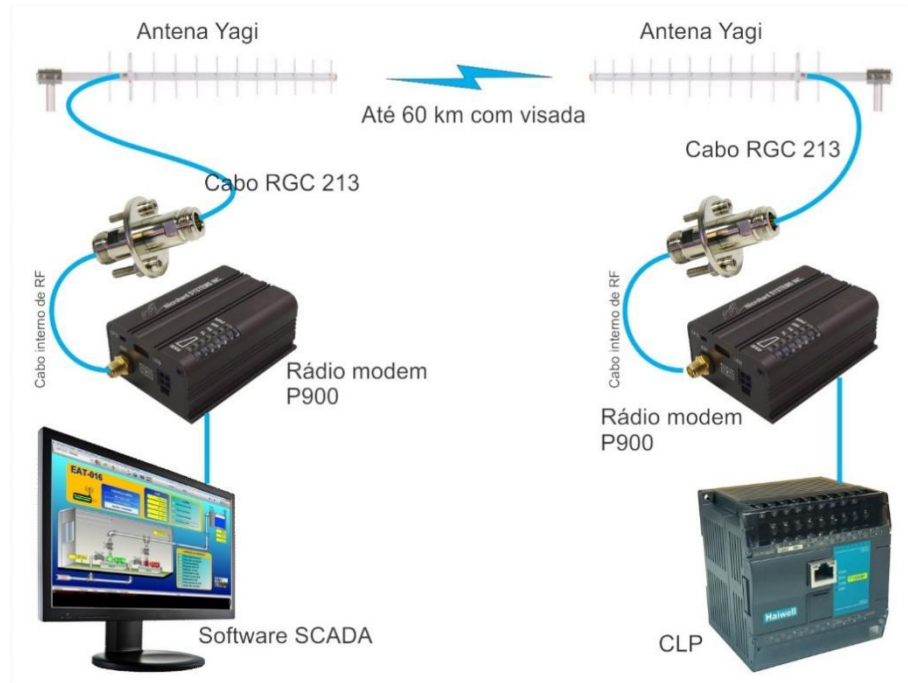
Esses moduladores e demoduladores estão fortemente relacionados ao meio físico que estão inseridos, podendo realizar as transmissões através do ar com o uso de rádio frequências por exemplo, ou em meio elétricos, utilizando protocolos específicos de comunicação, mas não se limitam apenas a esses dois métodos.

Na Figura 3 está representado um exemplo de aplicação dos MODENS onde estabelecem a comunicação entre um CLP e um sistema SCADA. Nessa comunicação, os



sinais são condicionados de um dispositivo para o outro através do uso dos modems, que condicionam os sinais para serem transmitidos via Rádio.

**Fonte 4: Desconhecido.**



**Figura 3: Exemplo de MODEM de comunicação.**

## 2.2 Protocolos de comunicação

De acordo com JUNIOR, (2019), um protocolo de comunicação é um conjunto de procedimentos que servem para controlar e regular uma comunicação, conexão ou transferência de dados entre sistemas computacionais ou de automação.

Queria algumas sugestões de como continuar aqui.

### 3. MATERIAIS E MÉTODOS

Neste tópico serão abordados os materiais e métodos utilizados para o desenvolvimento do sistema SCADA desenvolvido, tais quais linguagens de programação foram utilizadas, *frameworks* para trabalho e desenvolvimento dos *softwares* da aplicação e infraestrutura de processamento dos dados com detalhamento da unidade de processamento da aplicação e interface humano máquina previsto.

#### 3.1 Especificações de *Software* - Python 3.10

O projeto da interface fora desenvolvido inteiramente utilizando a linguagem de programação *Python* na versão “*Python3.10.0*” disponível gratuitamente para *download* em <https://www.python.org/downloads/> sobre a identificação de *download* “tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18” (Figura 4).

Fonte 5: Disponível em: [www.python.org/](http://www.python.org/)



Figura 4: Logo da linguagem de programação Python.

A linguagem de programação *Python* é uma linguagem de alto nível que se destaca por sua simplicidade e legibilidade. Ela foi criada por *Guido van Rossum* e lançada pela primeira vez em 1991, com o objetivo de ser uma linguagem fácil de aprender e usar. Desde então, a linguagem se tornou uma das mais populares e amplamente adotadas em diversos campos, como desenvolvimento *web*, ciência de dados, automação de tarefas, inteligência artificial e recentemente, no desenvolvimento de aplicativos e aplicações *mobile*.

Uma das características distintivas do *Python* é seu modelo de programação que enfatiza a legibilidade do código. A sintaxe limpa e clara da linguagem, juntamente com sua abordagem de indentação significativa, facilita a compreensão do código.

Além disso, *Python* é uma linguagem de programação interpretada, ao contrário de outras linguagens populares como C, o que significa que não requer um processo de compilação antes da execução. Isso permite uma prototipação rápida e uma abordagem iterativa no desenvolvimento de *software*. Além disso, a linguagem suporta múltiplos paradigmas de programação, incluindo programação procedural, orientada a objetos e funcional, fornecendo flexibilidade aos desenvolvedores para escolher a abordagem mais adequada para as suas necessidades.

A popularidade do *Python* tem crescido significativamente nos últimos anos. Apesar de ter sido criada em 1991, somente nos últimos anos ela começou a tomar espaço dentre as linguagens mais famosas, isso se deve em parte à sua comunidade ativa e engajada, que contribui com bibliotecas e *frameworks* de código aberto para desenvolvimento *Python*, suportando a comunicação com protocolos de rede e desenvolvimento de interfaces gráficas.

Por esses motivos, a linguagem de programação foi utilizada para o desenvolvimento da aplicação SCADA, sendo utilizada em todas as etapas do projeto, desde etapas de aquisição de dados do sistema, integração de drivers de rede, até o desenvolvimento gráfico das aplicações através de *frameworks* específicos para criação de aplicações gráficas.

Pela sua versatilidade, a linguagem permite que drivers de comunicação como os do protocolo *Modbus* sejam executados em tempo de execução, gerenciando o tráfego de dados via serial. Além disso, aplicações que usam armazenamento de informações em bancos de dados podem ser desenvolvidas diretamente na linguagem.

Devido às características dessas funcionalidades, serão discutidos individualmente cada *framework* utilizado no desenvolvimento da aplicação.

### **3.2 Comunicação de dados – *Modbus***

O protocolo de comunicação *Modbus* é um protocolo amplamente utilizado na indústria para trocar informações entre dispositivos eletrônicos. Desenvolvido na década de 70, ele se tornou um padrão de fato para a comunicação entre dispositivos de controle e monitoramento em sistemas de automação industrial, estando fortemente presente em indústrias atualmente.

O protocolo de comunicações *Modbus* foi criado inicialmente para ser usada nos CLPs (Controladores Lógicos Programáveis) da *Modicon* (Figura 4).

Fonte 6: Disponível em: <https://www.linkedin.com/pulse/voc%C3%AA-j%C3%A1-ouviu-falar-na-rede-modbus-rodri-go-moreira-borges/?originalSubdomain=pt>



**Figura 5: Primeiro CLP da Modcon - MODICON 084**

Hoje, a empresa *Modicon* foi adquirida pela *Schneider Electric*, resultando em uma mudança significativa no cenário da indústria de automação e controle. Com essa aquisição, o protocolo *Modbus*, utilizado para a comunicação entre dispositivos de controle e automação industrial, alcançou um status de ampla disseminação.

Desde então, a manutenção e desenvolvimento contínuo do protocolo têm sido responsabilidade da entidade conhecida como *Modbus Organization* (Figura 5).

Fonte 7: Disponível em: <https://www.linkedin.com/pulse/voc%C3%AA-j%C3%A1-ouviu-falar-na-rede-modbus-rodri-go-moreira-borges/?originalSubdomain=pt>



**Figura 6: Logo da ModBus Organization.**

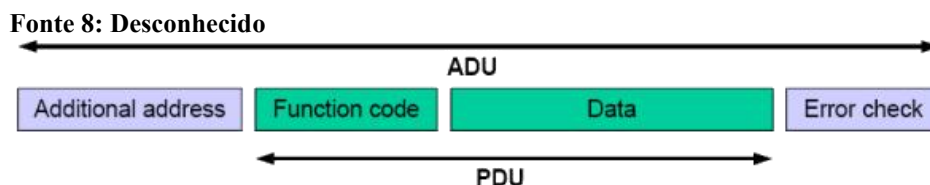
Em abril de 2004, a *Schneider Electric* transferiu os direitos do protocolo para essa organização, que se tornou uma associação de usuários e fabricantes comprometidos em promover e preservar a utilização do protocolo *Modbus*. A *Modbus Organization* desempenha um papel vital na defesa da continuidade do uso do protocolo, garantindo sua atualização e interoperabilidade entre os dispositivos compatíveis. Com seu compromisso em manter o protocolo atualizado e relevante para as necessidades da indústria, a *Modbus Organization* desempenha um papel crucial na promoção da comunicação eficiente e confiável entre dispositivos industriais.

Graças a esses esforços, o protocolo de comunicação *Modbus* se mantém até os dias de hoje, adaptando-se aos propósitos da indústria e conseguindo entregar um sistema robusto e confiável. O padrão passou por diversas modificações e adaptações para usos específicos, existindo 3 protocolos gratuitos disponíveis atualmente, sendo eles:

1. Modbus RTU
2. Modbus ASCII
3. Modbus TCP/IP

### 3.2.1 Estrutura dos pacotes *Modbus*

Todos modelos de comunicação seguem o mesmo propósito e possuem muitas similaridades entre eles. A fim de manter essa compatibilidade, o modelo Modbus divide seus pacotes em ADU (*Application Data Unit*), que diferencia os pacotes pelo tipo dentre os 3 modelos mostrados acima e engloba o PDU (*Protocol Data Unit*). Essa definição fica mais clara na Figura 7.



**Figura 7: Modelo de separação dos campos de dados no protocolo Modbus.**

Essa divisão é feita para permitir a flexibilidade e a compatibilidade do protocolo Modbus em diferentes camadas de comunicação.

O PDU é a parte central da mensagem *Modbus* e contém as informações específicas da função a ser executada. Ele inclui o endereço do dispositivo, a função a ser executada e os dados associados à função. O PDU é independente da camada de transporte ou meio físico de comunicação. Já o ADU é a estrutura completa da mensagem *Modbus*, incluindo o PDU e informações adicionais necessárias para a transmissão e recepção dos dados. Ele encapsula o PDU em uma estrutura de quadro (*frame*) adequada para a camada de transporte.

Na Figura 8 pode-se comparar dois pacotes de dados de dois modelos Modbus, o RTU e TCP/IP:

Fonte 9: Desconhecido

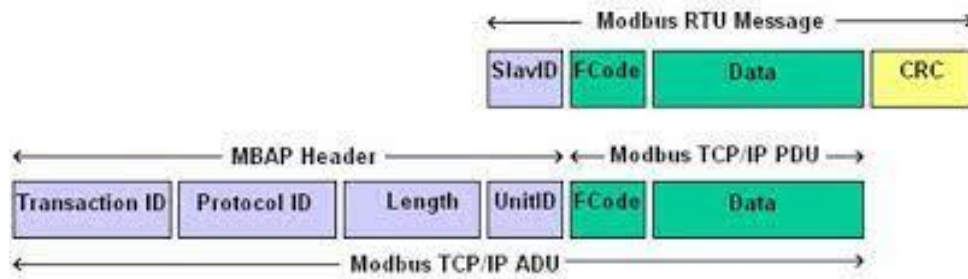


Figura 8: Pacotes Modbus.

Percebe-se na Figura 8 em verde, os campos de dados destinados ao PDU, sendo esse, igual nos dois protocolos RTU/ASCII e TCP/IP. Já em roxo se tem o ADU que diferencia os protocolos. No Modbus RTU, o PDU é encapsulado em um quadro que contém informações de início e parada de bits, bem como informações de controle de fluxo. O quadro é transmitido por meio de uma conexão serial síncrona (como RS-485 ou RS-232). No Modbus ASCII, o PDU é encapsulado em uma estrutura de quadro que usa caracteres ASCII para representar os dados. Ele também contém informações de início e fim de quadro. O quadro é transmitido por meio de uma conexão serial assíncrona. No Modbus TCP/IP, o PDU é encapsulado em pacotes TCP/IP, onde o ADU inclui informações de endereço IP e número de porta para direcionar a mensagem ao dispositivo escravo correto.

### 3.2.2 Modelo de comunicação mestre-escravo

Cada pacote *Modbus* possui claramente as informações de destino e o tipo de solicitação. Essa estrutura permite que o *Modbus* seja utilizado em um modelo de Mestre-Escravo, onde os dispositivos de controle solicitam mensagens aos dispositivos de campo e esses respondem por sua vez de acordo com o tipo de mensagem do mestre (Figura 9).

Fonte 10: Desconhecido.

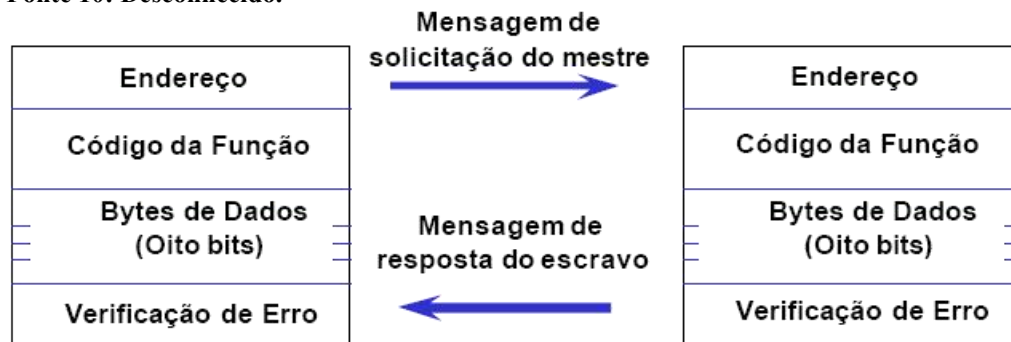


Figura 9: Modelo mestre-escravo *modbus*.

### 3.2.3 Código de função

O FC (Código de Função do inglês *Function Code*) auxilia o escravo a executar a ação requerida pelo mestre. Algumas funções podem ser:

- Endereços dos registradores (registro inicial);
- Quantidade de registros a serem lidos;
- Contador da quantidade de bytes no campo de dados;
- O campo de dados pode não existir. Neste caso o próprio código da função sozinho especifica a ação requerida;
- Se não ocorrer nenhum erro na função especificada na requisição, a resposta do escravo conterá o dado requisitado, caso contrário o campo dado conterá um código de exceção;

Quando respeitada essa requisição, o escravo retorna uma mensagem de resposta também formata de um modo que o mestre entenda. Nesse modelo, o mestre pode ser entendido com um cliente, enquanto o dispositivo de campo é o servidor do sistema (Figura 10).

Fonte 11: Desconhecido.

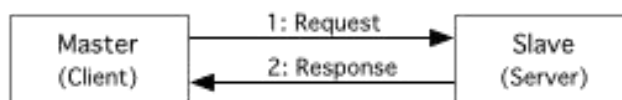


Figura 10: Modelo de requisição e resposta Modbus.

Dentro dos FC existem os códigos de função e os códigos de exceção. Os códigos de função variam do código 1 ao 127, enquanto que exceções variam de 128 a 255. As exceções são tipicamente codificadas em códigos de função FC + 128. Para cada função, ou seja, requisição de serviços do mestre, existe um código de função no Modbus. Assim há 3 tipos de PDUs como ilustra a Figura 11.

Fonte 12: Desconhecido.

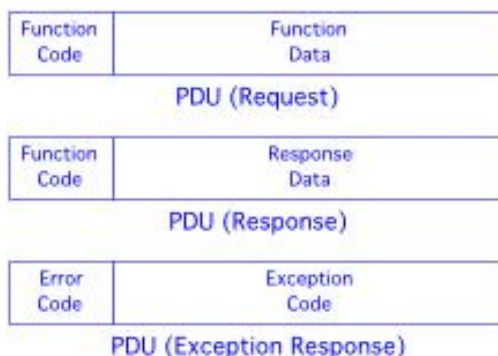


Figura 11: Modelos de PDUs do modelo Modbus.

### 3.2.4 Tipos de registradores

Os registradores *Modbus* são elementos fundamentais do protocolo *Modbus* e são usados para armazenar e acessar dados em dispositivos escravos (servidores) em uma rede *Modbus*. Existem 4 tipos principais de registradores como mostrados na Figura 12.

Fonte 13: Desconhecido.

Tipo de objeto	Acesso	Tamanho	Endereçamento
Coils	Ler e Escrever	1-bit	00001-09999
Discrete input	Só Leitura	1-bit	10001-19999
Input Register	Só Leitura	16-bits	30001-39999
Holding Register	Ler e Escrever	16-bits	40001-49999

Figura 12: Registradores *Modbus*.

1. *Input Registers* (IR): Os *Input Registers* são usados para armazenar dados somente leitura. Eles contêm informações que podem ser lidas pelo dispositivo mestre (cliente), como valores de sensores, estados de dispositivos ou qualquer outro dado que não possa ser alterado pelo mestre.
2. *Holding Registers* (HR): Os *Holding Registers* são usados para armazenar dados que podem ser lidos e escritos pelo dispositivo mestre. Eles podem conter informações como configurações de dispositivos, valores de controle ou qualquer dado que possa ser modificado pelo mestre.
3. *Input Discrete* (ID): Os *Input Discrete* são usados para representar entradas discretas no dispositivo escravo. Eles representam estados lógicos, como sensores binários ou interruptores, e são apenas para leitura.
4. *Coils*: Os *Coils* são semelhantes aos *Input Discrete*, mas podem ser lidos e escritos pelo dispositivo mestre. Eles são frequentemente usados para controlar dispositivos binários, como relés ou válvulas.

Cada registrador é identificado por um endereço único dentro do dispositivo escravo, permitindo que o mestre especifique qual registrador deseja ler ou escrever. Os registradores *Modbus* são organizados em tabelas, com cada tabela correspondendo a um tipo de registrador específico. A quantidade de registradores disponíveis em um dispositivo *Modbus* depende de sua capacidade e configuração específicas. Geralmente, os dispositivos *Modbus* têm uma quantidade limitada de registradores, com tamanhos e endereços pré-definidos.



Devido o fácil acesso desses registradores e organização, se torna fácil administrar um processo, pois os dados estarão armazenados de uma maneira padronizada, promovendo uma interoperabilidade entre dispositivos.

Para a aplicação SCADA em particular, as propriedades da comunicação *Modbus RTU* satisfazem muito bem os requisitos, uma vez que o sistema estará conectado próximo ao sistema e pode ser feito utilizando uma comunicação serial. Além disso, a linguagem de programação *Python* possui bibliotecas criadas especificamente para esse tipo de comunicação como o caso das bibliotecas:

1. ***pymodbus***: É uma biblioteca que implementa os protocolos *Modbus RTU*, *TCP* e *UDP*. Ela fornece uma API simples e abstrata para interagir com dispositivos *Modbus* e suporta tanto a escrita quanto a leitura de registradores *Modbus*.
2. ***minimalmodbus***: É uma biblioteca que oferece suporte a comunicação *Modbus RTU* exclusivamente por meio de uma porta serial. Ela é fácil de usar e fornece métodos para ler e escrever registradores *Modbus*, bem como configurações adicionais, como endereço do dispositivo, paridade, entre outros.
3. ***pyModbusTCP***: É uma biblioteca que implementa o protocolo *Modbus TCP* exclusivamente. Ela permite a comunicação com dispositivos *Modbus* por meio de uma rede *TCP/IP*, oferecendo métodos para ler e escrever registradores *Modbus*.

### 3.3 Interface gráfica - *Kivy*

O *framework* escolhido para o desenvolvimento da interface homem máquina também chamados de GUI (Interface Gráfica de Usuário do inglês *Graphical User Interface*) foi o *kivy*. O *Kivy* é um *framework* de código aberto escrito em *Python* que permite o desenvolvimento de aplicativos multi plataforma com interfaces gráficas voltadas para o usuário. Ele foi projetado para criar aplicativos que podem ser executados em uma variedade de plataformas, incluindo computadores pessoais, dispositivos móveis como *smartphones* e até mesmo plataformas independentes como o caso da *Raspberry Pi*, plataforma usada para o processamento da aplicação que será descrita em tópicos futuros.

O *Kivy* possui uma arquitetura flexível que segue padrões de desenvolvimento de código limpo como o padrão de desenvolvimento MVC (*Model-View-Controller*), permitindo a separação clara entre a lógica de negócios e a apresentação visual da aplicação. Isso facilita a manutenção do código, o reuso de componentes e a escalabilidade do aplicativo, sendo ideal

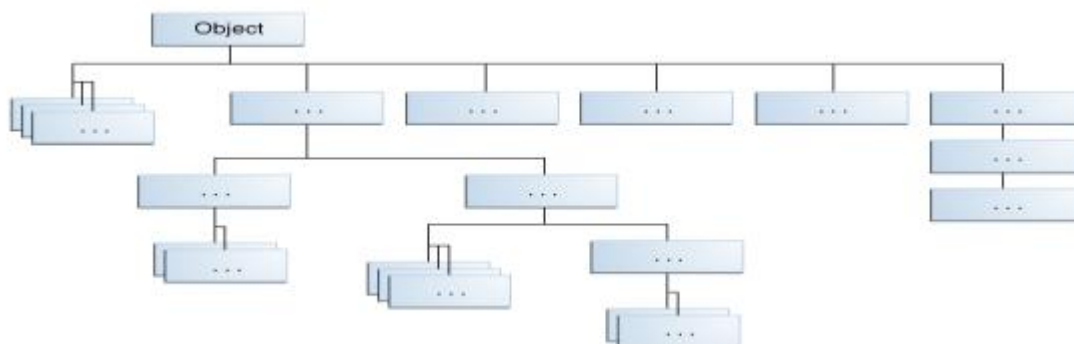
para o desenvolvimento da aplicação SCADA, uma vez que permite separar com clareza quais as regras de negócio entre a aplicação e o sistema supervisionado, permitindo que o programa rode de forma independente, blocos de código relacionados a comunicação com o sistema e blocos de código relacionados a renderização da interface gráfica, tornando o aplicativo mais robusto contra falhas.

Além disso, o *framework Kivy* conta com uma série de expansões de código aberto como o *KivyMD*, uma extensão que adiciona componentes gráficos relacionados ao termo “*Material Design*”, que é um conjunto de diretrizes padronizados pela empresa *Google* para a criação das suas interfaces gráficas. Dessa forma o *KivyMD* permite que as interfaces criadas possuam um estilo agradável, uma vez que padroniza as formas do aplicativo, cores, tipografia e animações em tela.

O modelo de programação em *kivy* permite que sejam criadas interfaces utilizando preceitos muito bem difundidos em outras linguagens de programação utilizadas para a criação de interfaces gráficas, como o *Java* por exemplo. O *Java* é uma linguagem de programação orientada a objetos muito popular na criação de interfaces, isso está ligado ao fato de que dentro de uma interface, existe uma hierarquia dos componentes mostrados em tela e muitos deles são objetos similares, que podem ser condensados em um objeto único, chamado de objeto de interface.

Dessa forma, um objeto de interface pode ser utilizado em diversos lugares com diversas funções, sem perder suas características fundamentais. Além disso, ele pode herdar informações de outros objetos de interface que possuem uma hierarquia maior que a sua, sendo essa uma técnica chamada de herança. Na Figura 4 é possível ver esse modelo de heranças acontecendo.

**Fonte 14: (ASSIS, 2012)**



**Figura 13: Modelo de hierarquia de objetos em interfaces orientadas a objetos.**

### 3.4 Banco de dados - *SQLite3*

Outro ponto fundamental para a aplicação está no desenvolvimento de um banco de dados que irá armazenar as informações do sistema. A necessidade de um banco de dados robusto é fundamental para garantir a integridade dos registros feitos pelos sensores do sistema. Dentro do escopo de programação em *Python*, uma boa solução para implementar um banco de dados é o *SQLite3*, uma biblioteca padrão da linguagem que possui um sistema de gerenciamento de dados leve e autônomo, utilizando de arquivos locais, sem a necessidade de um servidor dedicado para banco de dados.

Dentre as vantagens do *sqlite3* estão:

1. O *SQLite3* já está incluído na instalação padrão do *Python*, portanto, não é necessário instalar bibliotecas adicionais, basta importa-lo para usar.
2. Permite a criação de bancos de dados diretamente em arquivos locais, sem a necessidade de um servidor de banco de dados separado.
3. Suporta transações *ACID* (Atomicidade, Consistência, Isolamento e Durabilidade), garantindo a integridade dos dados mesmo em casos de falhas ou interrupções do sistema.
4. Foi projetado para ser rápido e eficiente, com boa performance mesmo em grandes conjuntos de dados.
5. É compatível com diferentes sistemas operacionais, incluindo *Windows*, *macOS* e *Linux*. Isso torna os aplicativos desenvolvidos com o *SQLite3* portáteis e independentes da plataforma.

Dessa forma é possível se criar tabelas de dados diretamente em código, permitindo que informações do sistema sejam armazenados de forma íntegra e acessíveis para o gerenciamento do sistema e criação de relatórios para avaliações de desempenho e geração.

### 3.5 Decisão

A integração dos módulos *Modbus*, *Kivy* e *SQLite3* para o desenvolvimento do projeto SCADA em *Python* se mostrou uma boa opção devido ao domínio das ferramentas e a facilidade que os módulos tem de conversarem entre si, permitindo que haja um fluxo de informações e modularização do sistema, onde cada módulo é responsável por uma tarefa dentro da aplicação.

O módulo *Modbus* permite a comunicação com dispositivos de automação industrial, permitindo a leitura e escrita de dados nos registradores *Modbus*. Com o *Modbus*, é possível estabelecer a comunicação com controladores programáveis, sensores e atuadores, obtendo informações em tempo real sobre o estado dos equipamentos e realizando ações de controle quando necessário. Além disso, ele possui uma estrutura simples para leitura e escrita dos seus registradores, permitindo uma fácil adaptação do protocolo para o uso na aplicação específica.

O *Kivy*, por sua vez, é um *framework* de interface gráfica de usuário multiplataforma que permite o desenvolvimento de interfaces interativas e intuitivas para aplicativos. Com o *Kivy* e sua extensão *KivyMD*, é possível criar telas, gráficos, botões e outros elementos visuais de forma bonita e agradável ao uso, para visualizar e interagir com os dados do sistema SCADA de forma personalizável.

Por fim, o *SQLite3* é um sistema de gerenciamento de banco de dados embutido que permite armazenar e consultar dados coletados pelo sistema SCADA. Ele oferece recursos para criar tabelas, inserir e atualizar registros, além de executar consultas *SQL* para análise e relatórios. O *SQLite3* é adequado para aplicações SCADA menores e de médio porte, permitindo armazenar dados históricos, configurações e outros dados relevantes para o sistema de supervisão, sendo perfeito para aplicação em específico, uma vez que é extremamente leve e otimizado quando comparado a outros bancos de dados, permitindo que ele seja utilizado em plataformas como *Raspberry Pi*.

Ao integrar esses três módulos em um desenvolvimento SCADA em *Python*, é possível criar um sistema completo para monitorar, controlar e armazenar dados do *Tracker*.

### 3.6 Especificações de *Hardware*

#### 3.6.1 Raspberry Pi 3

A *Raspberry Pi* é uma série de computadores de placa única (do inglês *single-board computer*) desenvolvida pela *Raspberry Pi Foundation*. A *Raspberry Pi Foundation* é uma organização sem fins lucrativos com sede no Reino Unido, cujo objetivo é promover o ensino da ciência da computação e a capacitação tecnológica para pessoas de todas as idades ao redor do mundo.

O *Raspberry Pi B3* por sua vez, é um modelo de computador de placa única que oferece um ambiente de baixo custo, baixo consumo de energia e tamanho compacto. Ele é amplamente utilizado para projetos de automação, incluindo aplicações SCADA devido as suas especificações.

Em termos de *hardware* o *Raspberry Pi 3 model B* tem muito poder de processamento graças ao processador *Broadcom BCM2837* de 64bits e *clock* de 1.2GHz 64-bit *quad-core ARMv8 CPU*, 1 GB de RAM, *Bluetooth 4.1* e possui um controlador *Ethernet* dedicado. Possui quatro portas USB. Além disso, ele permite que seja conectado um HD externo nas suas portas USB, ou então possui uma entrada de cartão SD para armazenamento de dados e armazenamento do sistema operacional.

Além disso, uma das maiores vantagens do *Raspberry pi* é que ele possui saída para conectar em monitores externos através de um cabo HDMI. O Cabo HDMI faz a transmissão de dados em alta resolução, como por exemplo, computadores, televisores e videogames. A transmissão ocorre de forma simultânea para vídeo e áudio em altíssima qualidade.

### 3.6.2 IHM

Para a visualização dos processos e interatividade com a interface gráfica, foi usado um display de 11.6 polegadas de 1920x1080 pixels, compatível com HDMI e tela Led IPS de forma compatível com as especificações de uso com o Raspberry Pi 3B ().

Fonte 15: Disponível em: <https://ae01.alicdn.com/kf/H644aca9d52b5427d89a8513404941302k/11-6-Polegada-1920x1080-Compat-vel-com-HDMI-1080P-Tela-LED-IPS-Monitor-HD-Port-til.jpg>



Figura 14: *Display* utilizado como IHM.

## 4. DESENVOLVIMENTO

No seguinte tópico, será apresentado as técnicas utilizadas durante e o desenvolvimento do projeto de software da interface de controle. Esse tópico estará dividido nas seguintes áreas de desenvolvimento: *Softwares* de controle, Interfaceamento com protocolo *modbus* e sistema físico.

Na área do *software* de controle, será apresentado as técnicas utilizadas para se criar o modelo de supervisão utilizando os frameworks apresentados no tópico 3. Na área de interfaceamento com o protocolo de comunicação Modbus, será apresentado como o software consegue comunicar com os dispositivos *modbus* conectados no sistema. Essa comunicação se dá via Drivers de comunicação. Na ultima área será apresentado brevemente o sistema físico utilizado para o projeto e apresentado aspectos construtivos.

### 4.1 *Software* de controle

Como apresentado nos tópicos anteriores, a interface de controle fora construída inteiramente utilizando a linguagem de programação *Python* e *frameworks* de desenvolvimento tais quais *Kivy* e *KivyMD*. A escolha dessas ferramentas se dá devido a facilidade de se criar sistemas e objetos visuais rodando em uma IHM.

#### 4.1.1 Organização do *Software* de interface

O primeiro ponto levado em consideração na hora de se desenvolver a interface, esteve no modelo que os arquivos iriam estar arquitetados dentro do repositório do projeto. No *KivyMD* existe uma função denominada criação de projeto (*Create\_project* em inglês) que serve para iniciar um repositório organizado dentro de um modelo de escolha. O modelo de arquitetura escolhido foi o MVC (*Model-View-Controller*) (Figura 4) que significa a hierarquia de Modelo, Visão e Controlador, onde:

Fonte 16: Próprio autor.

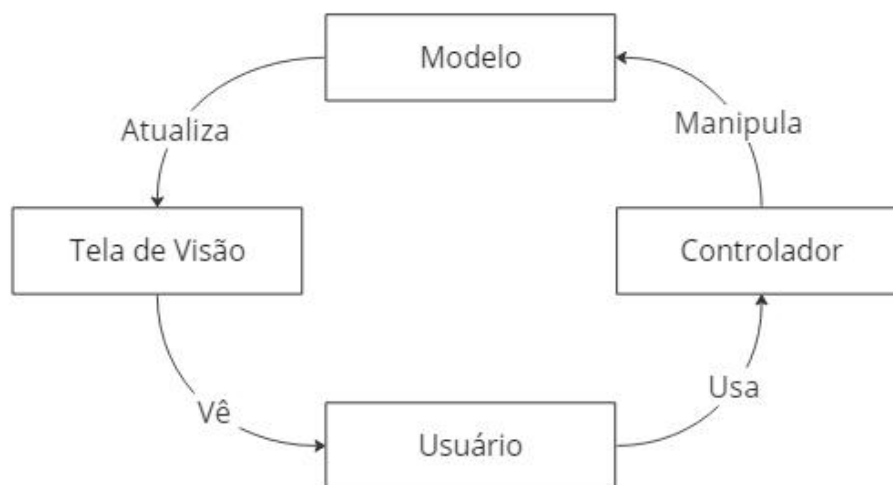


Figura 15: Hierarquia do modelo de organização MVC

**Modelo:** Responsável pela manipulação de dados sensíveis e informações armazenadas em bancos de dados. Em geral, essa parte da aplicação não deve ser acessível diretamente pelo usuário e apenas o controlador deve possuir acesso a ela. É nesse espaço do projeto que todas as manipulações de *login*, cadastro de usuários e manipulações de níveis de acesso são realizadas. O controlador tem o papel de atualizar as informações contidas no Modelo e ele por sua vez tem o papel de atualizar a interface gráfica das mudanças ocorridas por ele.

**Controlador:** É o responsável pelo gerenciamento do projeto como um todo. Todas as manipulações de dados da interface, deverão passar pelo controlador. Ele é a interface entre o modelo e a tela de visão onde o usuário irá ter acesso. Toda interação do usuário com a interface de controle irá passar pelo controlador.

**Tela de visão:** É a interface entre o usuário e o sistema físico. Nessa área todas as definições gráficas são criadas e pensadas para que o usuário do sistema (os operadores) tenha a melhor experiência possível de uso, garantindo que ela seja fácil de entender e usar.

Utilizando dessa arquitetura, fica fácil definir as regras de negócio que devem ser implementadas no sistema, definido as principais funções do sistema e conseguindo realizá-las de forma linear.

#### 4.1.2 Processos internos

Um ponto importante durante a implementação esteve na criação dos processos internos da interface de controle, o chamado *backend* da aplicação. Esses processos estão

intimamente relacionados à área do Controlador dentro do modelo MVC discutido acima. Como toda manipulação da interface corre nos domínios do controlador, esse é responsável pelo gerenciamento da aplicação e possui um papel fundamental dentro da aplicação.

Dentre as responsabilidades do *backend* estão:

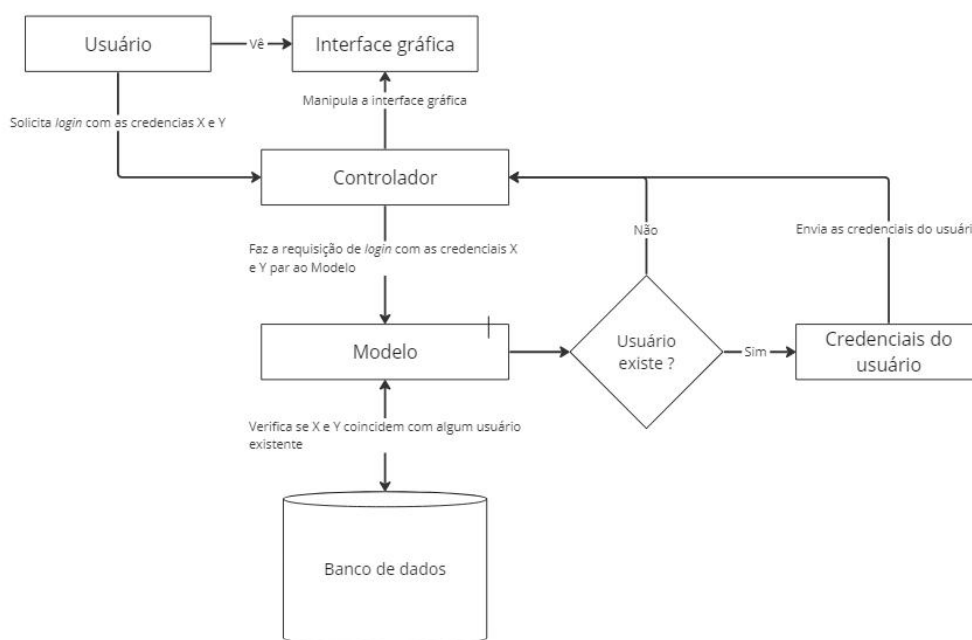
1. Gerenciar *logins* criação de novos usuários;
2. Garantir que usuário sem nível de acesso correto manipule áreas sensíveis.

Nesse ponto é importante diferenciar as tarefas do Modelo às do Controlador, uma vez que o Controlador manda solicitações de login para o Modelo, mas não manipula os dados sensíveis diretamente, sendo esse um processo interno do Modelo.

#### 4.1.2.1 Gerenciamento de logins e criação de usuários

Para o processo de gerenciamento de *logins*, na Figura 6 é possível ver a ordem de requisição realizada pelo Controlador.

**Fonte 17: Próprio autor.**



**Figura 16: Modelo de requisição de login.**

Dessa forma, percebe-se que o Controlador apenas intermedia a tela de interação com o usuário e o banco de dados usado pelo sistema, garantindo que o usuário não manipule dados sensíveis.



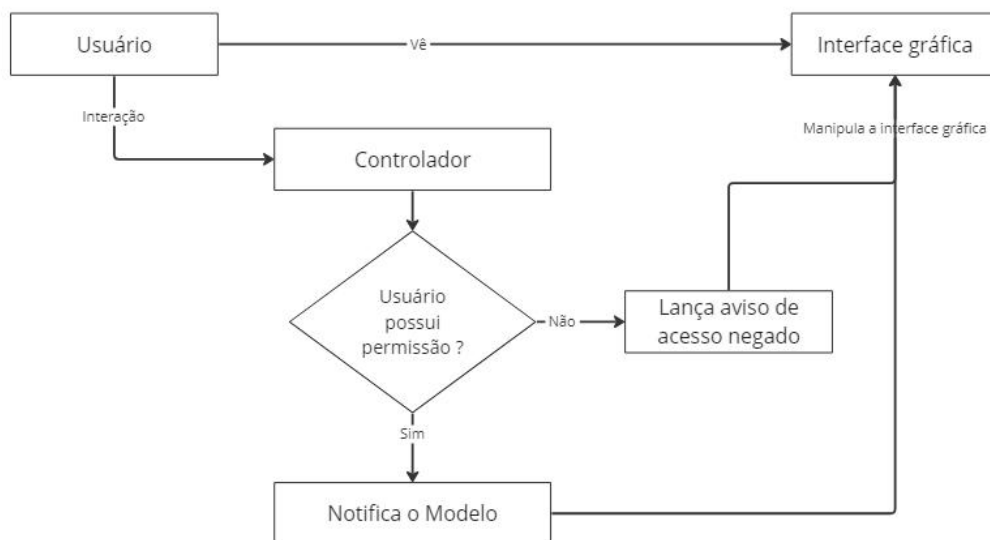
Para o gerenciamento de criação de novos usuários, o fluxograma do processo é semelhante ao processo de *login*, no entanto a solicitação do Controlador ao Modelo é a solicitação de registro.

#### 4.1.2.2 Controle do nível de acesso a usuários

Sem dúvidas, uma das tarefas mais importantes dentro do sistema, é o controle dos níveis de acesso que cada usuário possui, a fim de garantir que somente usuários confiáveis possam manipular as partes sensíveis do sistema, tais quais: Manipulação das rotinas de trabalho do sistema, credenciamento de novos usuários ou manipulação indevida das variáveis de funcionamento do sistema físico.

Dessa forma, toda vez que o usuário da interface realizar uma interação em zonas com proteção de nível de acesso, é papel do controlador verificar se o usuário possui tais permissões ou não para realizar essas ações. A Figura 7 mostra o fluxograma do processo de acesso a regiões do supervisório que possuem proteção de nível de acesso.

**Fonte 18: Próprio autor.**



**Figura 17: Fluxograma do processo de controle do nível de acesso.**

Pode-se perceber que o controlador é o responsável pela verificação do nível de acesso do usuário que executou o *login*, uma vez que possui as credenciais de *login*. No entanto, ele ainda possui a obrigação de notificar o Modelo para toda requisição ocorrida.

### 4.1.3 Modelo

O Modelo representa a lógica de negócios e os dados subjacentes ao aplicativo. Ele contém a lógica para recuperar, armazenar e manipular os dados, bem como implementar as regras de negócios. O Modelo é independente da interface do usuário e da interação do usuário. Ele notifica as outras partes do padrão sobre mudanças em seus dados, geralmente usando o padrão *Observer* (Observador).

No contexto do sistema Tracker, o Modelo desempenha um papel crucial ao lidar com as comunicações com o sistema utilizando o protocolo *Modbus* e estabelecendo conexões de *login* através de sockets TCP/IP com um servidor SQL disponível para a aplicação. Além disso, o Modelo é responsável por gerenciar os níveis de acesso internos do sistema, definido os limites dos usuários.

No que diz respeito às comunicações *Modbus*, o Modelo é encarregado de estabelecer a comunicação com os dispositivos do sistema *Tracker*, enviando e recebendo dados por meio desse protocolo e fazendo o interfaceamento ao meio físico RS-485 que é estabelecido pelo *Modbus RTU*. Ele implementa as funcionalidades necessárias para enviar comandos e receber respostas dos dispositivos, garantindo a integridade e a confiabilidade das comunicações *Modbus*, de forma que o sistema somente seja acessível através dele.

Quanto às conexões de *login* usando *sockets* TCP/IP e o servidor SQL, o Modelo irá gerenciar a autenticação e a autorização dos usuários. Ele verifica as credenciais fornecidas pelos usuários durante o processo de *login*, interage com o servidor SQL para validar as informações e determinar os níveis de acesso concedidos. Além disso, o Modelo mantém um controle interno dos níveis de acesso dos usuários, permitindo ou negando o acesso a recursos e funcionalidades específicas do sistema.

Dessa forma, o Modelo implementa as funcionalidades necessárias para estabelecer as conexões com dispositivos *Modbus*, realizar a autenticação dos usuários e gerenciar os níveis de acesso, garantindo um funcionamento adequado e seguro do sistema como um todo, garantindo que as regiões de acesso a dados sensíveis, sejam protegidas do usuário do sistema supervisor.

Na Figura 18 é possível se verificar o fluxograma dos processos que o Modelo executa e as relações que possui com cada etapa.

Fonte 19: Próprio autor.

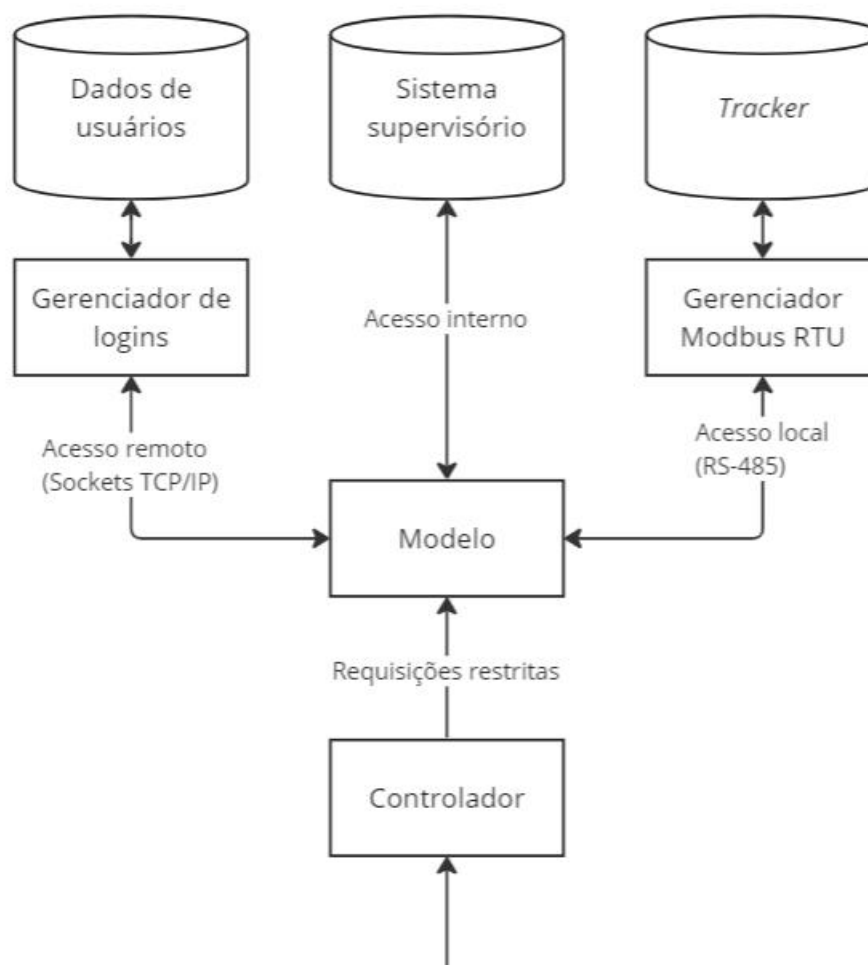


Figura 18: Fluxograma das responsabilidades do Modelo.

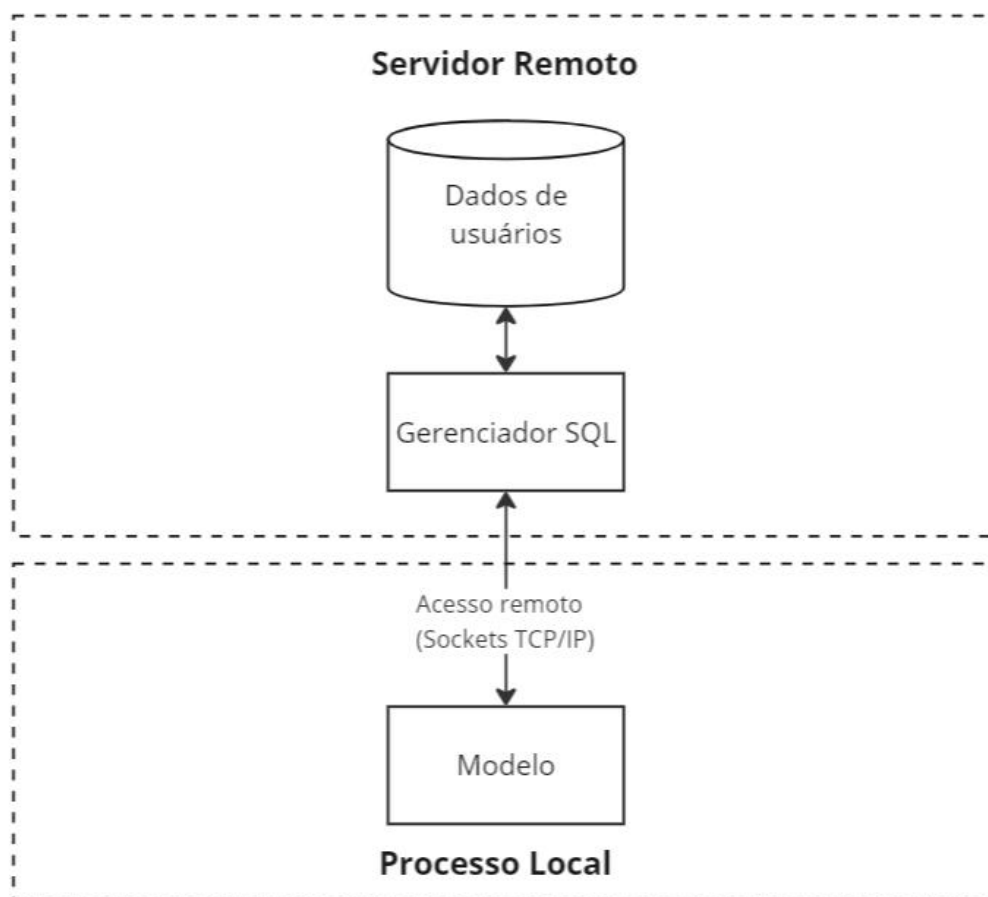
#### 4.1.3.1 Autenticações de *login*

Para realizar o processo de autenticações de *logins*, foi utilizado um serviço de autenticação a qual um servidor remoto é acessado via pacotes *sockets* com o protocolo de comunicação *TCP/IP*, onde pode-se garantir a entrega dos pacotes e estabelecer uma segurança na comunicação, através de métodos de criptografia. Na Figura 19 pode-se ver o fluxo de informações entre o Modelo e o servidor remoto.

Essa relação é utilizada para que se garanta a integridade dos dados armazenados no banco de dados de autenticação, permitindo que o sistema utilize um banco de dados mais robusto e modular, não armazenando os dados localmente. Para acessar os dados, é estabelecido uma padronização dos dados transmitidos, através de um gerenciador de SQL no lado servidor.

Esse gerenciador é responsável por receber as requisições de logins e as validar, além de fornecer as credenciais e níveis de acesso de cada usuário registrado.

**Fonte 20: Próprio autor.**



**Figura 19: Fluxograma relacional entre o Modelo e o gerenciador do servidor remoto.**

A fim de garantir uma segurança e integridade nos dados transitados entre o sistema e o banco de dados, foi aplicado um método de criptografia baseado em chaves simétricas, que criptografa todos os pacotes transmitidos. A criptografia com chaves simétricas é um método de criptografia que utiliza a mesma chave para tanto a criptografia quanto a descryptografia dos dados.

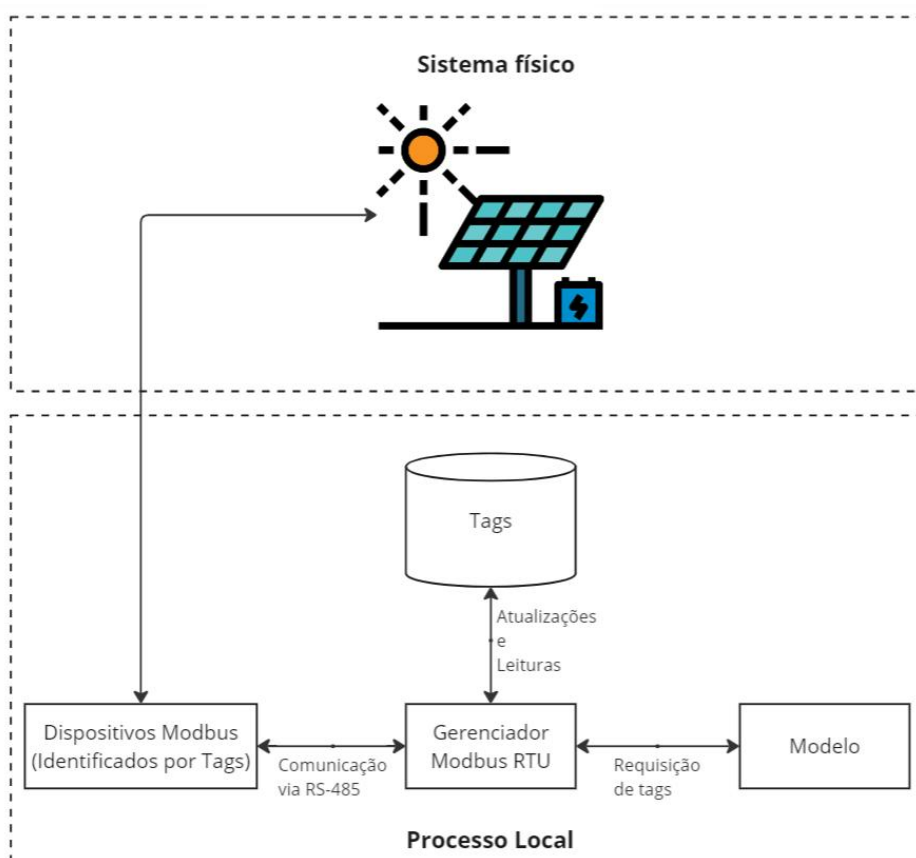
Nesse tipo de criptografia, a mesma chave é usada tanto pelo remetente quanto pelo destinatário para garantir a confidencialidade dos dados. O remetente aplica a chave compartilhada aos dados originais para transformá-los em uma forma ilegível, conhecida como texto cifrado. Em seguida, o texto cifrado é transmitido de forma segura ao destinatário. O destinatário utiliza a mesma chave para aplicar a operação inversa, ou seja, descryptografar o texto cifrado e obter os dados originais.

A principal vantagem da criptografia com chaves simétricas é a sua velocidade e eficiência, pois o processo de criptografia e descryptografia é relativamente rápido. No entanto, a segurança desse método depende da proteção adequada da chave compartilhada. Se a chave for comprometida, a confidencialidade dos dados pode ser comprometida.

De forma simplificada, o sistema supervisor se conecta ao servidor através do envio de uma chave de criptografia que será a chave comum aos dois. Como forma de resposta, o servidor envia uma mensagem padrão de recebimento utilizando a chave de criptografia gerada pela interface. Se o lado cliente, usando a mesma chave de criptografia, consegue descryptografar a mensagem padrão, significa então que cliente e servidor estão conectados e sincronizados entre si.

Para realizar o *login*, o Modelo recebe as credenciais do sistema através do controlador e os envia criptografados para o gerenciador SQL hospedado no lado servidor, se as credenciais estiverem de acordo, então o servidor retorna as mensagens contendo as informações do usuário, tal qual, nome, nível de acesso, fotos, registros e etc.

#### 4.1.3.2 Interface Modbus RTU



#### 4.1.4 Tela gráfica

A parte de visualização do sistema e interação entre operador e sistema é de responsabilidade da Tela de visão, que gerencia todos os desenhos dos objetos na tela e atrela os valores de tags em determinados objetos.

A interface gráfica possui 6 telas únicas que juntas, elas realizam as renderizações de todo sistema supervisorio. São elas:

1. Tela de Login;
2. Tela de Inicio;
3. Tela de Mapa;
4. Tela de conexão Serial;
5. Tela de Sensoriamento;
6. Tela de Diagnósticos.

Essas telas serão descritas com detalhes no tópico 5.

## 5. INTERFACE DE CONTROLE

### 5.1 Tela de *Login*

A primeira tela a ser implementada, é a tela de login, onde é efetuado a identificação do usuário e além disso, é a primeira tela a ser acessada pelo usuário do sistema, portanto, ela deve possuir uma boa impressão ao entrar. Na Figura 20 é possível se visualizar como a tela de login se parece ao ser acessada pela primeira vez.

Fonte 21: Próprio autor.

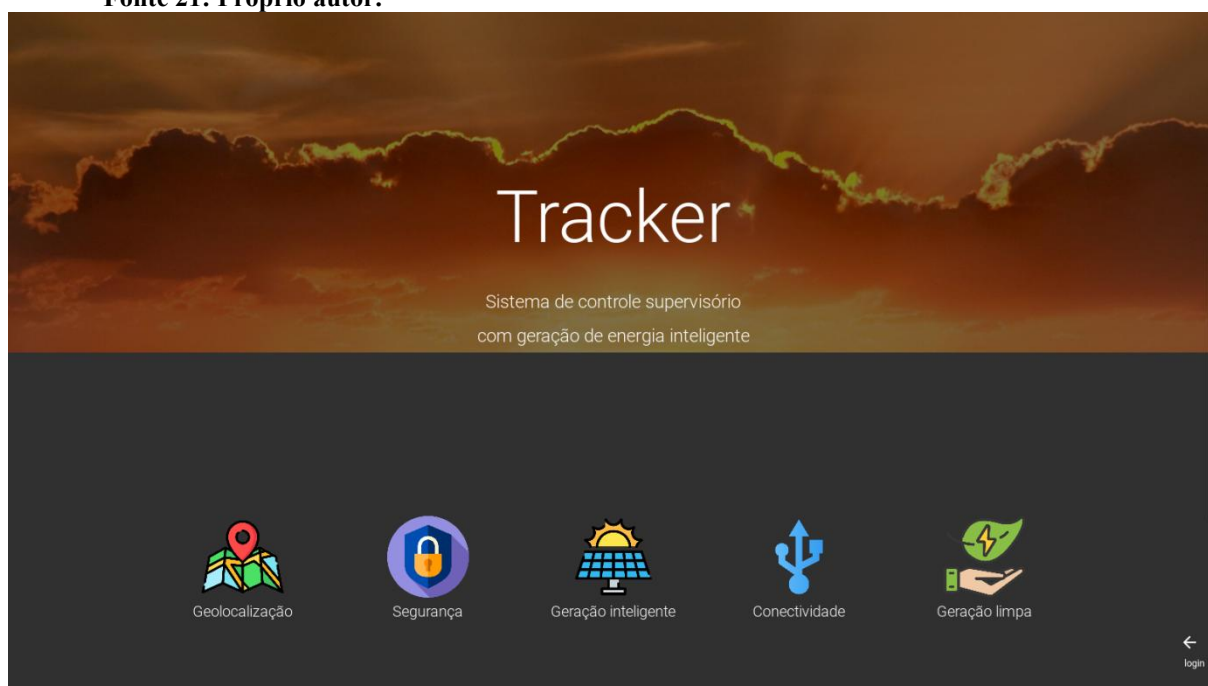


Figura 20: Tela de *login*.

Ela deve ser capaz de transmitir a sensação de que o sistema se trata de um sistema de rastreamento solar e passar uma visão dos benefícios de se utilizar esse sistema. A tela possui uma imagem de fundo que remete ao poder do sol e é possível ver os ícones informativos na parte inferior, que abordam os benefícios do sistema.

Para fins de clareza e com o intuito de não poluir a tela com muitas informações, a secção de login pode ser acessada clicando no botão localizado na borda inferior direita ou arrastando a barra lateral direita da direita para a esquerda. A barra de logins será acessada como mostrado na Figura 20.

Fonte 22: Próprio autor.

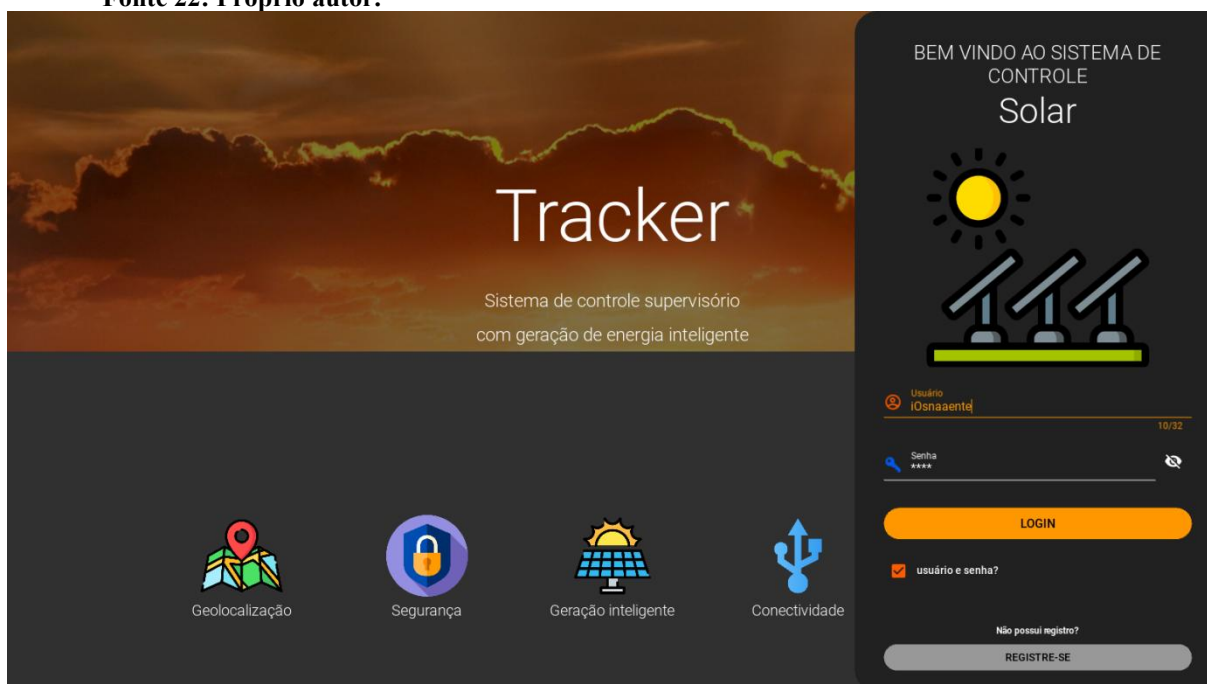


Figura 21: Acesso ao menu de login.

A secção de login possui como finalidade a identificação do usuário do sistema, ao qual deverá dar como entrada as credencias de Usuário e senha de acesso. Além disso, a fim de manter o input mais intuitivo, ele possui sinalizações de campo ativo e permite que a senha seja exposta se necessário (Figura 22).

Fonte 23: Próprio autor.

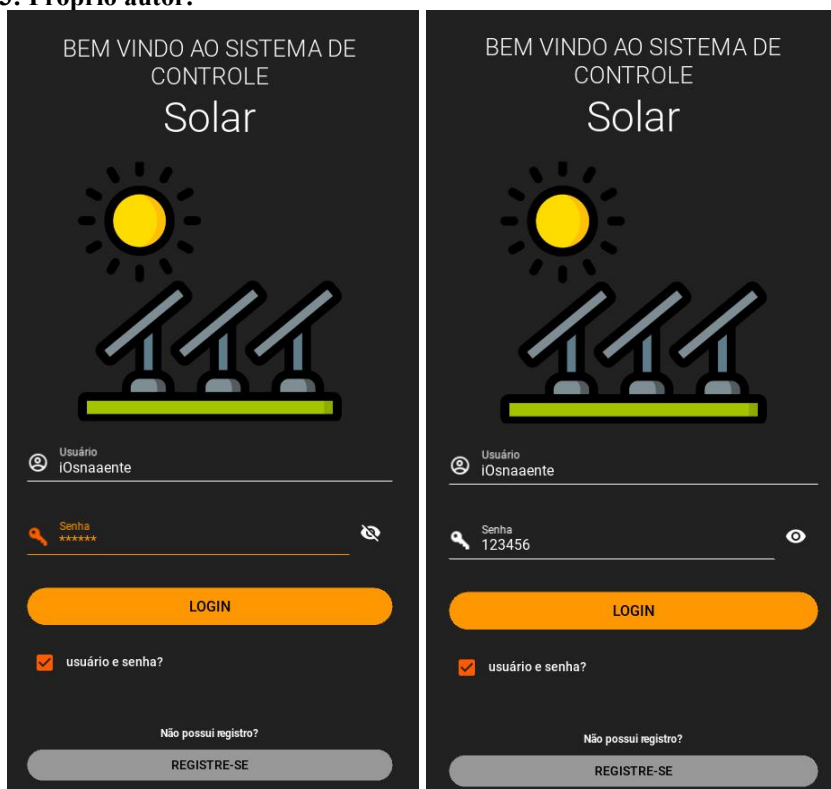


Figura 22: Secção de login.

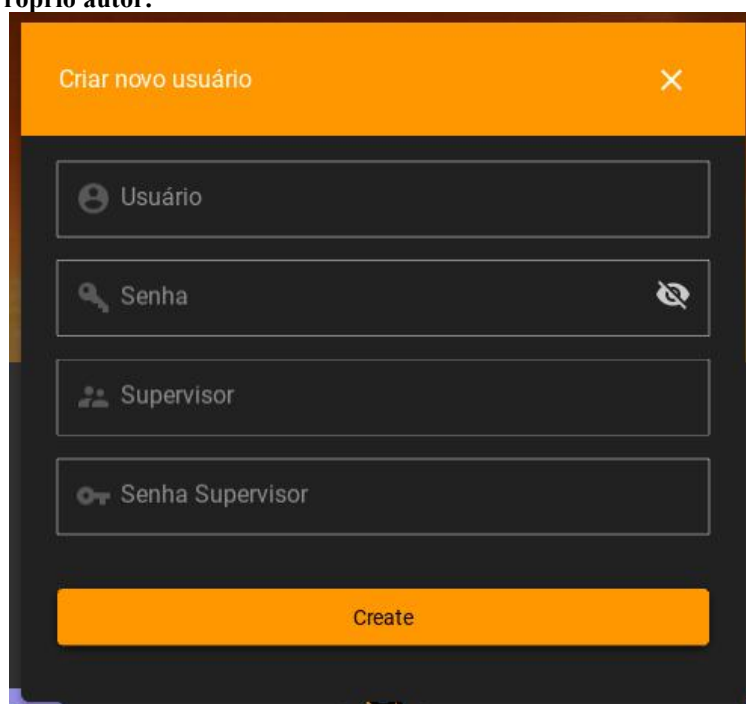


É possível também, se criar um novo usuário, clicando no botão “REGISTRE-SE” o qual será aberto uma pequena janela como mostrado na Figura 23. Nessa janela é possível se ver 4 campos, sendo eles:

1. Usuário: Nome do usuário que será registrado;
2. Senha: Senha do usuário que será registrado;
3. Supervisor: Nome do supervisor ou administrador do sistema;
4. Senha do supervisor: Senha do administrador ou supervisor do sistema.

Esse método de registro permite que seja seguro criar um novo usuário, somente com a permissão de um supervisor ou administrador do sistema, garantindo a segurança e integridade do *Tracker*.

**Fonte 24: Próprio autor.**



**Figura 23: Tela de registro de usuário.**

Outra vantagem do sistema de login é que ele é capaz de lançar mensagens de aviso para erros de autenticação, como:

1. Criação de usuários repetidor;
2. Usuário e senha incorretos;
3. Falta de conexão com o banco de dados.

Na Figura 24 é possível se ver uma mensagem de erro de autenticação ativa.

Fonte 25: Próprio autor.

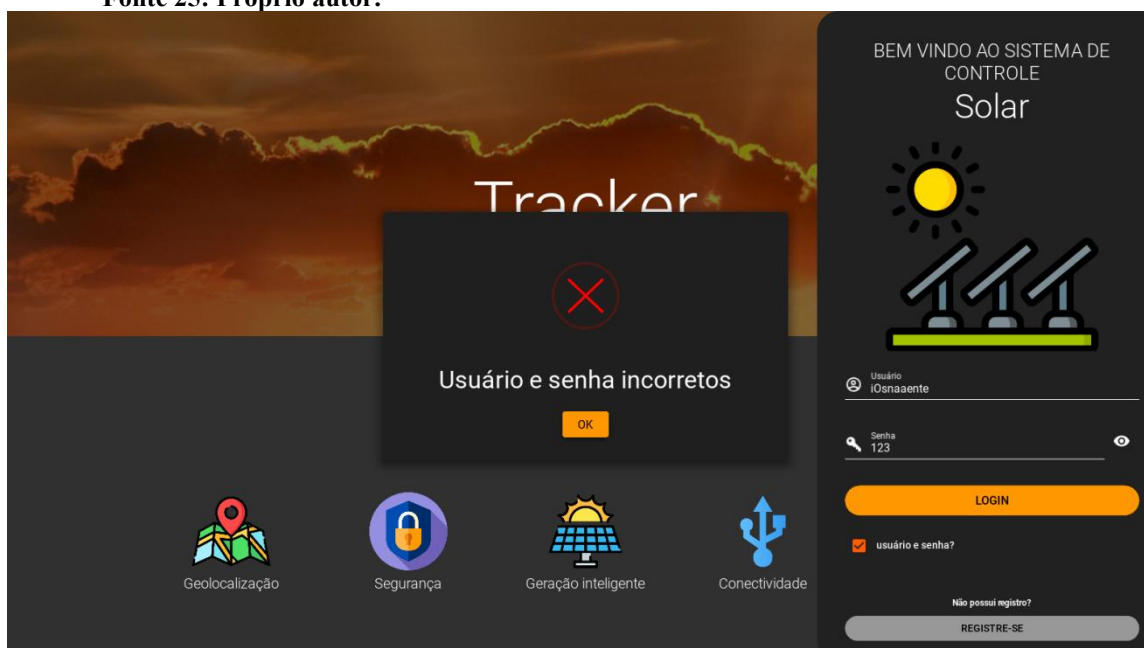


Figura 24: Mensagem de erro de autenticação.

Se autenticado com sucesso, a tela de *login* dará espaço para a nova tela de início que começará a ser renderizada assim que aberta.

## 5.2 Tela de início

A

## 5.3 Tela de mapas

A

## 5.4 Tela de conexão serial

A

## 5.5 Tela de sensoriamento

A

## 5.6 Tela de diagnosticos

A

## **6. RESULTADOS**

### **6.1 Comunicação com o sistema**

a

## **7. DISCUSSÃO**

### **7.1 Interpretação dos resultados**

### **7.2 Implicações teóricas da pesquisa**

### **7.3 Confiança estimada da conclusão**

### **7.4 Restrições de projeto**

### **7.5 Recomendações para pesquisas futuras**

## CONCLUSÃO

Solução ou não do problema abordado

## BIBLIOGRAFIAS

### 7.6 Introdução

COPÉRNICO. N. **Commentariolus**: Pequeno Comentário de Nicolau Copérnico sobre suas próprias hipóteses acerca dos movimentos celestes. Tradução por Roberto de Andrade Martins. 2. ed. São Paulo: Livraria da Física, 2003

DE MELO, Maciel S.: **da Ciência, G., visão de mundo de Nicolau, A., Copérnico, G. G., & Kepler, J. (2006). Uma breve história da Astronomia**. UNIMESP – Centro Universitário Metropolitano de São Paulo - <<http://files.katiafgp.webnode.com/200000276-d5580d5cc0/A%20vis%C3%A3o%20de%20mundo%20de%20Nicolau%20Cop%C3%A9rnico,%20Galileu%20Galilei%20e%20Johannes%20Kepler.pdf>>. Acesso em: 01 set 2021

RICHARDSON, L: **ENERGYSAGE**. mai 2018, Disponível em: <<https://news.energysage.com/the-history-and-invention-of-solar-panel-technology/>>. Acesso em: 01 set 2021.

SILVA, Rutelly Marques da. Energia Solar no Brasil: **dos incentivos aos desafios**. 2015. Disponível em: <<http://www12.senado.gov.br/publicacoes/estudos-legislativos/tipos-de-estudos/textos-para-discussao/td166>>. Acesso em: 01 set. 2021

TOSSATO, Claudemir Roque; MARICONDA, Pablo Rubén: **O método da astronomia segundo Kepler**. Scientiae Studia, v. 8, p. 339-366, 2010.

VALLDOREIX GREENPOWER: **The Benefits of Solar Trackers**. Julho de 2015. Disponível em: <<http://www.valldoreix-gp.com/the-benefits-of-solar-trackers/>>. Acesso em: 01 set 2021.

GREEN POWER: **Parque solar São Gonçalo**. 2021. Disponível em: <<https://www.enelgreenpower.com/pt/nossos-projetos/highlights/parque-solar-sao-goncalo>>. Acesso em: 01 set 2021.

sieBRASIL: **Ministério de minas e energia do brasil – Sistema de Informação de Energias SIE - Capacidade Instalada de Geração Elétrica**: Disponível em:

<[https://www.mme.gov.br/SIEBRASIL/consultas/reporte-dato42-  
jerarquizado.aspx?oc=30181&or=30182&ss=2&v=1](https://www.mme.gov.br/SIEBRASIL/consultas/reporte-dato42-<br/>jerarquizado.aspx?oc=30181&or=30182&ss=2&v=1)>. Acesso em: 01 set 2021.

## 7.7 Revisão bibliográfica

DANEELS, Axel; SALTER, Wayne. **What is SCADA?**. 1999.

SANTOS, Mayara Helena Moreira Nogueira dos. **Sistema de monitoramento integrado baseado nos novos procedimentos de rede do operador nacional do Sistema (ONS) para adequação a subestações digitais**.

JUNIOR, Eraldo Garcia. **Introdução a Sistemas de Supervisão, Controle e Aquisição de Dados: SCADA**. Alta Books Editora, 2019.

### Materiasi e métodos

ASSIS, Daniel Paulo. **INTERFACE GRÁFICA COMO FATOR DETERMINANTE NO DESENVOLVIMENTO DE SOFTWARE**. Instituto Municipal de Ensino Superior de Assis - IMESA e a Fundação Educacional do Município de Assis – FEMA, 2012.

PEREIRA, E. B.; et al: **Atlas brasileiro de energia solar. 2.ed. São José dos Campos: INPE**, 2017. 80p. Disponível em: <http://doi.org/10.34024/978851700089>. Acesso em: 2 nov de 2021.

GOVERNO DO ESTADO DO RIO GRANDE DO SUL. Ministério de Minas e energia. **Atlas solar Rio Grande do Sul**. 2018.

PERAZA, Danielle Goulart. **Estudo de viabilidade da instalação de usinas solares fotovoltaicas no estado do Rio Grande do Sul**. 2013.



KRENZINGER, A.; PRIEB, C. W. M.; GASPARIN, F. P. Mapas de produtividade fotovoltaica para o Rio Grande do Sul (Brasil). In: **CIES2020-XVII Congresso Ibérico e XIII Congresso Ibero-americano de Energia Solar**. LNEG-Laboratório Nacional de Energia e Geologia, 2020. p. 307-314.

FERREIRA; JACOBINA; SANTOS & BARROS. Trigonometria e os raios de sol na terra. **Revista Diálogos** - N.º 11, pág. 145. Jan 2014.

ABSOLAR. **Energia solar no Rio Grande do Sul ultrapassa 400 MW**. 2020. Disponível em: <https://www.absolar.org.br/noticia/energia-solar-no-rio-grande-do-sul-ultrapassa-400-mw/>. Acesso em: 02 set 2021

GARMS, Ibere & CALDAS, Iberê. Síntese das Leis de Kepler. **Rev. Bras. Ensino Fís.** Vol. 40 N° 2. 2018. Disponível em: <https://doi.org/10.1590/1806-9126-RBEF-2017-0253>. Acesso em: 02 set 2021.

NETO. Gastão. **Astronomia de Posição**. Instituto de Astronomia, Geofísica e Ciências Atmosféricas (IAG) - Universidade de São Paulo (USP). Fev 2021. Disponível em: <http://www.astro.iag.usp.br/~gastao/astroposicao.html>. Acesso em: 06 set 2021.

BIOLOGIA TOTAL. **Leis de Kepler: tudo o que você precisa saber**. 2020. Disponível em: <https://blog.biologiatotal.com.br/leis-de-kepler/> . Acesso em 3 set 2021.

MILONE, Andre. **A ASTRONOMIA NO DIA-A-DIA**. 1999.

IBGE. **Noções Básicas de Cartografia**. INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATISTICA - 1999.

DIAS, Wilton S. e PIASSI, Luis. **Por que a variação da distância Terra-Sol não explica as estações do ano?**. Revista Brasileira de Ensino de Física. 2007, v. 29, n. 3. Disponível em: <<https://doi.org/10.1590/S0102-47442007000300003>>. Acesso em: 6 set 2021 , pp. 325-329.

LODI, Cristiane. Perspectivas para a geração de energia elétrica no Brasil utilizando a tecnologia solar térmica concentrada. 2011 **Universidade Federal do Rio de Janeiro/Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia**. Disponível em: [http://objdig.ufrj.br/60/teses/coppe\\_m/CristianeLodi.pdf](http://objdig.ufrj.br/60/teses/coppe_m/CristianeLodi.pdf) . Acesso em: 08 set 2021.

DUFFIE, John A.; BECKMAN, William A. **Solar engineering of thermal processes**. New York: Wiley, 1980.

SILVA, Itã Teodoro da. **Desenvolvimento de um sistema mecatrônico para posicionamento de um painel fotovoltaico e comparação com painel fixo**. UNIVERSIDADE FEDERAL DA BAHIA - Salvador, 2010.

ALVES, Alceu & CAGNON, José & BORDON, M.E.. **AVALIAÇÃO DE DESEMPENHO DE UM SISTEMA DE POSICIONAMENTO AUTOMÁTICO PARA PAINÉIS FOTOVOLTAICOS. ENERGIA NA AGRICULTURA**. (2010). 10.17224/EnergAgric.2010v25n2p01-19.

DANTAS NETO, Pedro Moises et al. **Aumento da eficiência na captação de raios solares na produção de energia elétrica em células fotovoltaicas, por meio de um seguidor solar**. 2018.

PRINSLOO, Gerro; DOBSON, R. T. Solar tracking. **Stellenbosch: SolarBoo7s**. ISBN 978Y0Y620Y61576Y1, p. 1-542, 2015.

MICHALSKY, Joseph J. The astronomical almanac's algorithm for approximate solar position (1950–2050). **Solar energy**, v. 40, n. 3, p. 227-235, 1988.

REDA, Ibrahim; ANDREAS, Afshin. Solar position algorithm for solar radiation applications. **Solar energy**, v. 76, n. 5, p. 577-589, 2004.

ZULKAFI, R. S. et al. Dual axis solar tracking system in Perlis, Malaysia. **Journal of Telecommunication, Electronic and Computer Engineering (JTEC)**, v. 10, n. 1-14, p. 91-94, 2018.

RIZK, J. C. A. Y.; CHAIKO, Y. Solar tracking system: more efficient use of solar panels. **World Academy of Science, Engineering and Technology**, v. 41, n. 2008, p. 313-315, 2008.

LEWANDOSKI, Cristiano Fernando et al. SISTEMA DE RASTREADOR SOLAR DE EIXO SIMPLES BASEADO EM INTENSIDADE SOLAR E DESEMPENHO. **International Journal of Environmental Resilience Research and Science**, v. 4, n. 2, p. 1-11, 2021.

## **ANEXO**

Anexo de projetos e material de apoio sem autoria

## APÊNDICE