FACULDADE DE TECNOLOGIA DE SÃO PAULO

JOSÉ ROBERTO CÂNDIDO DA SILVA

Desenvolvimento de Software Livre em Python Para o Dimensionamento de Sistemas Fotovoltaicos

SÃO PAULO

2021

FACULDADE DE TECNOLOGIA DE SÃO PAULO

JOSÉ ROBERTO CÂNDIDO DA SILVA

Desenvolvimento de Software Livre em Python Para o Dimensionamento de Sistemas Fotovoltaicos

Trabalho submetido como exigência parcial
para a obtenção do Grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas
Orientador: Prof. Dr. Silvio do Lago Pereira

SÃO PAULO

2021

FACULDADE DE TECNOLOGIA DE SÃO PAULO

JOSÉ ROBERTO CÂNDIDO DA SILVA

Desenvolvimento de Software Livre em Python Para o Dimensionamento de Sistemas Fotovoltaicos

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Parecer do Professor Orientador

_____
_____
_____

Conceito/Nota Final: _____

Atesto o conteúdo contido na postagem do ambiente TEAMS pelo aluno e assinada por mim para avaliação do TCC.

Orientador: Prof. Dr. Silvio do Lago Pereira

SÃO PAULO, ____ de _____ de 2021.

Assinatura do Orientador

Assinatura do aluno

Resumo

A utilização da energia solar, tanto como fonte de calor quanto de luz, é hoje uma das alternativas mais promissoras para atender à demanda enérgica global. Entre as diversas aplicações da energia solar podemos citar o uso comercial, residencial, transportes, eletrificação rural, estradas solares, satélites e sistemas autônomos. Os sistemas fotovoltaicos isolados, ou autônomos, (*off grid*) são caracterizados por não se conectar à rede elétrica. O sistema abastece diretamente os aparelhos que utilizarão a energia. Esta solução é bastante utilizada em locais remotos, já que muitas vezes é o modo mais econômico e prático de se obter energia elétrica nestes lugares. O dimensionamento de um sistema fotovoltaico é ajustar corretamente a relação entre a energia radiante recebida do sol pelos módulos fotovoltaicos e a energia elétrica demandada pelas cargas. O dimensionamento adequado do gerador solar e do sistema de armazenamento desempenha um papel importante na confiabilidade operacional de uma fonte de energia fotovoltaica. As diferentes etapas realizadas no dimensionamento de um sistema fotovoltaico exigem determinado grau de conhecimento técnico, sendo geralmente realizada por um especialista, o que contribui para o elevado custo de instalação dos sistemas fotovoltaicos. Alguns softwares de dimensionamento estão disponíveis comercialmente, porém, geralmente seu valor é elevado, ou é exigido conhecimento especializado em sistemas fotovoltaicos para sua operação, tendo como público alvo empresas especializadas ou profissionais do setor. O presente trabalho apresenta os pontos fundamentais da energia solar fotovoltaica e da sua aplicação em sistemas autônomos, mostrando os diversos equipamentos utilizados em um sistema fotovoltaico e as principais técnicas usadas para o seu dimensionamento. Foi desenvolvido um software de uso livre para o dimensionamento de sistemas fotovoltaicos usando a linguagem Python, com o principal objetivo de simplificar a etapa de dimensionamento de sistemas fotovoltaicos, tornando esse tipo de tecnologia mais acessível, principalmente em países em desenvolvimento.

Abstract

The use of solar energy, both as a source of heat and light, is today one of the most promising alternatives to meet the global energy demand. Among the various applications of solar energy we can mention the use in commercial, residential, transport, rural electrification, solar roads, satellites and autonomous systems. Isolated or autonomous (off grid) photovoltaic systems are characterized by not being connected to the power grid. The system supplies the devices that will use the energy directly. This solution is widely used in remote locations, as it is often the most economical and practical way of obtaining electricity in these places. The dimensioning of a photovoltaic system is to correctly adjust the relationship between the radiant energy received from the sun by the photovoltaic modules and the electrical energy demanded by the loads. The proper dimensioning of the solar generator and the storage system plays an important role in the operational reliability of a photovoltaic energy source. The different steps taken in the design of a photovoltaic system require a certain degree of technical knowledge, which is usually carried out by a specialist, which contributes to the high cost of installing photovoltaic systems. Some sizing software is commercially available, however, generally, their value is high, or specialized knowledge in photovoltaic systems is required for their operation, targeting specialized companies or professionals in the sector. This work presents the fundamental points of photovoltaic solar energy and their application in autonomous systems, showing the different equipment used in a photovoltaic system and the main techniques used for its design. Free software was developed for the dimensioning of photovoltaic systems using the Python language, with the main objective of simplifying the step of dimensioning photovoltaic systems, making this type of technology more accessible, especially in developing countries.

Sumário

1 Introdução

Os sistemas fotovoltaicos convertem a energia solar em energia elétrica, fornecendo essa energia diretamente para equipamentos elétricos ou para a rede elétrica.

Dimensionar um sistema fotovoltaico é ajustar corretamente a relação entre a energia radiante, recebida do sol pelos módulos fotovoltaicos, e a energia elétrica demandada pelas cargas. O dimensionamento correto do gerador solar e do bloco de armazenamento desempenha um papel importante na confiabilidade operacional de uma fonte de energia fotovoltaica.

As diferentes etapas de dimensionamento de um sistema fotovoltaico pressupõem um conhecimento técnico especializado, o que contribui para o elevado custo de instalação deste tipo de sistema.

Para fins de comparação, a instalação de um sistema fotovoltaico conectado à rede elétrica, para suprir o consumo médio de energia elétrica na cidade de São Paulo, no ano de 2020, pode variar de R$ 14.246,23 até R$ 17.095,48 [1], valor muito acima do poder aquisitivo para a maioria das famílias de países em desenvolvimento.

Os sistemas fotovoltaicos isolados, ou autônomos, (*off grid*) são caracterizados por não se conectar à rede elétrica. O sistema abastece diretamente os aparelhos que utilizarão a energia. Esta solução é bastante utilizada em locais remotos, já que muitas vezes é o modo mais econômico e prático de se obter energia elétrica nestes lugares. Exemplos de uso são sistemas de bombeamento de água, refrigeradores para armazenamento de vacinas, postes de iluminação, estações replicadoras de sinal, etc. [2]

Mesmo em regiões com pleno abastecimento de energia elétrica, os sistemas isolados podem representar uma economia significativa no consumo de energia elétrica, por exemplo através da iluminação a partir de painéis de LED, alimentados por energia solar, alimentação de sistemas autônomos de monitoramento e sensores utilizados em instalações industriais ou residenciais. A principal desvantagem deste tipo de sistema é a dependência direta da radiação solar incidente para o funcionamento correto dos equipamentos, uma vez que não há conexão com a rede elétrica, necessitando de uma maior precisão no seu dimensionamento.

Existem muitas ferramentas computacionais destinadas ao dimensionamento e otimização de sistemas fotovoltaicos, como por exemplo o PV*SOL®[3] e o PVsyst® [4], porém, estas ferramentas possuem custo elevado, além de possuírem um certo grau de complexidade na sua operação, sendo geralmente destinadas às empresas especializadas na instalação de sistemas fotovoltaicos.

O desenvolvimento de uma ferramenta gratuita, e de fácil utilização para o usuário final, representa um fator que pode contribuir significativamente para o aumento do número de sistemas fotovoltaicos *on grid* e *off grid* instalados nas residências, principalmente em países em desenvolvimento, uma vez que possibilitaria a instalação destes sistemas tanto por eletricistas profissionais, quanto por entusiastas com um conhecimento básico em eletricidade.

A linguagem de programação Python possui diversas bibliotecas e pacotes de uso gratuito. Essas bibliotecas e pacotes destinam-se a uma variedade de soluções modernas, tais como Matplotlib para escrever gráficos e plotagens bidimensionais, Numpy, para processamento de arrays do Python, Opencv, para processamento de imagens, PyQt, para desenvolvimento de interfaces gráficas, Folium, para trabalhos com mapas, entre outras. A grande maioria desses módulos e bibliotecas são de uso gratuito, o que permite a criação e distribuição de soluções diversas por parte de entusiastas e da comunidade Python.

Este trabalho tem como objetivo o desenvolvimento de uma ferramenta para o dimensionamento de sistemas fotovoltaicos *on grid* e *off grid*, a partir da linguagem Python, que seja gratuita e de fácil utilização, mesmo por usuários que não possuem conhecimento especializado em sistemas de energia solar.

2 Energia Solar

2.1 Conceitos Básicos

A energia solar é simplesmente a energia produzida diretamente pelo sol e coletada em outro lugar, normalmente a Terra. O sol cria sua energia através de um processo termonuclear que converte cerca de 650 milhões de toneladas de hidrogênio em hélio a cada segundo. O processo cria calor e radiação eletromagnética. O calor permanece no sol e contribui para a manutenção da reação termonuclear. A radiação eletromagnética (incluindo luz visível, luz infravermelha e radiação ultravioleta) flui para o espaço em todas as direções. [5]

Apenas uma fração muito pequena da radiação total produzida pelo sol atinge a Terra. Essa radiação é a fonte indireta de quase todos os tipos de energia usados hoje. As exceções são a energia geotérmica, a fissão e a fusão nuclear. Mesmo os combustíveis fósseis devem suas origens ao sol; eles já foram plantas vivas e animais cuja vida era dependente do sol. [5]

Devido à natureza da energia solar, dois componentes são necessários para ter um gerador de energia solar funcional. Estes dois componentes são um coletor e uma unidade de armazenamento. O coletor simplesmente coleta a radiação incidente sobre ele e converte uma fração dela para outras formas de energia (seja eletricidade e calor ou apenas calor). A unidade de armazenamento é necessária devido à natureza não-constante da energia solar. Em certos momentos apenas uma quantidade muito pequena de radiação será recebida. À noite ou durante a cobertura de nuvens pesadas, por exemplo, a quantidade de energia produzida pelo coletor será bastante pequena. A unidade de armazenamento pode conter o excesso de energia produzida durante os períodos de produtividade máxima, e liberá-lo quando a produtividade cai. [5]

De forma simplificada, a energia solar geralmente é aproveitada nas formas de energia solar térmica e energia solar fotovoltaica.

2.2 Energia Solar Térmica

No aproveitamento deste tipo de energia, o interesse recai na quantidade de energia que um determinado corpo é capaz de absorver, sob a forma de calor, a partir da radiação solar incidente. A utilização dessa forma de energia implica não somente a captação correta, mas também o armazenamento adequado. Os equipamentos mais difundidos com o objetivo específico de se utilizar a energia solar térmica são conhecidos como coletores solares.

Os coletores solares são aquecedores de fluidos (líquidos ou gasosos) e são classificados em coletores concentradores e coletores planos, em função da existência ou não de dispositivos de concentração da radiação solar. O fluido aquecido pode ser mantido em reservatórios termicamente isolados até o seu uso final. Os coletores concentradores estão associados a aplicações em temperaturas superiores a 100ºC, podendo alcançar temperaturas de até 400ºC, para o acionamento de turbinas a vapor e posterior geração de eletricidade. Já os coletores planos são utilizados fundamentalmente para aplicações residenciais e comerciais em baixa temperatura (por volta de 60ºC), tais como: água aquecida para banho, ar quente para secagem de grãos, aquecimento de piscinas, água aquecida para limpeza em hospitais e hotéis, etc. [6]

2.3 Energia Solar Fotovoltaica

A energia solar fotovoltaica é obtida através da conversão da radiação solar em eletricidade por intermédio de materiais semicondutores. Esse fenômeno é conhecido como efeito fotovoltaico. Enquanto a luz estiver incidindo na célula solar (o nome do elemento fotovoltaico individual), ela gera energia elétrica. Quando a luz deixa de incidir, a eletricidade deixa de ser gerada. As células solares nunca precisam ser recarregadas como uma bateria. Algumas estão em operação contínua, ao ar livre, na Terra ou no espaço, há mais de 30 anos. [4]

A história da energia fotovoltaica remonta ao século XIX. O primeiro dispositivo fotovoltaico funcional, construído intencionalmente, foi feito por Fritts [8] em 1883. Ele derreteu uma camada fina de selênio sobre um substrato metálico e pressionou uma folha de ouro como o contato superior. Este dispositivo possuía quase 30 cm^2 de área. Fritts

observou que, se a corrente gerada não fosse utilizada imediatamente, ela poderia ser armazenada em baterias, ou transportada para utilização em dispositivos distantes do gerador, estabelecendo o princípio de funcionamento dos geradores fotovoltaicos há mais de cem anos.

A era moderna da energia fotovoltaica começou em 1954, quando pesquisadores da Bell Labs nos EUA descobriram acidentalmente que os diodos de junção p-n geravam uma tensão quando as luzes da sala estavam ligadas. A partir deste fenômeno, dentro de um ano, eles produziram células solares de junção p-n de silício com uma eficiência de 6% [9]. No mesmo ano, o grupo da Wright Patterson Air Force Base nos EUA publicou os resultados de uma célula solar de heterojunção de filmes finos baseada em Cu2S / CdS também com 6% de eficiência [10]. Um ano depois, o RCA Labs relatou uma célula solar de GaAs, também com 6% de eficiência nos EUA [11]. No final da década de 1950 e início da década de 1960, vários artigos publicados (Prince [12], Loferski [13], Rappaport e Wysoski [14], Shockley e Queisser [15]) desenvolveram os fundamentos da operação das células solares de junção p-n, incluindo a relação teórica entre bandas de energia, incidência do espectro solar, temperatura, termodinâmica e eficiência. Também foram produzidas células de filmes finos de CdTe com 6% de eficiência [16]. Naquele momento, o programa espacial dos EUA estava utilizando células fotovoltaicas de silício para alimentar satélites. Uma vez que aplicações espaciais eram o principal foco de utilização da energia fotovoltaica, estudos sobre efeitos da radiação e sobre dispositivos tolerantes à radiação, foram feitos usando silício dopado com lítio [17]. Em 1970, um grupo do Instituto Ioffe, liderado por Alferov, na URSS, desenvolveu uma célula solar heterojunção de GaAlAs / GaAs [18] que resolveu um dos principais problemas que afetaram os dispositivos a base de GaAs e apontou o caminho para novas estruturas de dispositivos. As células de GaAs foram de grande interesse devido à sua alta eficiência e à sua resistência às radiações ionizantes no espaço exterior. [7]

O ano de 1973 foi fundamental para a energia fotovoltaica, tanto em áreas técnicas como não técnicas. Uma melhora significativa no desempenho, que ocorreu em 1973 foi a "célula violeta", com uma resposta melhorada para comprimentos de onda curtos, levando a um aumento relativo de 30% na eficiência em células de silício de última geração [19]. Também foram desenvolvidas células de heteroestrutura de GaAs pela IBM, nos EUA, com 13% de eficiência [20]. Ainda em 1973, ocorreu um evento não técnico crucial, chamado de conferência Cherry Hill, que recebeu o nome da cidade de

Nova Jersey, EUA, onde um grupo de pesquisadores e chefes de organizações científicas do governo dos Estados Unidos se reuniram para avaliar o mérito e o potencial científico da energia fotovoltaica. O resultado foi a decisão de que a energia fotovoltaica era digna de apoio do governo, resultando na formação da Agência de Pesquisa e Desenvolvimento de Energia dos EUA, o primeiro grupo governamental do mundo, cuja missão incluiu a promoção de pesquisas sobre energia renovável, que finalmente se tornou o Departamento de Energia dos EUA. [7]

Em outubro de 1973, o primeiro Embargo Mundial do Petróleo foi instituído pelos produtores de petróleo do Golfo Pérsico. Isso repercutiu através do mundo industrializado, e a maioria dos governos iniciou programas para incentivar as energias renováveis, especialmente a energia solar. Vários autores consideram que isso inaugurou a era moderna da energia fotovoltaica e deu um novo senso de urgência à pesquisa e aplicação deste tipo de energia em aplicações terrestres.

Na década de 1980, a indústria começou a amadurecer, à medida que a ênfase na fabricação e os custos cresceram. As instalações de fabricação para a produção de módulos fotovoltaicos das células solares baseadas em lâminas de silício, formando uma junção p-n, foram construídas nos EUA, Japão e Europa. Novas tecnologias começaram a ser desenvolvidas pelo governo, laboratórios universitários e industriais. As empresas tentaram expandir as tecnologias fotovoltaicas de filmes finos, como a-Si e CuInSe2, que alcançaram eficiências superiores a 10%, para dispositivos de área pequena (1 cm^2), desenvolvidos em laboratório. [7]

Os Estados Unidos foram líderes mundiais na produção de tecnologia fotovoltaica durante a maior parte da década de 1990. No final dessa década, políticas de governo na Alemanha e no Japão resultaram em aumentos substanciais no desenvolvimento desse mercado. Essas políticas foram impulsionadas, em parte, por um forte compromisso com a redução de CO2, conforme previsto pelo Protocolo de Kyoto, e em parte para desenvolver o mercado dessa tecnologia para exportação. [6]

Em 1998, a produção mundial de células fotovoltaicas atingiu a marca de 150 MWp (Wp é uma medida de potência energética, normalmente associada com células fotovoltaicas), sendo o silício quase absoluto dentre os materiais utilizados. O grande salto no desenvolvimento do mercado fotovoltaico resultou do rápido aumento da produção chinesa, observado desde 2006. Em 2003, a Ásia não figurava entre os dez

maiores fabricantes do mundo, entretanto, em 2008, três destes eram da China e um de Taiwan e, em 2009, a China já ocupava a liderança na fabricação de módulos fotovoltaicos. [6]

3 Sistemas Fotovoltaicos

Os sistemas fotovoltaicos convertem a energia solar em energia elétrica, fornecendo essa energia diretamente para equipamentos elétricos ou para a rede elétrica.

Um sistema fotovoltaico é constituído por um bloco gerador, um bloco de condicionamento de potência e, opcionalmente, um bloco de armazenamento [6]. O bloco gerador é formado pelo arranjo de módulos fotovoltaicos, em diferentes associações, além do cabeamento elétrico que os interliga e da sua estrutura de suporte. O bloco de condicionamento de potência pode conter conversores c.c.-c.c., seguidor de ponto de potência máxima (SPPM), do inglês *Maximum Power Point Tracking (MPPT)*, inversores, caso o sistema trabalhe com tensão CA, controladores de carga (se houver armazenamento) e quaisquer outros dispositivos de proteção, supervisão e controle. Finalmente, o bloco de armazenamento é constituído por acumuladores elétricos (geralmente baterias). A Figura 1 exemplifica um sistema fotovoltaico genérico.

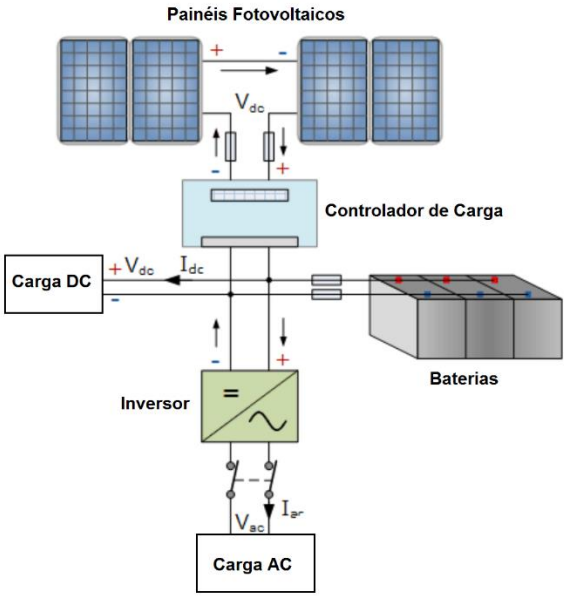

Figura 1 – Sistema fotovoltaico. Fonte: adaptado de http://www.alternative-energy-tutorials.com/solar-power/stand-alone-pv-system.html

3.1 Painéis fotovoltaicos

Os painéis fotovoltaicos são os dispositivos usados para absorver os raios do sol e convertê-los em eletricidade, através do efeito fotovoltaico.

Um painel solar fotovoltaico é na verdade uma coleção de células solares (ou fotovoltaicas), ou seja, materiais semicondutores com características intrínsecas de geração de eletricidade através de luz incidente. Essas células são organizadas em um padrão de grade na superfície dos painéis solares. A Figura 2 apresenta um painel solar de uso residencial.

Figura 2 – Painel solar de uso residencial. Fonte: https://www.thoughtco.com/history-of-solar-cells-1992435

3.2 Bloco de Armazenamento

Os sistemas fotovoltaicos autônomos requerem armazenamento de energia para compensar períodos em que a irradiação solar não é suficiente para alimentar o sistema, ou seja, durante a noite ou em dias nublados. Em todos os casos em que é necessário o armazenamento de energia elétrica, as baterias eletroquímicas são a forma mais conveniente de armazenamento de energia para um sistema fotovoltaico, especialmente devido ao seu funcionamento em corrente contínua, que permite a conexão direta entre o gerador fotovoltaico e a bateria. [21] A Figura 3 Apresenta uma bateria solar da marca Freedom.

Figura 3 - Bateria Solar Freedom. Fonte: https://www.madeiramadeira.com.br/bateria-solar-centrium-energy-df-300-freedom-12v-30ah-1618884.html

3.3 Controlador de Carga

O controlador de carga é um dispositivo eletrônico usado com o objetivo de limitar as cargas que vão do painel solar para a bateria, protegendo a mesma.

Embora os sistemas fotovoltaicos possam ser usados sem controladores de carga, prática que pode ser encontrada muito frequentemente em sistemas fotovoltaicos pequenos, é preciso afirmar que, enquanto se planeja o funcionamento a longo prazo de um sistema fotovoltaico autônomo, é necessário evitar a sobrecarga e a descarga profunda das baterias, tarefa realizada pelo controlador de carga.

Quando a complexidade do sistema aumenta, outros aspectos devem ser considerados na concepção de um sistema fotovoltaico autônomo: com esquemas de gerenciamento de energia adequados, o uso da eletricidade gerada e a vida útil dos componentes sensíveis do sistema podem ser otimizados, por exemplo, desligando cargas de menor prioridade para evitar situações críticas para a bateria. Essas unidades de gerenciamento de energia devem ser projetadas especificamente para a aplicação prevista.

Os controladores devem desconectar o gerador fotovoltaico quando a bateria atingir carga plena e interromper o fornecimento de energia quando o estado de carga da bateria atingir um nível mínimo de segurança. Alguns controladores também monitoram o desempenho do sistema (corrente e tensão de carregamento da bateria ou da carga) e acionam alarmes quando ocorre algum problema. Para melhorar o desempenho do

controlador de carga, este pode ainda incorporar um sensor de temperatura, com a função de compensar o efeito da variação da temperatura nos parâmetros das baterias. [6]

Os controladores mais sofisticados adotam circuitos de seguimento de ponto de potência máxima (*Maximum Power Point Tracker* - MPPT), que são circuitos eletrônicos DC-DC, que permitem que a matriz de células solares forneça a máxima potência durante sua operação, evitando o desperdício de energia, o que reduz o número necessário de células solares do painel fotovoltaico, visando aumentar a eficiência do processo de carga. A Figura 4 Apresenta um controlador de carga do tipo MPTT.

Figura 4 – Controlador de Carga MPTT. Fonte: https://loja.solarproengenharia.com/controlador-de-carga-40a-1224v-mppt-ep-solar-tracer-4210AN

3.4 Inversor

Um inversor é um dispositivo eletrônico que fornece energia elétrica em corrente alternada (AC) a partir de uma fonte de energia elétrica em corrente contínua (DC). A energia DC pode ser proveniente, por exemplo, de baterias, células a combustível ou módulos fotovoltaicos. A tensão AC de saída deve ter amplitude, frequência e conteúdo harmônico adequados às cargas a serem alimentadas. Adicionalmente, no caso de sistemas conectados à rede elétrica, a tensão de saída do inversor deve ser sincronizada com a tensão da rede. [6]

Os inversores são usados em várias configurações de sistemas fotovoltaicos, destacando os sistemas conectados à rede e os sistemas autônomos com baterias recarregáveis.

O planejamento de um sistema fotovoltaico conectado à rede começa com a escolha de um inversor adequado. Isso determina a tensão do sistema no lado DC, e o gerador solar pode então ser configurado de acordo com as características de entrada do inversor solar. O inversor é o segundo componente mais importante de um sistema fotovoltaico conectado à rede (após o gerador solar). Sua tarefa é converter a corrente contínua gerada pelas células solares em corrente alternada de 50 ou 60 Hz, conforme a rede. Em contraste com os inversores destinados apenas à operação autônoma, os destinados à operação paralela devem responder tão bem às características da rede quanto à performance do gerador solar. À medida que toda a corrente, gerada a partir da energia solar, flui através do inversor, suas propriedades afetam fundamentalmente o comportamento e os resultados operacionais do sistema fotovoltaico. A Figura 5 apresenta um inversor de frequência da marca WEG.

Figura 5 – Inversor de frequência WEG. Fonte: https://www.eletrolico.com.br/inversor-de-frequencia-cfw500-p835

3.5 Dimensionamento do Sistema

Dimensionar um sistema fotovoltaico é ajustar corretamente a relação entre a energia radiante, recebida do sol pelos módulos fotovoltaicos, e a demanda de energia elétrica solicitada pelas cargas.

Além da qualidade dos componentes do sistema e da sua construção, o dimensionamento do gerador solar e do bloco de armazenamento desempenham um papel importante na confiabilidade operacional de uma fonte de energia fotovoltaica. As dimensões do gerador solar e a densidade energética da bateria determinam a proporção da demanda de energia do consumidor pela eletricidade gerada, a partir do efeito fotovoltaico. Um sistema fotovoltaico deve ser dimensionado de acordo com algumas etapas de projeto [6, 21]:

1. Definição da localização e configuração do sistema;
2. Levantamento adequado do recurso solar disponível no local da aplicação;
3. Determinação da demanda energética e otimização do consumo.
4. Desenvolvimento do conceito: ajuste do nível de tensão e do tipo de sistema fotovoltaico (DC, AC, DC combinado com AC, conectado à rede, autônomo).
5. Dimensionamento do inversor de frequência, caso o sistema trabalhe com tensões AC.
6. Dimensionamento do gerador solar e do bloco de armazenamento.
7. Dimensionamento do controlador de carga.
8. Dimensionamento dos cabos: as quedas de tensão que ocorrem nos cabos não devem ser ignoradas, particularmente em sistemas maiores operando em um nível de baixa tensão.
9. Determinação da direção e ângulo de inclinação dos painéis fotovoltaicos.

Pode ser necessária a iteração nos passos 3 a 6 várias vezes. Por exemplo, o resultado obtido no passo de dimensionamento 6 pode exigir uma redução adicional no consumo ou uma mudança no conceito do sistema (por exemplo, incorporação de um gerador adicional), por várias razões (área insuficiente para o gerador solar, custo do sistema etc.).

Geralmente, diversas variáveis são consideradas durante a fase de planejamento, que diferem em aspectos, como a confiabilidade do fornecimento de energia, custo, demanda de manutenção e configuração elétrica e estrutural. A prioridade dada a cada aspecto depende da aplicação específica destinada ao fornecimento de energia fotovoltaica. Para

um sistema em que é necessário operar de forma totalmente autônoma, a confiabilidade de fornecimento é decisiva, enquanto essa questão é de importância secundária para um suprimento doméstico, no qual um gerador de backup está incluído de qualquer maneira.

O ângulo de inclinação ideal para o gerador solar depende das condições específicas sob as quais o sistema é operado. Deve ser feita uma clara distinção entre sistemas conectados à rede e autônomos. Os sistemas conectados à rede geralmente são otimizados para alcançar o máximo de rendimento anual possível. Como toda a produção de energia do gerador solar é usada pelo consumo direto ou para alimentar a rede elétrica pública, o rendimento do sistema depende da orientação e ângulo de inclinação dos painéis com relação à radiação solar incidente.

Em sistemas autônomos, a diferença fundamental no comportamento operacional, em relação aos sistemas conectados à rede, ocorre devido ao bloco de armazenamento. Como os sistemas autônomos geralmente têm que alcançar uma determinada fração solar pré-determinada dentro de um determinado período operacional, o tempo decisivo nesse período para o dimensionamento do sistema é aquele em que o nível de radiação é o mais baixo. O ângulo de inclinação ideal depende não apenas da característica de radiação, mas também do próprio sistema.

O método do mês crítico, também chamado de intuitivo, consiste na realização do dimensionamento de um sistema fotovoltaico isolado, considerando o mês do ano com menor incidência de radiação solar em uma determinada região. Como esse método utiliza valores médios mensais de irradiação solar e da carga, considerando somente os valores do mês mais desfavorável na relação carga/irradiação, o sistema produzirá mais energia nos períodos em que as condições forem mais favoráveis, possibilitando o correto funcionamento do sistema durante todo o ano.

3.5.1 Avaliação do Recurso Solar

O termo “radiação solar” é usado de forma genérica e pode ser referenciado em termos de fluxo de potência, quando é especificamente denominado de irradiância solar, ou em termos de energia por unidade de área, denominado, então, de irradiação solar.

Esta etapa do projeto busca quantificar a radiação solar global incidente sobre o painel fotovoltaico em uma determinada região.

Conforme mencionado anteriormente, as características elétricas de um gerador fotovoltaico dependem basicamente da radiação solar incidente e da temperatura nos módulos; porém a influência da radiação solar é muito mais significativa do que a da temperatura, já que a radiação pode sofrer variações abruptas em curtos intervalos de tempo, enquanto que a variação da temperatura é amortecida pela capacidade térmica dos módulos fotovoltaicos.

Considerando o fluxo de radiação extraterrestre como um valor fixo, a variação da distância terra-sol, conduz à variação desse fluxo na faixa de ± 3,3%. Uma equação simples com precisão adequada para a maioria dos cálculos de engenharia é dada por [22]

$$G_{on} = G_{SC}\left(1 + 0.033\ \ cos\frac{360n}{365}\right) \quad (1)$$

onde G_{on} é a radiação extraterrestre incidente no plano normal para a radiação no dia n do ano. O dia do ano (n) deve de ser especificado uma sequência variando de 1 para 1 de janeiro e 365 para 31 de dezembro. G_{SC} é a constante solar, ou seja, quantidade de energia disponível em uma superfície de 1 m² perpendicular aos raios solares na superfície da atmosfera quando a Terra se encontra à uma distância média do Sol, e segundo o Centro Mundial de Monitoramento da Radiação (WRMC - *World Radiation Monitoring Center* é de 1367 W/m² +- 1% [22]. A Figura 6 apresenta a variação da radiação extraterrestre ao longo do ano.

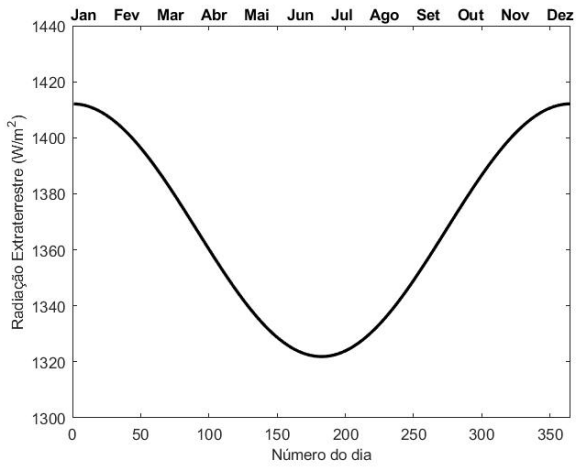

Figura 6 - Variação da radiação extraterrestre ao longo do ano.

A radiação solar é obtida em função de diversos parâmetros, como a altitude, latitude e longitude, dia do ano, hora do dia e inclinação da superfície em relação ao horizonte, sendo necessário adotar modelos matemáticos ou estatísticos para estimar o seu valor.

A declinação δ é a posição angular do sol no meio-dia solar (isto é, quando o sol está no meridiano local) em relação ao plano do equador, essa inclinação, juntamente com o movimento de translação da terra, dá origem às estações do ano. Este ângulo é positivo ao Norte e negativo ao Sul do Equador; -23.45◦ ≤ δ ≤ 23.45◦. Para encontrar a declinação para um determinado dia n do ano usamos a equação aproximada [22]:

$$\delta = 23.45 \sin\left(360 \frac{284+n}{365}\right) \quad (2)$$

Para determinar o dia n do correspondente a cada mês, podemos usar a Tabela I.

Tabela I- Dias Médios Recomendados para Meses e Valores de n por Meses Fonte [22]

Month	n for ith Day of Month	For Average Day of Month		
		Date	n	δ
January	i	17	17	−20.9
February	$31 + i$	16	47	−13.0
March	$59 + i$	16	75	−2.4
April	$90 + i$	15	105	9.4
May	$120 + i$	15	135	18.8
June	$151 + i$	11	162	23.1
July	$181 + i$	17	198	21.2
August	$212 + i$	16	228	13.5
September	$243 + i$	15	258	2.2
October	$273 + i$	15	288	−9.6
November	$304 + i$	14	318	−18.9
December	$334 + i$	10	344	−23.0

O ângulo horário do sol ou hora angular (ω), é deslocamento angular Leste-Oeste do meridiano do Sol, a partir do meridiano local, devido ao movimento de rotação da Terra. Cada hora solar corresponde a um deslocamento de 15°. São adotados, como convenção, valores negativos para o período da manhã, positivos para o período da tarde, e zero ao meio dia solar (momento em que o Sol cruza o meridiano local) [6].

O ângulo horário do pôr-do-sol (ω_s) tem que ser estimado para avaliar a quantidade de horas de luz solar disponível para um determinado dia do ano. Esse ângulo é dado em função da latitude (ϕ) e da declinação (δ), e pode ser calculado por:

$$\omega_s = cos^{-1}(-\tan\phi \tan\delta) \tag{3}$$

e a partir de ω_s o número de horas (N) com disponibilidade de sol são dadas por [22]:

$$N = \frac{2}{15} * \omega_s \tag{4}$$

Uma vez encontrado o ângulo da hora do pôr do sol, a irradiância extraterrestre média diária total (H_O) disponível a partir do sol pode ser estimada utilizando [22]:

$$H_O = \frac{24*1367}{\pi}\left(1 + 0.033\ \cos\frac{360n}{365}\right) * \left(\cos\phi\cos\delta\,\mathrm{sn}\,\omega_s + \frac{\pi\omega_s}{180}\sin\phi\sin\delta\right) \quad (5)$$

Também é interessante calcular a radiação extraterrestre por um período de uma hora. Integrando a Equação (5) por um período entre os ângulos de horas ω_1 e ω_2 que definem uma hora (onde ω_2 é o maior),

$$I_O = \frac{12*1367}{\pi}\left(1 + 0.033\ \cos\frac{360n}{365}\right) * \left(\cos\phi\cos\delta\,(\sin\omega_2 - \sin\omega_1) + \frac{\pi(\omega_2-\omega_1)}{180}\sin\phi\sin\delta\right) \quad (6)$$

Os cálculos realizados até o momento não levam em consideração os efeitos da atmosfera, como a possibilidade de o sol ser encoberto por nuvens, neste caso os dados sobre a Insolação diária média mensal ($\bar{n}$) são a melhor fonte de informação para estimar a radiação incidente média. Estes dados são amplamente disponíveis em muitas centenas de estações em diversos países, e geralmente são baseados em dados obtidos com os instrumentos de Campbell-Stokes (Heliógrafos). Na falta desses dados ou de dados de locais próximos com clima semelhante, é possível usar relações empíricas para estimar a radiação a partir das horas com disponibilidade de sol ou de nebulosidade. Os dados sobre a cobertura de nuvens (nebulosidade) também estão amplamente disponíveis, mas são baseados em estimativas visuais e são provavelmente menos úteis do que os dados da Insolação diária média [22].

A radiação diária média mensal ($\overline{H}$) em uma superfície horizontal, considerando os efeitos da atmosfera, pode ser estimada usando [23]:

$$\overline{H} = H_O * \left(0.16 + 0.87\frac{\bar{n}}{\bar{N}} - 0.61\left(\frac{\bar{n}}{\bar{N}}\right)^2 + 0.34\left(\frac{\bar{n}}{\bar{N}}\right)^3\right) \quad (7)$$

onde $\overline{N}$ é a média mensal diária de horas com a disponibilidade de sol, ou seja, a duração média do dia no mês, calculada como a média do número de horas (N) para cada mês do ano, $\bar{n}$ é o insolação média diária mensal (horas), disponível em muitas centenas de

estações meteorológicas em muitos países. A Figura 7 mostra o valor de $\bar{n}$ para o mês de junho, no Brasil.

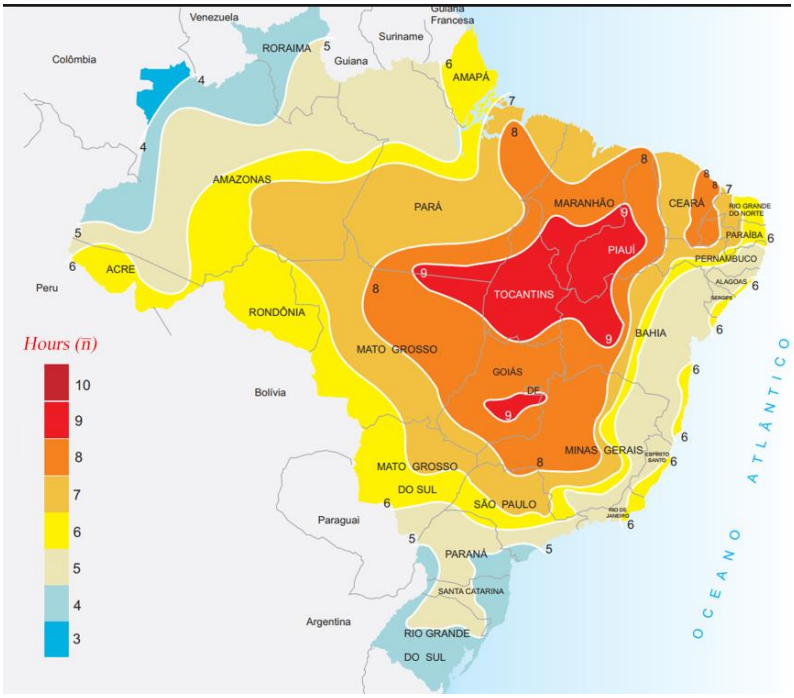

Figura 7 - Insolação média diária mensal para o mês de junho, no Brasil, fonte: Adaptado de [24].

Aqui, propomos o uso de uma fonte de dados que leva em consideração a irradiação horizontal global média diária, fornecida pelo Global Solar Atlas, para obter o valor de $\bar{n}$. Este valor é obtido dividindo a irradiação horizontal global média diária pela potência nominal de $1000\ W\ /\ m^2$. A Figura 8 mostra a irradiação horizontal global média diária na cidade de São Paulo, Brasil.

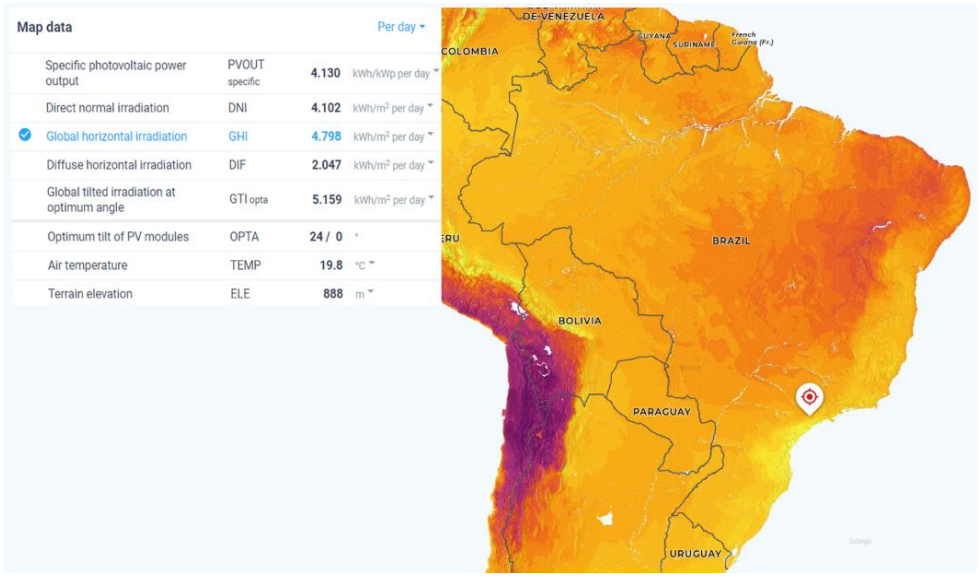

Figura 8 - Dados de irradiação solar para a cidade de São Paulo, Brasil. Fonte: [25]

Para o dimensionamento de sistemas fotovoltaicos conectados à rede, ou para sistemas autônomos não críticos, o dimensionamento levando em consideração a radiação diária média mensal pode ser o suficiente. Porém, para sistemas críticos, é necessário dimensionar o sistema para o dia com a menor incidência de radiação solar, considerando que, se o sistema for capaz de operar no dia mais frio do ano, em um determinado local, será capaz de operar nos demais dias do ano.

A radiação total horária (I_b) por metro quadrado em uma determinada hora, pode ser estimada usando [22]:

$$I_b = \overline{H} * \left(\frac{\pi}{24}(c + d\cos\omega)\frac{\cos\ \omega - \cos\ \omega_s}{\sin\ \omega_s - \frac{\pi\omega_s}{180}\ \cos\omega_s}\right) \quad (8)$$

onde, ω é o ângulo horário para cada hora do dia (h), que pode ser estimado usando:

$$\omega = (15h - 180)\pi/180 \quad (9)$$

para radianos, ou para graus usando:

$$\omega = (Hora\ local - 12) * 15$$

As constantes c e d na Equação (8) são dadas por [22]:

$$c = 0.409 + 0.5016\ \sin(\omega_s - 60)$$

$$d = 0.6609 - 0.4767\ \sin(\omega_s - 60)$$

As radiações para cada hora do dia (0 - 23h) são somadas para encontrar a energia total do sol em um dia, por metro quadrado. Com o valor n variando de 1 até 365 em (2) e (5) e com os devidos valores da relação $\bar{n}/\ \bar{N}$ em (7), são gerados 365 valores da soma de I_b. O menor destes valores corresponde ao dia do ano com a menor incidência da radiação solar, servindo como referência no dimensionamento do sistema.

3.5.2 Levantamento da Demanda e do Consumo de Energia Elétrica

Um sistema fotovoltaico deve ser dimensionado de forma a gerar mais eletricidade do que o limite estabelecido para o consumo. Basicamente é definido um período de tempo e o quanto de energia elétrica é solicitada pelos equipamentos neste período. Assim, a energia elétrica produzida no período estabelecido deve ser maior do que a energia consumida.

Uma maneira tradicional e eficiente para determinar a demanda de energia consumida pelo sistema é somar as respectivas energias para cada equipamento utilizando uma planilha. Esta estimativa pode ser realizada em média semanal, por exemplo, obtendo-se um valor médio de energia elétrica consumida por dia pelo conjunto de equipamentos. A Tabela II apresenta um exemplo de cálculo para três tipos de equipamentos.

Tabela II - Exemplo de cálculo de consumo diário de energia (média semanal). Fonte: [2]

Carga	Potência (W)		Horas de utilização por dia		Dias de utilização por semana		Consumo diário (Wh)	
Equipamento 1	15	x	3	x	4	÷7	=	25,71
Equipamento 2	60	x	2	x	2	÷7	=	34,29
Equipamento 3	100	x	1,5	x	7	÷7	=	150,00
Potência total	**175**	**Consumo diário total**					**=**	**210,00**

Caso o sistema atenda cargas AC, o consumo diário deve ser dividido por um fator decimal representativo da eficiência média do inversor de frequência. Caso o fabricante indique valores de 90% de eficiência, bastante comuns, o consumo deve ser dividido por 0.9 [2].

Para calcular o consumo médio de energia (kWh) de um equipamento de acordo com o seu hábito de uso, procure a potência do aparelho no catálogo ou manual do fabricante e utilize a seguinte expressão [2]:

$$C_m = \frac{P_e.N_d.D_m}{1000} \tag{10}$$

onde:

C_m (kWh/mês) – consumo médio mensal;

P_e (W) – potência nominal do equipamento (dados da placa ou do manual do fabricante);
N_d (h/dia) – número médio de horas diárias de utilização do equipamento;
D_m (dias/mês) – número médio de dias de utilização do equipamento, por mês.

O dimensionamento do sistema de geração em aplicações isoladas pode ser feito totalizando-se o consumo diário de cada equipamento para um dimensionamento simplificado, ou construindo-se, com a maior fidelidade possível, uma curva de carga (Figura 9) para um dimensionamento por meio de um programa de simulação, mais aprofundado. O levantamento preciso da curva de carga, identificando as possíveis

sazonalidades mensais e anuais, pode implicar uma redução importante do custo do sistema e reduzir o risco de falta de energia [2].

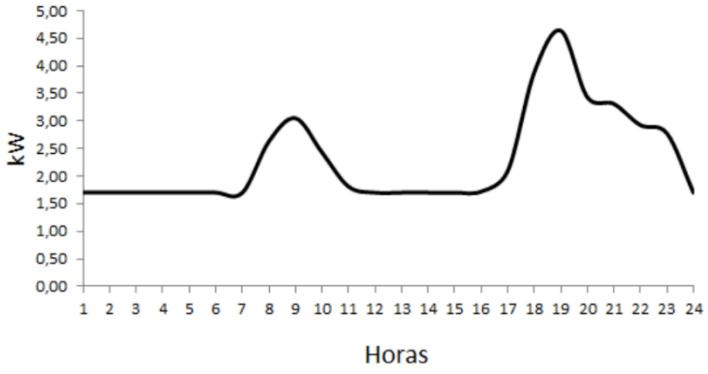

Figura 9– exemplo de curva de carga. Fonte: [2].

3.5.3 Dimensionamento da Unidade de Geração

Para calcular a energia ativa necessária diariamente (L) leva-se em conta o tipo de carga do sistema em corrente alternada e em corrente contínua, se houver, e a eficiência dos elementos que participam do processo de armazenamento e condicionamento de potência, conforme [2]:

$$L = \left(\frac{L_{cc}}{\eta_{bat}}\right) + \left(\frac{L_{ca}}{\eta_{bat}\eta_{inv}}\right) \quad (11)$$

Onde:

L_{cc} (Wh/dia) - quantidade de energia consumida diariamente em corrente contínua em determinado período;

L_{ca} (Wh/dia) - quantidade de energia consumida diariamente em corrente alternada no mesmo período;

η_{bat} (%) - eficiência global da bateria;

η_{inv} (%) - eficiência do inversor de frequência.

Com base na Equação 11, deve ser calculado o valor médio diário de energia requerido para cada um dos meses do ano, e a potência necessária para o painel fotovoltaico, por sua vez, deve ser obtida conforme [2]:

$$P_m = \left(\frac{L}{I_{bTotal} \times 0{,}75 \times 0{,}9}\right) \tag{12}$$

onde I_{bTotal} corresponde à soma de todos os valores de I_b obtidos na Equação (8) em Kw/h, para o dia mais frio do ano. A constante 0.75 corresponde ao fator de redução relacionado às perdas de potência nos módulos fotovoltaicos devido ao acúmulo de sujeira na superfície e à degradação ao longo do tempo, além das perdas devido à temperatura. A constante 0.9 é o fator de redução da potência devido a perdas no sistema, incluindo fiação, controlador, diodos etc.

O número de painéis fotovoltaicos ligados em paralelo para compor o sistema é encontrado através da divisão da potência (Pm) pela potência do painel utilizado. Esse valor deve ser arredondado para maior e, caso o arredondamento seja maior do que 0.5, deve ser escolhido outro modelo, de modo a não tornar o sistema mais caro.

3.5.4 Dimensionamento do Banco de Baterias

A partir da energia solicitada pelas cargas L, Equação (11), é possível calcular a capacidade do sistema de armazenamento segundo as Equações [2]:

$$CB_{C20}(wh) = \frac{L.N}{P_d} \tag{13}$$

$$CBI_{C20}(Ah) = \frac{CB_{C20}}{V_{sist}} \tag{14}$$

onde CB_{C20} é a capacidade do banco de baterias em Wh para o regime de descarga em 20 horas (C20) e CBI_{C20} é a respectiva capacidade em Ah (ampères hora); *N é* o número de dias de autonomia (o qual varia em função da região onde se instala o sistema), tipicamente entre 2 e 4, e não deve ser menor que 2; P_d a máxima profundidade de descarga da bateria, considerando o período de autonomia. Os valores típicos de profundidade de descarga utilizados para baterias de ciclo raso são entre 20 e 40% e, para as de ciclo profundo, de 50 a 80%. A eficiência global da bateria já foi considerada no cálculo de L_m [2].

Na medida em que há maior disponibilidade de radiação solar em um dado local, estabelece-se um número menor de dias para o valor da autonomia, *N*. Já em regiões com longos períodos de chuva, o valor de *N* é maior.

Por vezes o catálogo do fabricante de bateria não apresenta a capacidade C20 e sim em regime C10 ou C100. Neste caso pode-se utilizar as seguintes expressões para conversão [2]:

$$CB_{C20}(wh) = 1{,}1.\, CB_{C10} \tag{15}$$

$$CB_{C20}(wh) = 0{,}9.\, CB_{C100} \tag{16}$$

Após o cálculo da capacidade do sistema de armazenamento, a determinação do número de baterias em paralelo realiza-se pela equação [2]:

$$N°baterias\ paralelo = \frac{CBI}{CBI_{bat}} \tag{17}$$

onde CBI_{bat} representa a capacidade da bateria selecionada, em Ah, no mesmo regime de descarga do valor calculado para *CBI*.

O mesmo critério utilizado para arredondar para maior o número de módulos fotovoltaicos em paralelo e em série pode ser utilizado para a quantidade de baterias. É importante observar que deve ser utilizado o menor número possível de baterias em

paralelo, sendo que os fabricantes recomendam um máximo entre 4 e 6, de forma que se deve adotar modelos de maior capacidade se este número for superado [2].

Já a combinação em série de baterias depende da tensão nominal do sistema.

$$N°baterias\ série = \frac{V_{sist}}{V_{bat}} \tag{18}$$

onde V_{bat} é a tensão nominal da bateria selecionada, em Volts.

3.5.5 Dimensionamento do Controlador de Carga

Com a utilização de um controlador de carga convencional, a determinação do número de módulos em série deve considerar, além da tensão do sistema (V_{sist}), a tensão de máxima potência dos módulos quando estiverem operando na temperatura mais elevada prevista para o módulo em uma dada localidade ($V_{mp}T_{max}$), de acordo com [2]:

$$N°módulos\_série = \frac{1{,}2.V_{sist}}{V_{mp}T_{max}} \tag{19}$$

O valor obtido para o número de módulos em série deve ser arredondado para maior, respeitando a tensão máxima de entrada do controlador de carga. Caso este arredondamento seja superior a 0.5, recomenda-se que seja selecionado outro módulo para compor o gerador fotovoltaico, evitando sobredimensionamento. [2]

O coeficiente 1.2 na Equação 19 considera que um módulo fotovoltaico tem que carregar uma bateria até uma tensão 20% acima da nominal (por exemplo, uma bateria de 12V de Pb-ácido tem uma tensão de carregamento em torno de 14,4 V e de equalização de 14,7 V) e considera, ainda, alguma perda ôhmica. Considerando-se que um módulo de 36 células em climas quentes perde entre 2 e 3 V devido ao aumento da temperatura, é necessário dispor de um módulo que forneça uma tensão nominal de potência máxima, nas condições padrão de teste, de aproximadamente 17 V.

A partir da potência P_m calculada, obtém-se, a seguir, a corrente que deve ser gerada pelo painel fotovoltaico, de acordo com [2]:

$$I_m = \frac{P_m}{V_{sist}} \tag{20}$$

Onde:

I_m (Acc) – corrente do painel fotovoltaico;

P_m (Wp) - potência do painel fotovoltaico (calculada pela Equação 12);

V_{sist} (Vcc) - tensão nominal do sistema (é igual à tensão nominal do banco de baterias), que é igual ao número de baterias conectadas em série vezes a tensão nominal da bateria *Vb*.

O valor de I_m para a corrente é o valor mínimo da corrente no ponto de máxima potência – I_{mp} que o gerador fotovoltaico deve fornecer. Pode-se então calcular o número de módulos a serem conectados em paralelo pela equação [2]:

$$Nºmódulos\_paralelo = \frac{I_m}{I_{mp}} \tag{21}$$

onde I_{mp} representa a corrente de cada módulo, no ponto de máxima potência, nas condições-padrão de ensaio. As mesmas regras para o arredondamento dos módulos em série podem ser utilizadas.

Para sistemas que utilizem controladores de carga com seguimento do ponto de potência máxima (MPPT), o número de módulos em série deve ser tal que a tensão de saída do painel fotovoltaico esteja dentro da faixa ótima de operação do controlador recomendada pelo fabricante [2].

$$\frac{V_{SPPMmin}}{VmpT_{max}} < N°módulos\ série < \frac{V_{SPPMmax}}{V_{mpTmin}} \tag{22}$$

onde $V_{SPPMmax}$ é a máxima tensão de operação e $V_{SPPMmin}$ é a mínima tensão de operação do MPPT do controlador; $VmpT_{max}$ e V_{mpTmin} são as tensões de máxima potência do módulo fotovoltaico nas suas temperaturas de operação máxima e mínima, respectivamente [2].

Para o cálculo do número de fileiras em paralelo, deve-se considerar a potência total do gerador (P_m) e a potência de cada fileira, conforme [2]:

$$I_m = N°módulos\ paralelo.\ I_{mp} \tag{23}$$

onde I_{mp} representa a corrente do módulo no ponto de máxima potência, nas condições-padrão de ensaio.

Para o dimensionamento da corrente máxima do controlador (I_c) é considerada a corrente de curto circuito do painel fotovoltaico acrescida de um fator mínimo de segurança de 25%, assumindo que o painel pode receber uma irradiância de até 1.250 W/m^2 (ainda que por curtos períodos). A corrente de curto circuito do painel é a corrente de curto circuito do módulo (I_{sc}) vezes o número de módulos em paralelo [2].

$$I_c = 1{,}25.\ N°módulos\_paralelo.\ I_{sc} \tag{24}$$

Há modelos de controladores que permitem a operação em paralelo. Isso pode ser necessário se a corrente I_c for elevada para apenas um controlador. Para calcular o número necessário de controladores em paralelo, considerando a corrente máxima do controlador I_{ctl}, usamos [2]:

$$N°controladores\_paralelo = \frac{I_c}{I_{ctl}} \tag{25}$$

A máxima tensão de operação do controlador de carga (V_{cmax}) deve sempre ser maior do que a tensão máxima de saída do painel fotovoltaico.

$$N°módulos\ série.V_{ocTmin} < Vc_{max} \quad (26)$$

onde V_{ocTmin} é a tensão de circuito aberto do módulo, na menor temperatura de operação prevista.

3.5.6 Dimensionamento do Inversor

Uma forma conservadora para o dimensionamento do inversor é considerar a potência do inversor igual ou superior à potência instalada (o somatório da potência de todas as cargas do usuário), se houver grande probabilidade de que estas possam operar simultaneamente.

Para cargas que demandam potência de pico, como motores elétricos durante a partida, é preciso também ter conhecimento dessa potência, juntamente com a respectiva duração, para definir a capacidade de surto necessária no inversor.

O inversor deve apresentar a tensão de entrada igual à tensão DC do sistema (tensão do banco de baterias) e tensão AC de saída conforme a necessidade, normalmente 127 ou 220V, 60Hz. Em geral, inversores até 5 kW são monofásicos. Alguns modelos permitem a operação em paralelo de mais de uma unidade, além de poder ser integrados para criar circuitos bifásicos ou trifásicos. É recomendável a utilização de inversores de forma de onda senoidal, principalmente no caso de cargas eletrônicas que são sensíveis a ondas com distorção harmônica.

Outra condição que dever ser verificada é a compatibilidade entre inversor e controlador, pois alguns modelos não aceitam trabalhar com fabricantes distintos.

3.5.7 Ângulo de Inclinação do painel Fotovoltaico

Um método simplificado, sugerido por fabricantes de módulos fotovoltaicos, para a determinação do melhor ângulo de inclinação para os painéis solares instalados é apresentado na Tabela III.

Tabela III - Ângulo de inclinação recomendado para os painéis solares. Fonte: [26]

Latitude do Local de Instalação	Ângulo de Inclinação Recomendado
0° a 10°	a = 10°
11° a 20°	a = latitude
21° a 30°	a = latitude + 5°
31° a 40°	a = latitude + 10°
41° ou mais	a = latitude + 15°

A orientação recomendada para os painéis é norte, caso a localização de instalação seja no hemisfério sul, e sul caso a localização seja no hemisfério norte.

3.6 Exemplo de aplicação

Dimensionar um sistema fotovoltaico para alimentar uma placa com 48 leds com potência nominal de 2,88 W, funcionando por 24 horas, utilizando painéis solares de 20w e 12v.

Consumo diário:

L=24*2,88 =69,12 =70 Wh.

$$L = \left(\frac{L_{cc}}{\eta_{bat}}\right)$$

Usando uma eficiência da bateria de 0,85.

$$L = \left(\frac{70}{0{,}85}\right)$$

L=82,35

Radiação solar disponível no dia mais frio do ano:

2, 897 Kw.

Como se trata de uma carga fixa usamos:

$P_m = \left(\frac{L}{HSP \times Red_1 \times Red_2}\right)$ ou $P_m = \left(\frac{82,35}{3,10*0,75*0,9}\right)$ =42,11W

Quantidade mínima de painéis:

42,11/20 = 2,11 = 2

Dimensionamento da bateria:

$CB_{C20}(wh) = \frac{L.N}{P_d}$

Bateria de ciclo profundo (60%) e 2 dias de autonomia:

$CB_{C20}(wh) = \frac{42,11*2}{0,6} = 140,36wh$

$CBI_{C20}(Ah) = \frac{CB_{C20}}{V_{sist}}$

$CBI_{C20}(Ah) = \frac{140,36}{12} = 11,69Ah$

Dimensionamento do controlador de carga

$I_c = 1,25.N°módulos_{paralelo}.I_{sc}$

$I_c = 1,25 * 2 * 1,23 = 3,075A$

4 Software Proposto

O Sistema proposto tem como principal objetivo o desenvolvimento de uma interface gráfica com o usuário, de fácil utilização, baseada na biblioteca Pyqt5, do Python. A Figura 10 apresenta o diagrama de blocos, com as entradas e saídas do software. Isso permite uma visão geral de todo o fluxo de dimensionamento de um sistema fotovoltaico, onde as características geográficas afetam o modelo da irradiação solar e a duração do dia, o consumo das cargas e a eficiência dos componentes afetam na escolha dos painéis solares, baterias, inversor de frequência e controlador de carga. O fluxograma do funcionamento é exemplificado na Figura 11.

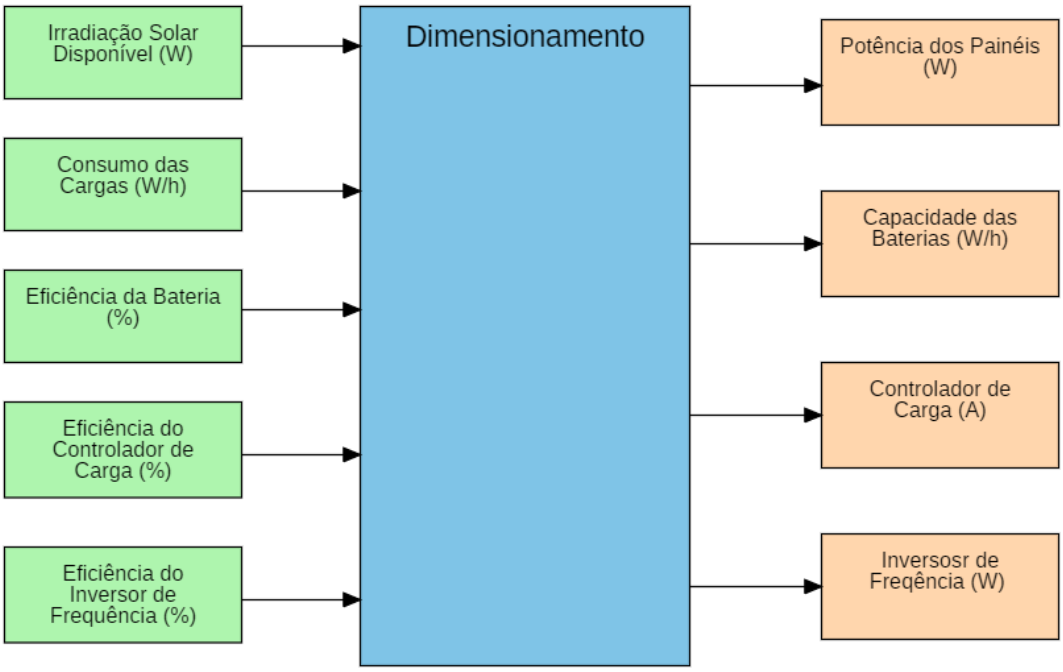

Figura 10 - Diagrama de blocos do software proposto.

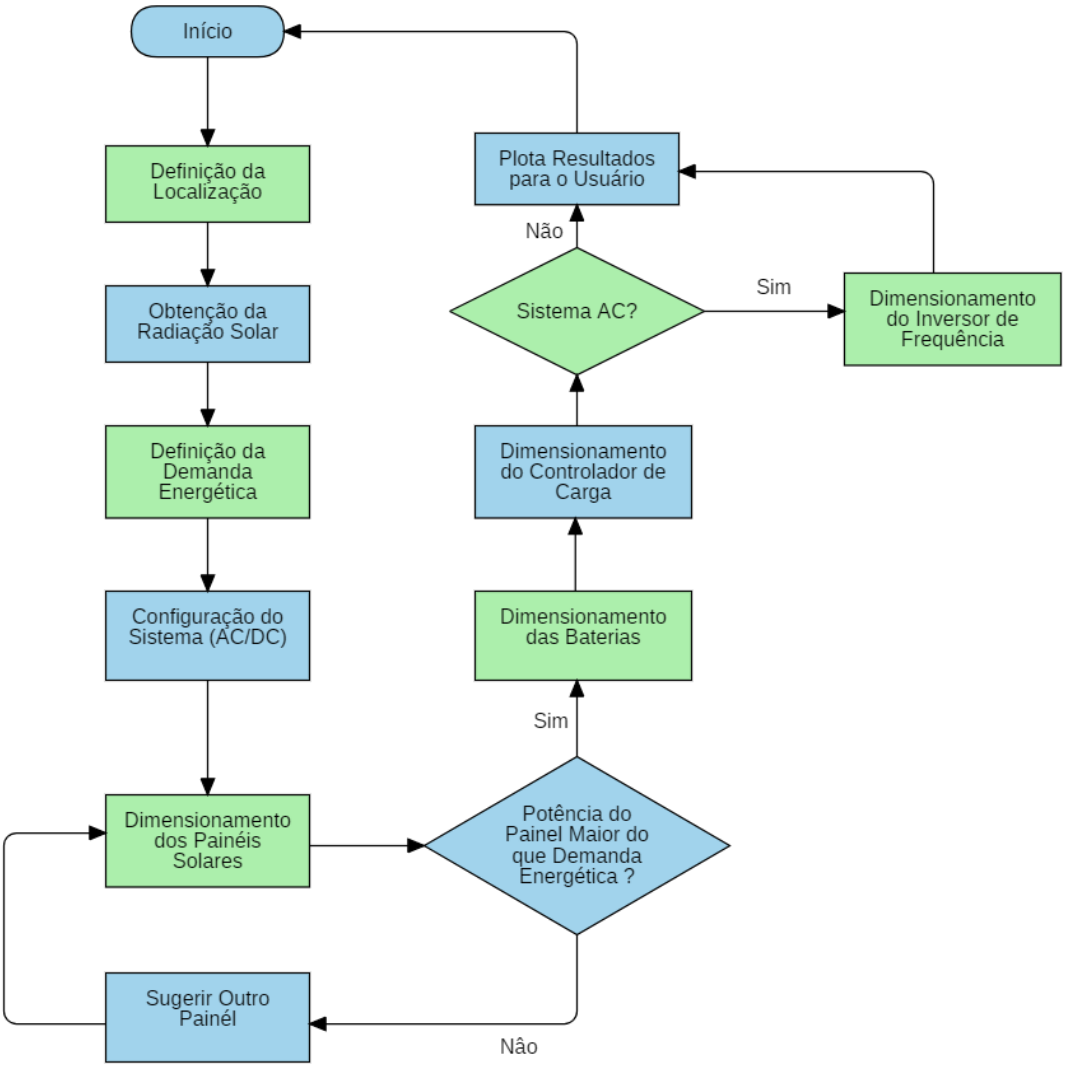

Figura 11 - Fluxograma do software proposto.

4.1 Principais Bibliotecas Python Usadas

- PyQt - Biblioteca para criação de interfaces gráficas com python.
- Folium - Biblioteca para criação, edição e visualização de Maps interativos.
- Matplotlib - Biblioteca para visualização e criação de gráficos.
- Numpy - Biblioteca usada principalmente para realizar cálculos em Arrays Multidimensionais.
- cx_Freeze – Biblioteca para criação de executáveis em python.

4.2 Link Para Download

O software está disponível para download em uma versão prévia para testes, no seguinte repositório:

https://github.com/Roberto-Candido/TCC-FATEC

5 Resultados

Para fins de comparação, foi proposto um sistema fotovoltaico isolado da rede para uma residência localizada na cidade de São Paulo. O cálculo da demanda energética foi baseado em um sistema de iluminação de 12V, utilizando painéis de LED com 3,0 W de potência nominal, de acordo com a distribuição dada na Tabela IV, totalizando um o consumo diário total de energia de 108 W. Todos os valores simulados no sistema proposto foram comparados com o software de dimensionamento de sistemas solares PVsyst.

Tabela IV - Estimativa Da Demanda Energética Em Uma Residência.

Ambiente	**Quantidade**	**Potência nominal (W)**	**Horas de uso/dia**	**Consumo diário Wh**
Varanda	1	3,0	4	12
sala	1	3,0	4	12
cozinha	1	3,0	4	12
Quarto 1	1	3,0	4	12
Quarto 2	1	3,0	4	12
Quarto 3	1	3,0	4	12
Quarto 4	1	3,0	4	12
Banheiro	1	3,0	4	12
Banheiro	1	3,0	4	12

O primeiro passo no dimensionamento do sistema solar é a definição da localização, conforme a Figura 12. A Figura 13 apresenta a etapa de definição da localização do software PVsyst.

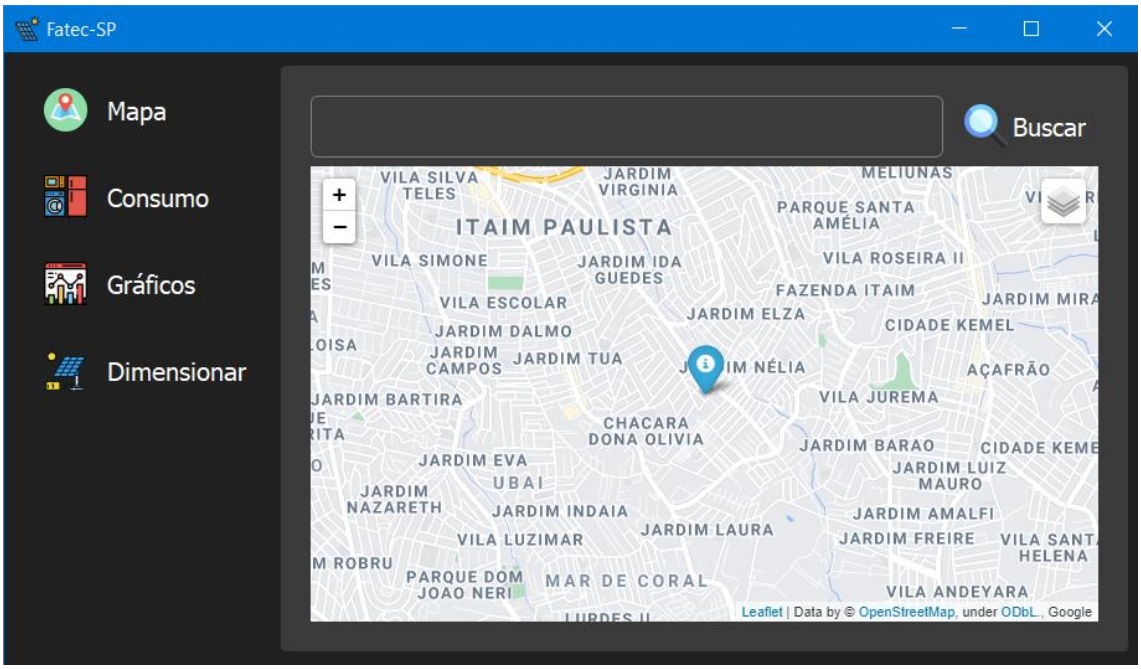

Figura 12 – Definição da localização no sistema proposto.

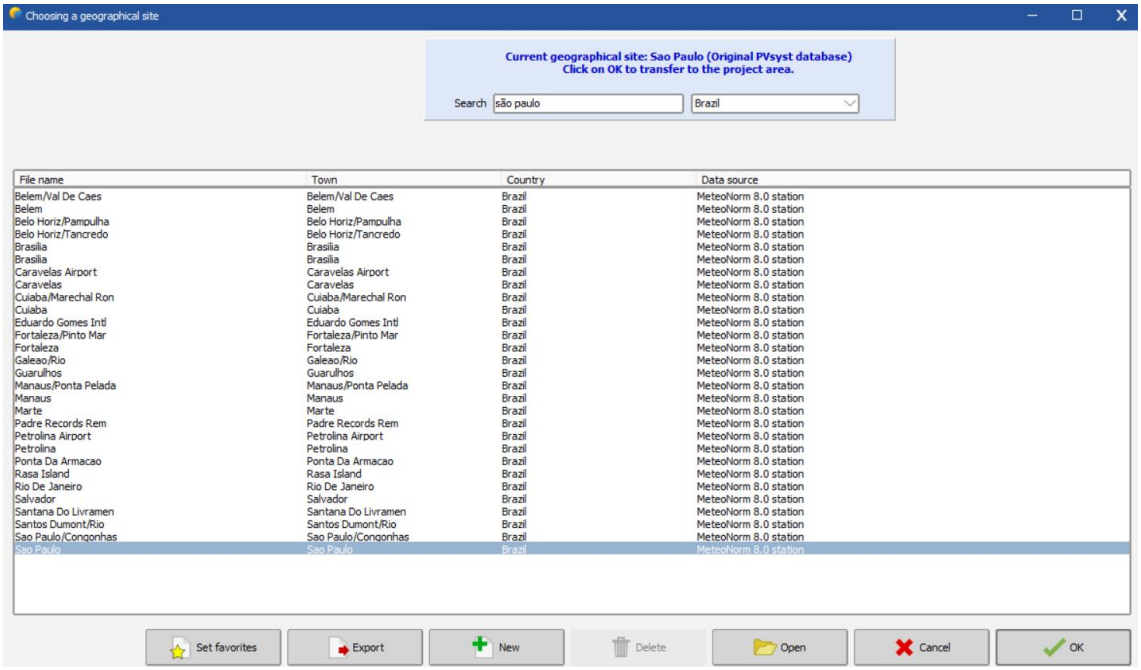

Figura 13 – Definição da localização no software PVsyst.

O próximo passo na operação do software é informar a potência consumida pelas cargas e o período de operação, ou seja, a demanda energética do sistema. A partir da demanda energética e dos valores típicos para a eficiência dos componentes disponíveis comercialmente, como baterias, inversores e controlador de carga, o cálculo de

dimensionamento do sistema é iniciado. A Figura 14 apresenta a etapa de potência consumida do sistema proposto, enquanto a Figura 15 apresenta a mesma etapa para o software PVsyst.

Figura 14 – Definição da potência consumida pelas cargas no sistema proposto.

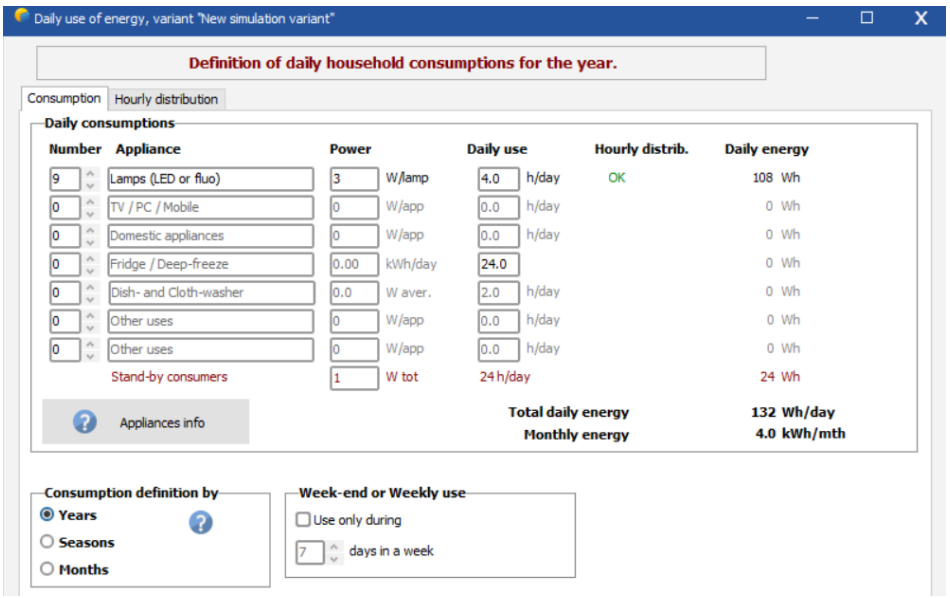

Figura 15 – Definição da potência consumida pelas cargas no software PVsyst.

É possível verificar algumas métricas do sistema, como energia gerada e consumida no período informado, conforme a Figura 16, para o sistema proposto e a Figura 17 para o PVsyst.

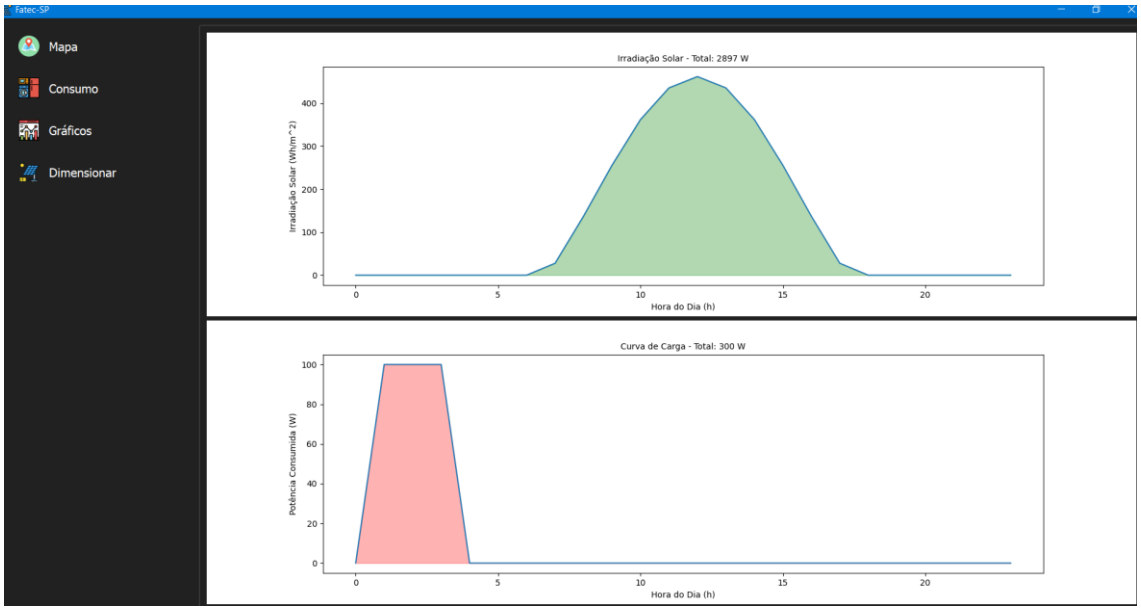

Figura 16 – Visualização gráfica de curvas no sistema proposto.

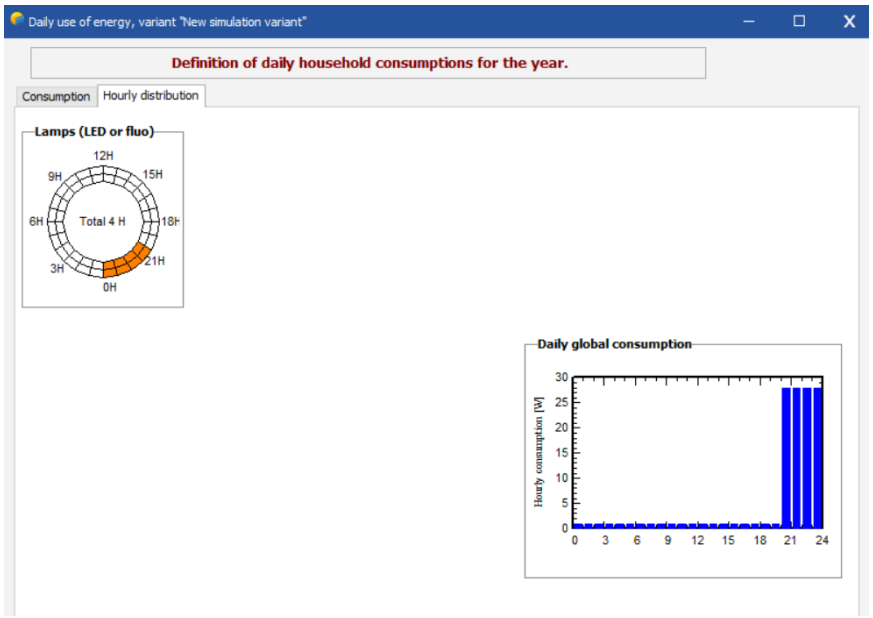

Figura 17 – Visualização gráfica de curvas nosoftware PVsyst.

A última etapa do dimensionamento do sistema é informar os valores nominais dos painéis solares que serão utilizados e o período de autonomia do sistema. Em seguida basta clicar em calcular, para ver os resultados do dimensionamento, conforme a Figura 18. As Figuras 20 a 21 apresenta as respectivas telas do PVsyst.

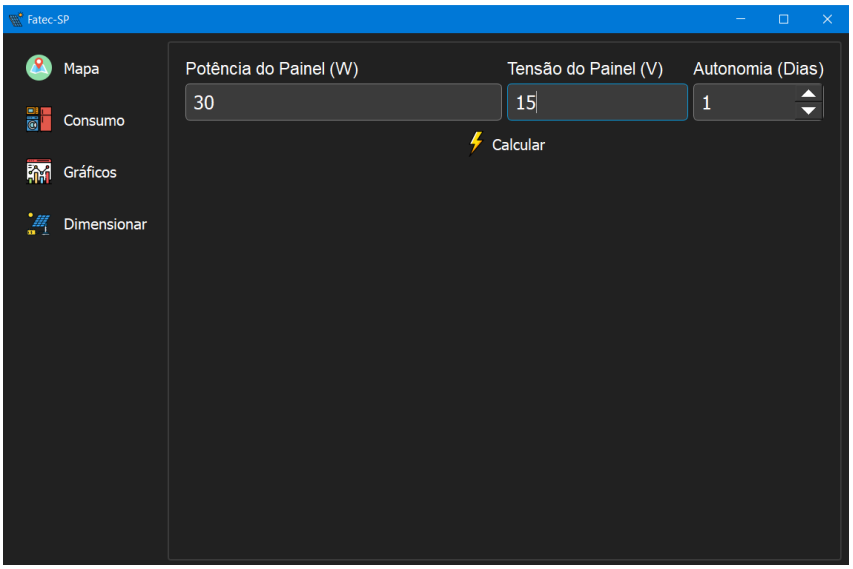

Figura 18 – Etapa de dimensionamento no sistema proposto.

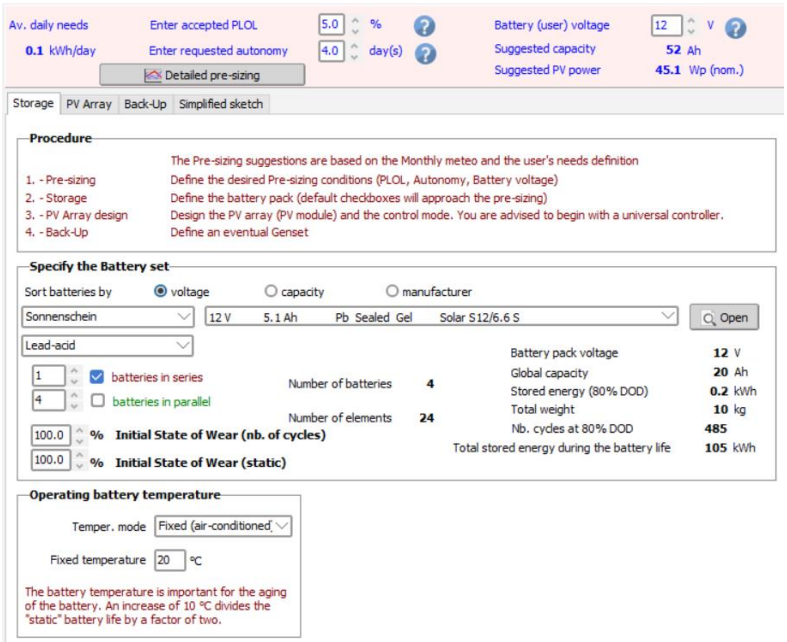

Figura 19 – Seleção de bateria no software PV syst.

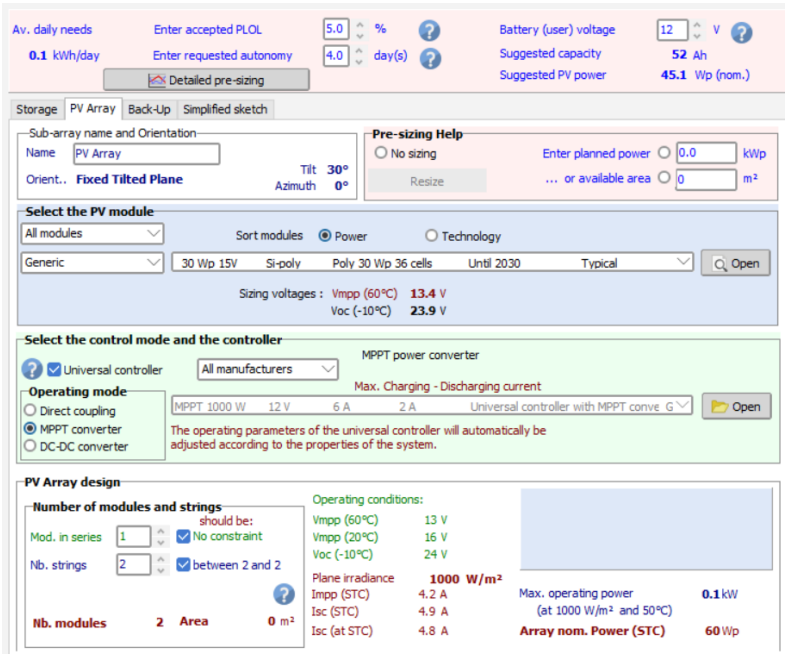

Figura 20 – Seleção dos painéis solares no software PV syst.

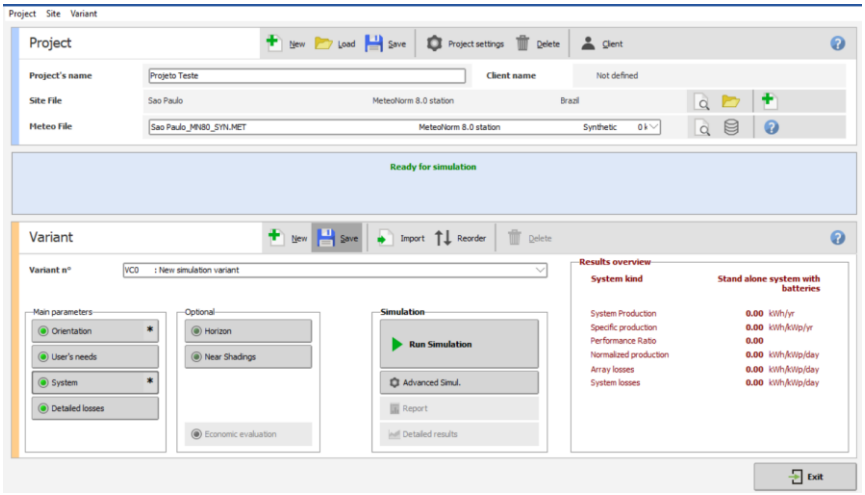

Figura 21 – Etapa de dimensionamento no software PV syst.

5.1 Análise dos Resultados

Ambos os softwares determinaram que seriam necessários 3 painéis solares para suprir a demanda enérgica do sistema. A bateria utilizada deve ter uma capacidade mínima de 18 A/h, segundo o sistema proposto e 20 A/h segundo o PVsyst. Essa diferença ocorre devido ao sistema PVsyst trabalhar diretamente com componentes disponíveis comercialmente, ou seja, o modelo de bateria que mais se aproxima da necessidade do

sistema possui uma capacidade de 5 A/h, usando 4 baterias em paralelo. O controlador de carga utilizado precisa suportar 7 A segundo o software proposto e 6.3 A, segundo o PVsyst. Essa diferença ocorre devido a característica de arredondamento utilizada no sistema proposto. Por questões de segurança, qualquer valor de corrente para o controlador de carga será arredondado para cima. Por se tratar de um sistema que não utiliza tensão alternada, não foi necessário incluir um inversor de frequência no projeto. A Tabela V apresenta a comparação entre os valores fornecidos pelo software PVsyst e pelo sistema proposto. A Figura 22 apresenta os resultados obtidos pelo sistema proposto enquanto a Figura 23 apresenta os resultados obtidos pelo PVsyst.

Tabela V. Comparação Entre O Sistema Proposto E O Software Pvsyst.

Parâmetro	**PVsyst**	**Sistema Proposto**
Quantidade de módulos	3	3
Capacidade da bateria (A/h)	20	18
Controlador de carga (A)	6.3	7

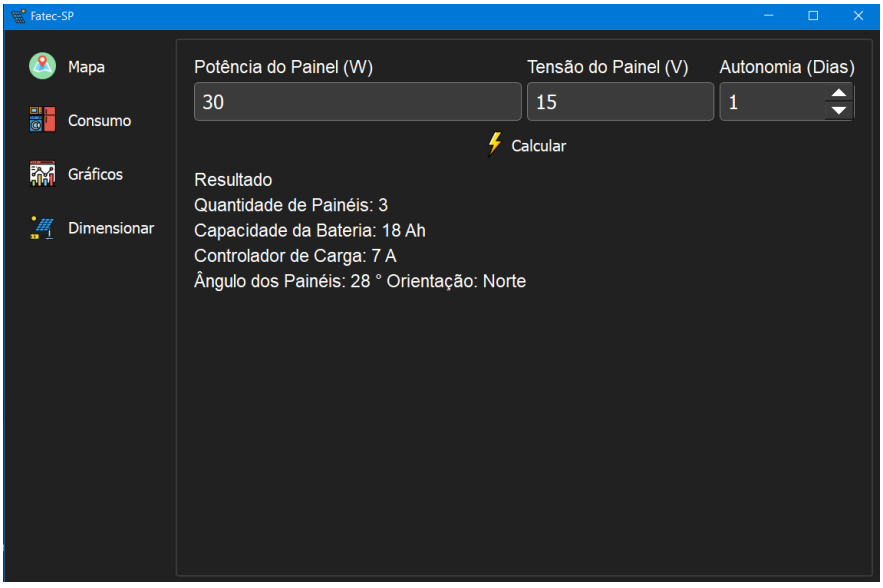

Figura 22 - Resultados obtidos pelo sistema proposto.

PV Array Characteristics

PV module

Manufacturer	Generic
Model	Poly 30 Wp 36 cells
(Original PVsyst database)	
Unit Nom. Power	30 Wp
Number of PV modules	3 units
Nominal (STC)	90 Wp
Modules	3 Strings x 1 In series
At operating cond. (50°C)	
Pmpp	88 Wp
U mpp	14 V
I mpp	6.3 A

Battery

Manufacturer	Generic
Model	Solar S12/6.6 S
Technology	Lead-acid, sealed, Gel
Nb. of units	4 in parallel
Discharging min. SOC	20.0 %
Stored energy	0.2 kWh
Battery Pack Characteristics	
Voltage	12 V
Nominal Capacity	20 Ah (C10)
Temperature	Fixed 20 °C

Controller

Universal controller	
Technology	MPPT converter
Temp coeff.	-5.0 mV/°C/Elem.
Converter	
Maxi and EURO efficiencies	97.0 / 95.0 %

Battery Management control

Threshold commands as	SOC calculation
Charging	SOC = 0.92 / 0.75
approx.	14.2 / 12.4 V
Discharging	SOC = 0.20 / 0.45
approx.	11.5 / 12.1 V

Total PV power

Nominal (STC)	0 kWp
Total	3 modules
Module area	0.7 m²
Cell area	0.4 m²

Figura 22 - Resultados obtidos pelo software PVsyst.

6 Conclusão

Este trabalho apresentou o desenvolvimento e testes preliminares de um software livre desenvolvido em Python para o dimensionamento de sistemas fotovoltaicos. Os resultados indicam que o software proposto tem potencial para ser utilizado no projeto de sistemas solares fotovoltaicos como uma opção gratuita e de fácil operação, quando comparada às ferramentas disponíveis comercialmente.

Embora se encontre numa fase inicial de desenvolvimento, necessitando implementar um grande número de melhorias, tais como dados referentes aos catálogos de equipamentos fotovoltaicos disponíveis no mercado e geração de relatórios contendo as informações de projeto, o fato de ser um software livre tende a torná-lo mais eficiente devido à possível colaboração de diversos centros de pesquisa ou universidades, dispostos a colaborar com o seu desenvolvimento, proporcionando possíveis melhorias.

Quando comparado a uma ferramenta de uso profissional, como o PVsyst, verificou-se o quanto a operação do sistema é simples, o que pode representar um fator decisivo para atrair usuários que não tenham um conhecimento prévio acerca de sistemas fotovoltaicos, contribuindo com o aumento na instalação de sistemas solares autônomos, e, consequentemente, incentivando o uso de fontes renováveis de energia.

Os próximos passos do projeto são incluir tutoriais para guiar o usuário tanto na utilização do software, quanto na instalação de sistemas solares fotovoltaicos autônomos. Além da geração de relatórios detalhados, que contenham os dados do projeto e apresentem a melhor configuração do sistema fotovoltaico de acordo com a localidade, incluindo conexão do equipamento e melhor custo benefício do sistema baseado em equipamentos de uso comercial.

7 Referências

[1] Portal Solar - Simulador solar, [Online]. Disponível: https://www.portalsolar.com.br/calculo-solar.[Acessado: 06-Abril -2021].

[2] P. J. Tavares; G. M. Antônio (Org.). Manual de Engenharia para Sistemas Fotovoltaicos. Rio de Janeiro: Cepel, 2014.

[3] PV*SOL, [Online]. Disponivel: https://www.valentin-software.com/it/prodotti/pvsol. [Acessado: 23-Maio-2018].

[4] PVSYST [Online]. Disponivel: http://www.pvsyst.com/en/. [Acessado: 23-Maio-2018].

[5] BROWN, E. An introduction to Solar Energy, 1988.

[6] PINHO, J. T.; GALDINO, M. A. (Org.). **Manual de Engenharia para Sistemas Fotovoltaicos**. Rio de Janeiro: Cepel, 2014.

[7] STEVEN, S.; HEGEDUS, L.; ANTONIO, L. **Status, Trends, Challenges and the Bright Future of Solar Electricity from Photovoltaics**. in Handbook of Photovoltaic Science and Engineering, Second Edition, Chichester, UK: John Wiley & Sons, 2011, ch1.

[8] FRITTS, A. **New Form of Selenium Cell**. Am. Assoc. Adv. Sci. 33, 97 (1883).

[9] CHAPIN, D.; FULLER, C. PEARSON, G. A New Silicon p-n Junction Photocell for Converting Solar Radiation into Electrical Power. Journal of Applied Physics 25, 676 (1954).

[10] REYNOLDS, D.; LEIES, G.; ANTES, L.; MARBURGER, R. **Photovoltaic Effect in Cadmium Sulfide.** Physical Review. 96, 533, 534 (1954).

[11] JENNY, D.; LOFERSKI, J.; RAPPAPORT, P. **Photovoltaic Effect in GaAs p−n Junctions and Solar Energy Conversion**. Physical Review. 101, 1208, 1209 (1956).

[12] PRINCE, M. **Silicon Solar Energy Converters.** Journal of Applied Physics. 26, 534–540 (1955).

[13] LOFERSKI, J. Theoretical Considerations Governing the Choice of the Optimum Semiconductor for Photovoltaic Solar Energy Conversion. Journal of Applied Physics. 27, 777–784 (1956).

[14] WYSOCKI, J.; RAPPAPORT, P. **Effect of Temperature on Photovoltaic Solar Energy Conversion**. Journal of Applied Physics. 31, 571–578 (1960).

[15] SHOCKLEY, W.; QUEISSER, H. **Detailed Balance Limit of Efficiency of p-n Junction Solar Cells**. Journal of Applied Physics. 32, 510–519 (1961).

[16] CUSANO, D. CdTe solar cells and PV heterojunctions in II-VI compounds. *Solid State Electron.* 6, 217–232 (1963).

[17] WYSOCKI J.; RAPPAPORT, P.; DAVISON, E.; HAND, R. **Lithium-Doped, Radiation-Resistant Silicon Solar Cells.** Applied Physics. *Lett.* 9, 44–46 (1966).

[18] ALFEROV, Z.; *Fiz. Tekh. Poluprovodn.* 4, 2378 (1970).

[19] LINDMAYER, J.; ALLSION, J. **The violet cell: an improved silicon solar cell.** *COMSAT Tech. Rev.* 3, 1–22 (1973).

[20] HOVEL, H.; WOODALL, J. The Effect of Depletion Region Recombination Currents on the Efficiences of Si and GaAs Solar Cells. *Proc. 10th IEEE Photovoltaic Specialist Conf.*, 25–30 (1973).

[21] PREISER, K. Photovoltaic Systems. In: **Handbook of Photovoltaic Science and Engineering**. Second Edition, Chichester, UK: John Wiley & Sons, 2011, ch 17.

[22] DUFFIE, J. A.; BECKMAN, W. A. **Solar Engineering of Thermal Process**. 3. ed. Hoboken: John Wiley & Sons, Inc., 2006.

[23] V. Bahel, H. Bakhsh, and R. Srinivasan, "A correlation for estimation of global solar radiation," *Energy*, pp. 12:131–5, 1987.

[24] C. Tiba, "Atlas Solarimétrico do Brasil, banco de dados solarimétricos", Ed. Universitária da UFPE, Recife, 2000.

[25] https://globalsolaratlas.info/map

[26] Installation and Safety Manual of the Bosch Solar Modules: http://www.bosch-solarenergy.de/media/bosch_se_serviceorganisation/kundendienst/north_america/crystalline_solar_modules/Bosch_Solar_Module_NA30117_NA42117_V11_29052013.pdf. Acessado em 08/05/2021.

Apêndice A – Código Fonte

Classe Main

```
import sys

from PyQt5.QtGui import QIcon

from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton,
QHBoxLayout, QWidget,QVBoxLayout, QTabWidget

from Maps import Map

from Charts import daily_radiation, charge_curve

from Styles import Style

from Loads import Load

from SystemSizing import Sizing


class Window(QMainWindow):

    def __init__(self):

        super().__init__()


        self.coordinate = []

        self.map = Map()

        self.load = Load()

        self.index = 0

        self.sizing = Sizing()

        self.coordinate = self.map.coordinate

        fig_view = daily_radiation(self.map.hourly_radiation)

        fig_view_02 = charge_curve([])


        # set the title of main window

        self.setWindowTitle('Fatec-SP')


        # set the size of window

        self.Width = 800

        self.height = int(0.618 * self.Width)

        self.resize(self.Width, self.height)

        self.setWindowIcon(QIcon('img/solar-panel.png'))

        self.setStyleSheet("Window {"

                           "background-color: #212121;"

                           "}")
```

```
# add all widgets

self.btn_1 = QPushButton('  Mapa', self)
self.btn_2 = QPushButton('  Consumo', self)
self.btn_3 = QPushButton('  Gráficos', self)
self.btn_4 = QPushButton('  Dimensionar', self)
self.btn_1.resize(200, 200)
self.btn_2.resize(200, 200)
self.btn_3.resize(200, 200)
self.btn_1.clicked.connect(self.button1)
self.btn_1.setIcon(QIcon('img/mapa.png'))
self.btn_2.setIcon(QIcon('img/electronics.png'))
self.btn_3.setIcon(QIcon('img/analytics.png'))
self.btn_4.setIcon(QIcon('img/solar-energy.png'))
self.btn_1.setStyleSheet(Style.button)
self.btn_2.setStyleSheet(Style.button)
self.btn_3.setStyleSheet(Style.button)
self.btn_4.setStyleSheet(Style.button)
self.btn_2.clicked.connect(self.button2)
self.btn_3.clicked.connect(self.button3)
self.btn_4.clicked.connect(self.button4)

# add tabs
self.tab1 = self.ui1()
self.tab2 = self.ui2()
# self.tab3 = self.ui3()

self.main_layout = QVBoxLayout()
self.main_layout.addWidget(fig_view)
self.main_layout.addWidget(fig_view_02)

# main_layout.addStretch(5)
self.main = QWidget()
self.main.setLayout(self.main_layout)

self.tab3 = self.main
self.tab4 = self.ui4()
```

```
        self.initUI()

    def initUI(self):
        left_layout = QVBoxLayout()
        left_layout.addWidget(self.btn_1)
        left_layout.addWidget(self.btn_2)
        left_layout.addWidget(self.btn_3)
        left_layout.addWidget(self.btn_4)
        left_layout.addStretch(5)
        left_layout.setSpacing(20)
        left_widget = QWidget()
        left_widget.setLayout(left_layout)
        self.right_widget = QTabWidget()
        self.right_widget.tabBar().setObjectName("mainTab")
        self.right_widget.addTab(self.tab1, '')
        self.right_widget.addTab(self.tab2, '')
        self.right_widget.addTab(self.tab3, '')
        self.right_widget.addTab(self.tab4, '')
        self.right_widget.setCurrentIndex(self.index)
        self.right_widget.setStyleSheet(Style.right_widget)
        main_layout = QHBoxLayout()
        main_layout.addWidget(left_widget)
        main_layout.addWidget(self.right_widget)
        main_layout.setStretch(0, 40)
        main_layout.setStretch(1, 200)
        main_widget = QWidget()
        main_widget.setLayout(main_layout)
        self.setCentralWidget(main_widget)
    # -----------------
    # buttons

    def button1(self):
        self.right_widget.setCurrentIndex(0)

    def button2(self):
        self.right_widget.setCurrentIndex(1)

    def button3(self):
```

```
    self.right_widget.setCurrentIndex(2)
    self.coordinate = self.map.coordinate
    fig_view = daily_radiation(self.map.hourly_radiation)
    fig_view_02 = charge_curve(self.load.loads)
    self.main_layout.itemAt(0).widget().deleteLater()
    self.main_layout.itemAt(1).widget().deleteLater()
    self.main_layout.addWidget(fig_view)
    self.main_layout.addWidget(fig_view_02)
    # main_layout.addStretch(5)
    self.main.setLayout(self.main_layout)
    self.tab3 = self.main

def button4(self):
    self.right_widget.setCurrentIndex(3)
    self.sizing.irradiation = sum(self.map.hourly_radiation)
    self.sizing.load = self.load.get_total_load()
    self.sizing.lat = self.map.coordinate[0]

# -----------------
# pages
def ui1(self):
    main_layout = QVBoxLayout()
    main_layout.addWidget(self.map)
    # main_layout.addStretch(5)

    main = QWidget()
    main.setStyleSheet("QWidget{"
                       "background-color: #3b3b3b;"
                       "}")
    main.setLayout(main_layout)
    return main

def ui2(self):
    main_layout = QVBoxLayout()
    main_layout.addWidget(self.load)
    main_layout.addStretch(5)
    main = QWidget()
    main.setLayout(main_layout)
```

```
        return main

    def ui3(self):
        main_layout = QVBoxLayout()
        main_layout.addWidget(self.fig_view)
        # main_layout.addStretch(5)
        main = QWidget()
        main.setLayout(main_layout)
        return main

    def ui4(self):
        main_layout = QVBoxLayout()
        main_layout.addWidget(self.sizing)
        main_layout.addStretch(5)
        main = QWidget()
        main.setLayout(main_layout)
        return main

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = Window()
    ex.show()
    sys.exit(app.exec_())
```

Classe Maps

```
import io

from PyQt5.QtGui import QIcon

from PyQt5.QtWebEngineWidgets import QWebEngineView

from PyQt5.QtWidgets import  QMainWindow, QPushButton, QLineEdit, QHBoxLayout, QWidget, QVBoxLayout, QMessageBox

import folium

import geocoder

from Styles import Style

from SolarResource import SolarResource


widget_map = QWidget


class Map(QMainWindow):

    def __init__(self):

        super().__init__()


        self.coordinate = [-23.511290, -46.385411]

        self.location = ''

        self.layout = QVBoxLayout()

        self.layout_top = QHBoxLayout()

        self.setGeometry(100, 100, 800, 600)

        self.setWindowTitle('Janela')

        self.input_text = QLineEdit(self)

        self.input_text.move(10, 10)

        self.input_text.resize(200, 30)

        self.button_01 = QPushButton('Buscar', self)

        self.button_01.move(100, 100)

        self.button_01.resize(140, 50)

        self.button_01.setIcon(QIcon('img/search.png'))

        self.setStyleSheet(Style.map)

        self.input_text.setStyleSheet(Style.input_text)

        self.button_01.setStyleSheet(Style.button)

        self.button_01.clicked.connect(self.click)

        self.widget_map = create_maps(self.coordinate, self.location)

        self.layout_top.addWidget(self.input_text)

        self.layout_top.addWidget(self.button_01)

        self.layout.addLayout(self.layout_top)

        self.layout.addWidget(self.widget_map)
```

```
        self.widget = QWidget()
        self.widget.setLayout(self.layout)
        self.setCentralWidget(self.widget)
        self.hourly_radiation =
SolarResource.get_total_hourly_radiation(self.coordinate[0],
self.coordinate[1])


    def click(self):
        text = self.input_text.text()
        if text != '':
            g = geocoder.arcgis(text)
            self.coordinate = g.latlng
            self.location = g.current_result
            widget_map = create_maps(self.coordinate, self.location)
            layout = QVBoxLayout()
            layout_top = QHBoxLayout()
            layout_top.addWidget(self.input_text)
            layout_top.addWidget(self.button_01)
            layout.addLayout(layout_top)
            layout.addWidget(widget_map)
            widget = QWidget()
            widget.setLayout(layout)
            self.setCentralWidget(widget)
            self.hourly_radiation =
SolarResource.get_total_hourly_radiation(self.coordinate[0],
self.coordinate[1])


        else:
            msg = QMessageBox()
            msg.setIcon(QMessageBox.Critical)


            msg.setText("Atenção")
            msg.setInformativeText("É Necessário Digitar um Endereço Válido")
            msg.setWindowTitle("Erro")
            msg.setDetailedText("Utilize o Seguinte Formato: \nRua, Bairro,
Cidade")
            msg.show()
            msg.exec_()
```

```
def create_maps(coordinate, location):
    layout = QHBoxLayout()
    m = folium.Map(
        zoom_start=14,
        location=coordinate
    )
    html = ''

    text = str(location)
    text = text[1:-1]
    x = text.split(", ")

    for i in x:
        html += ('<br>' + i)

    iframe = folium.IFrame(html)
    popup = folium.Popup(iframe,
                         min_width=250,
                         max_width=250)
    folium.Marker(
        location=coordinate,
        popup=popup,
        icon=folium.Icon(icon="info-sign"),
    ).add_to(m)

    # Earth
    tile = folium.TileLayer(
        tiles="https://mt1.google.com/vt/lyrs=s&x={x}&y={y}&z={z}",
        attr="Google",
        name="Google Satellite",
        overlay=True,
        control=True,
    ).add_to(m)

    # ROADMAP
    tile = folium.TileLayer(
        tiles="https://mt1.google.com/vt/lyrs=m&x={x}&y={y}&z={z}",
        attr="Google",
```

```
    name="Google Maps",
    overlay=True,
    control=True,
).add_to(m)

folium.LayerControl().add_to(m)
# save map data to data object
data = io.BytesIO()
m.save(data, close_file=False)

web_view = QWebEngineView()
web_view.setHtml(data.getvalue().decode())
layout.addWidget(web_view)
return web_view
```

Classe Loads

```
from operator import itemgetter

from PyQt5 import QtWidgets

from PyQt5.QtCore import Qt

from PyQt5.QtGui import QIcon, QFont, QIntValidator

from PyQt5.QtWidgets import  QMainWindow, QPushButton, QLabel, QLineEdit,
QHBoxLayout, QWidget,QVBoxLayout, QMessageBox

from Styles import Style

widget_map = QWidget


class Load(QMainWindow):

    def __init__(self):

        super().__init__()


        self.layout = QVBoxLayout()

        self.index = 4

        self.loads = []

        self.layout_01 = QVBoxLayout()

        self.layout_02 = QVBoxLayout()

        self.layout_03 = QVBoxLayout()

        self.layout_04 = QHBoxLayout()

        self.layout_05 = QVBoxLayout()

        self.layout_top_01 = QHBoxLayout()

        self.setGeometry(100, 100, 800, 600)

        self.setWindowTitle('Janela')

        self.text_01 = QLabel('Aparelho')

        self.text_02 = QLabel('Consumo (W/h)')

        self.text_03 = QLabel('Hora Inicial')

        self.text_04 = QLabel('Hora Final')

        font = QFont("Arial", 14)

        self.text_01.setFont(font)

        self.text_02.setFont(font)

        self.text_03.setFont(font)

        self.text_04.setFont(font)

        self.text_01.setStyleSheet(Style.label)

        self.text_02.setStyleSheet(Style.label)

        self.text_03.setStyleSheet(Style.label)

        self.text_04.setStyleSheet(Style.label)

        self.input_text_01 = QLineEdit(self)
```

```
self.input_text_01.move(10, 10)
self.input_text_01.resize(400, 30)
self.input_text_02 = QLineEdit(self)
self.input_text_02.move(10, 10)
self.input_text_02.resize(200, 30)
self.input_text_03 = QtWidgets.QSpinBox(self)
self.input_text_03.setRange(0, 23)
self.input_text_03.move(10, 10)
self.input_text_03.resize(200, 30)
self.input_text_04 = QtWidgets.QSpinBox(self)
self.input_text_04.setRange(0, 23)
self.input_text_04.move(10, 10)
self.input_text_04.resize(200, 30)
self.input_text_01.setStyleSheet(Style.input_text)
self.input_text_02.setStyleSheet(Style.input_text)
self.input_text_03.setStyleSheet(Style.spinBox)
self.input_text_04.setStyleSheet(Style.spinBox)
self.input_text_02.setValidator(QIntValidator())
# self.input_text_03.setValidator(QIntValidator(0, 2))
self.button_01 = QPushButton('Adicionar', self)
self.button_01.move(100, 100)
self.button_01.resize(140, 50)
self.button_02 = QPushButton('Remover', self)
self.button_02.move(100, 100)
self.button_02.resize(140, 50)
self.button_01.setIcon(QIcon('img/plus.png'))
self.button_02.setIcon(QIcon('img/negative.png'))
self.setStyleSheet(Style.window)
self.button_01.setStyleSheet(Style.table_button)
self.button_02.setStyleSheet(Style.table_button)
self.button_01.clicked.connect(self.add)
self.button_02.clicked.connect(self.remove)
self.layout_01.addWidget(self.text_01)
self.layout_01.addWidget(self.input_text_01)
self.layout_top_01.addLayout(self.layout_01)
self.layout_02.addWidget(self.text_02)
self.layout_02.addWidget(self.input_text_02)
self.layout_04.addLayout(self.layout_02)
```

```
        self.layout_03.addWidget(self.text_03)
        self.layout_03.addWidget(self.input_text_03)
        self.layout_04.addLayout(self.layout_03)
        self.layout_05.addWidget(self.text_04)
        self.layout_05.addWidget(self.input_text_04)
        self.layout_04.addLayout(self.layout_05)
        self.layout_top_01.addLayout(self.layout_04)
        self.layout.addLayout(self.layout_top_01)
        self.layout.addWidget(self.button_01)
        self.layout.addWidget(self.button_02)
        text = QLabel('Aparelhos ')
        text.setStyleSheet(Style.label)
        font = QFont("Arial", 14)
        text.setFont(font)
        self.layout.addWidget(text)
        self.layout.addStretch(1)
        self.widget = QWidget()
        self.widget.setLayout(self.layout)
        self.setCentralWidget(self.widget)


    def add(self):


        if self.input_text_01.text() != '' and self.input_text_02.text() != ''
\
                and self.input_text_03.text() != '' and
self.input_text_04.text() != '':
            if self.index < 19:
                text = QLabel(
                    'Aparelho: ' + self.input_text_01.text() + ', Consumo: ' +
self.input_text_02.text() + 'W, Horario de Funcionamento:' +
self.input_text_03.text() + ':00 - ' + self.input_text_04.text() + ':00')
                text.setStyleSheet(Style.label_2)
                font = QFont("Arial", 12)
                text.setAlignment(Qt.AlignCenter)
                text.setFont(font)
                self.layout.addWidget(text)
                self.widget.setLayout(self.layout)
                self.setCentralWidget(self.widget)
                self.index += 1
                self.loads.append(
```

```
                {'consumo': int(self.input_text_02.text()),
                 'hora inicial': int(self.input_text_03.text()),
                 'hora final': int(self.input_text_04.text()),
                 })

    else:
        msg = QMessageBox()
        msg.setIcon(QMessageBox.Critical)
        msg.setText("Atenção")
        msg.setInformativeText("É Necessário Preencher Todos os Campos")
        msg.setWindowTitle("Erro")
        msg.setDetailedText("")
        msg.show()
        msg.exec_()

def remove(self):
    if self.index > 4:
        self.layout.itemAt(self.index).widget().deleteLater()
        self.index -= 1
        self.loads.pop()

def get_total_load(self):
    sorted_list = sorted(self.loads, key=itemgetter('hora inicial'))

    load = []
    for i in range(0, 24):
        load.append(0)

    for item in sorted_list:
        for i in range(0, 24):
            if item['hora inicial'] == i:
                j = i
                while item['hora final'] > j:
                    load[j] += item['consumo']
                    j += 1

    return sum(load)
```

Classe Charts

```
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg
from matplotlib.figure import Figure
from operator import itemgetter



class MplCanvas(FigureCanvasQTAgg):
    def __init__(self, parent=None, width=5, height=4, dpi=100):
        fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = fig.add_subplot(111)
        super(MplCanvas, self).__init__(fig)


def daily_radiation(hourly_radiation):
    x = []
    for i in range(0, 24):
        x.append(i)


    sc = MplCanvas(width=5, height=4, dpi=100)
    sc.axes.plot(x, hourly_radiation, label='Inline label')


    maxy = max(hourly_radiation)


    sc.axes.fill_between(x, hourly_radiation, color='green', alpha=0.3)
    sc.legend = 'Total: ' + str(int(sum(hourly_radiation)))


    sc.axes.set_title('Irradiação Solar - Total: ' +
str(int(sum(hourly_radiation))) + ' W', fontsize=10)
    sc.axes.set_ylabel('Irradiação Solar (Wh/m^2)', fontsize=10)
    sc.axes.set_xlabel('Hora do Dia (h)', fontsize=10)
    return sc


def charge_curve(loads):
    # getting the data
    x = []
    for i in range(0, 24):
        x.append(i)


    sorted_list = sorted(loads, key=itemgetter('hora inicial'))
```

```
    load = []
    for i in range(0, 24):
        load.append(0)

    for item in sorted_list:
        for i in range(0, 24):
            if item['hora inicial'] == i:
                j = i
                while item['hora final'] > j:
                    load[j] += item['consumo']
                    j += 1

    maxy = max(load)

    sc = MplCanvas(width=5, height=4, dpi=100)
    sc.axes.plot(x, load, label='Inline label')
    sc.axes.fill_between(x, load, alpha=0.30, color='red')
    sc.axes.set_title('Curva de Carga - Total: ' + str(sum(load)) + ' W',
fontsize=10)
    sc.axes.set_ylabel('Potência Consumida (W)', fontsize=10)
    sc.axes.set_xlabel('Hora do Dia (h)', fontsize=10)
    return sc
```

Classe PVSystem

```
import math
def get_panel_power_total(load, Ib):
    print(Ib)
    L = load / 0.85
    panel_power_total = L / ((Ib / 1000) * 0.75 * 0.9)
    return panel_power_total

def get_panel_parallel(panel_power, panel_power_total):
    return int(math.ceil(panel_power_total / panel_power))

def get_panel_current(panel_power, panel_voltage):
    return panel_power / panel_voltage

def storage_block_sizing(load, n):
    CB = (load * n) / 0.5
    CBI = CB / 12
    return CBI

def charge_controller_sizing(panel_parallel, panel_current):
    return 1.25 * panel_parallel * panel_current

def get_panel_angle(lat):
    if 0 < lat < 11:
        return 10
    if 11 < lat < 21:
        return lat
    if 21 < lat < 30:
        return lat + 5
    if 31 < lat < 41:
        return lat + 10

    return lat + 15

def get_panel_orientation(lat):
    if lat >= 0:
        return 'Sul'
    return 'Norte
```

Classe Solar Resource

```
import numpy as np

import math

import urllib.request

import json

from types import SimpleNamespace


class SolarResource:

    @staticmethod

    def get_date_by_day_number(day_number):

        month = ['January', 'February', 'March', 'April', 'May', 'June',
'July', 'August', 'September', 'October',

                 'November', 'December']


        number_of_days_by_month = [31, 59, 90, 120, 151, 181, 212, 243, 273,
304, 334, 365]

        date = []

        for value in number_of_days_by_month:

            if value >= day_number:

                if value > 31:


date.append(number_of_days_by_month[number_of_days_by_month.index(value) - 1])


date.append(number_of_days_by_month[number_of_days_by_month.index(value)])

                    date.append((day_number -
number_of_days_by_month[number_of_days_by_month.index(value) - 1]))

                else:

                    date.append(1)


date.append(number_of_days_by_month[number_of_days_by_month.index(value)])

                    date.append(day_number)


                date.append(month[number_of_days_by_month.index(value)])

                return date


    @staticmethod

    def get_declination_list():

        declination_list = []

        for n in range(1, 366):

            declination_list.append(23.45 * (math.sin(math.radians((360 / 365)
* (284 + n)))))
```

```
        return declination_list


    @staticmethod
    def get_hourly_angle_of_the_sunset(latitude, declinations_list):
        hourly_angle_of_sunset_list = []
        for declination in declinations_list:
            hourly_angle_of_sunset_list.append((math.acos((-
math.tan(math.radians(latitude))
                                                           *
(math.tan(math.radians(declination)))))))
        return hourly_angle_of_sunset_list


    @staticmethod
    def get_hours_with_sunlight_list(hourly_angle_of_sunset_list):
        hours_with_sunlight_list = []
        for hourly_angle_of_sunset in hourly_angle_of_sunset_list:
            hours_with_sunlight_list.append((2 / 15) *
math.degrees(hourly_angle_of_sunset))
        return hours_with_sunlight_list


    @staticmethod
    def get_daily_extraterrestrial_irradiation(latitude):
        n = 1
        extraterrestrial_irradiation_list = []
        declination_list = SolarResource.get_declination_list()
        hourly_angle_of_sunset_list =
SolarResource.get_hourly_angle_of_the_sunset(latitude, declination_list)


        for angle in hourly_angle_of_sunset_list:
            extraterrestrial_irradiation_list.append(
                10443.1107 * (1 + 0.033 * math.cos(math.radians(0.98630137 *
n)))
                * ((math.cos(math.radians(latitude)) * math.cos(math.radians(
                    declination_list[n - 1])) * math.sin(angle)) + (0.01745329
*

math.degrees(angle)) * math.sin(
                    math.radians(latitude)) *
                   math.sin(math.radians(declination_list[n - 1]))))


            n = n + 1
```

```
        return extraterrestrial_irradiation_list


    @staticmethod
    def get_average_daily_duration_of_month(hours_with_sunlight_list):


        n = hours_with_sunlight_list.index(min(hours_with_sunlight_list))


        date = SolarResource.get_date_by_day_number(n)
        a = date[0]
        b = date[1]
        list_of_hours = hours_with_sunlight_list[a:b + 1]
        average = np.mean(list_of_hours)


        return average


    @staticmethod
    def get_average_daily_extraterrestrial_irradiation(
            daily_extraterrestrial_irradiation_list, hours_with_sunlight_list,
n, lat, long):


        contents = urllib.request.urlopen(
            "https://api.globalsolaratlas.info/data/lta?loc=" + str(lat) + ','
+ str(long)).read()
        x = json.loads(contents, object_hook=lambda d: SimpleNamespace(**d))


        if hasattr(x.annual.data, 'GHI'):
            average_hours_of_month = (math.ceil(x.annual.data.GHI / 365))
        else:
            average_hours_of_month = 3


        N =
SolarResource.get_average_daily_duration_of_month(hours_with_sunlight_list)
        a = average_hours_of_month / N


        average_daily_extraterrestrial_irradiation =
daily_extraterrestrial_irradiation_list[n] * (
                0.16 + 0.87 * a - 0.61 * math.pow(a, 2) + 0.34 * math.pow(a,
3))
        return average_daily_extraterrestrial_irradiation
```

```
    @staticmethod
    def get_total_hourly_radiation(lat, long):
        declinations_list = SolarResource.get_declination_list()
        hourly_angles_of_sunset =
SolarResource.get_hourly_angle_of_the_sunset(int(lat), declinations_list)
        hours_with_sunlight_list =
SolarResource.get_hours_with_sunlight_list(hourly_angles_of_sunset)
        daily_extraterrestrial_irradiation_list =
SolarResource.get_daily_extraterrestrial_irradiation(int(lat))

        coldest_day_of_year =
hours_with_sunlight_list.index(min(hours_with_sunlight_list))

        average_daily_extraterrestrial_irradiation =
SolarResource.get_average_daily_extraterrestrial_irradiation(
            daily_extraterrestrial_irradiation_list, hours_with_sunlight_list,
coldest_day_of_year, lat, long)

        c = 0.409 + 0.5016 *
math.sin((hourly_angles_of_sunset[coldest_day_of_year]) - math.radians(60))
        d = 0.6609 - 0.4767 *
math.sin((hourly_angles_of_sunset[coldest_day_of_year]) - math.radians(60))

        hourly_angle_of_sun_list = []
        total_hourly_radiation_list = []

        for hour in range(0, 24):
            angle = (hour - 12) * 15
            hourly_angle_of_sun_list.append(angle)

        for hourly_angle_of_sun in hourly_angle_of_sun_list:
            x = (0.1308997 * (c + d *
math.cos(math.radians(hourly_angle_of_sun))))
            y = (math.cos(math.radians(hourly_angle_of_sun)) -
math.cos(hourly_angles_of_sunset[coldest_day_of_year]))
            z = (math.sin(hourly_angles_of_sunset[coldest_day_of_year]) -
(hourly_angles_of_sunset[coldest_day_of_year])
                 * math.cos(hourly_angles_of_sunset[coldest_day_of_year]))

total_hourly_radiation_list.append(average_daily_extraterrestrial_irradiation
* x * (y / z))

        for i in range(0, 24):
            if 6 < i < 18:
```

```
        if total_hourly_radiation_list[i] < 0:
            total_hourly_radiation_list[i] = 0
    else:
        total_hourly_radiation_list[i] = 0
return total_hourly_radiation_list
```

Classe SystemSizing

```
from PyQt5 import QtWidgets

from PyQt5.QtGui import QIcon, QFont, QIntValidator

from PyQt5.QtWidgets import  QMainWindow, QPushButton, QLabel, QLineEdit,
QHBoxLayout, QWidget,QVBoxLayout, QMessageBox

from Styles import Style

from PVSystem import *


class Sizing(QMainWindow):

    def __init__(self):

        super().__init__()


        self.load = 0

        self.irradiation = 0

        self.panel_power = 200

        self.panel_voltage = 12

        self.days_of_autonomy = 1

        self.lat = 0


        self.index = 1


        self.layout = QVBoxLayout()

        self.layout_01 = QVBoxLayout()

        self.layout_02 = QVBoxLayout()

        self.layout_03 = QVBoxLayout()

        self.layout_04 = QHBoxLayout()

        self.layout_05 = QVBoxLayout()

        self.layout_top_01 = QHBoxLayout()

        self.setGeometry(100, 100, 800, 600)

        self.setWindowTitle('Janela')


        self.text_01 = QLabel('Potência do Painel (W)')

        self.text_02 = QLabel('Tensão do Painel (V)')

        self.text_03 = QLabel('Autonomia (Dias)')

        font = QFont("Arial", 14)

        self.text_01.setFont(font)

        self.text_02.setFont(font)
```

```
self.text_03.setFont(font)
self.text_01.setStyleSheet(Style.label)
self.text_02.setStyleSheet(Style.label)
self.text_03.setStyleSheet(Style.label)
self.input_text_01 = QLineEdit(self)
self.input_text_01.move(10, 10)
self.input_text_01.resize(400, 30)
self.input_text_02 = QLineEdit(self)
self.input_text_02.move(10, 10)
self.input_text_02.resize(200, 30)
self.input_text_03 = QtWidgets.QSpinBox(self)
self.input_text_03.setRange(1, 5)
self.input_text_01.setStyleSheet(Style.input_text)
self.input_text_02.setStyleSheet(Style.input_text)
self.input_text_03.setStyleSheet(Style.spinBox)
self.input_text_01.setValidator(QIntValidator())
self.input_text_02.setValidator(QIntValidator())
# self.input_text_03.setValidator(QIntValidator(0, 2))
self.button_01 = QPushButton('Calcular', self)
self.button_01.move(100, 100)
self.button_01.resize(140, 50)
self.button_01.setIcon(QIcon('img/flash.png'))
self.setStyleSheet(Style.window)
self.button_01.setStyleSheet(Style.table_button)
self.button_01.clicked.connect(self.sizing)
self.layout_01.addWidget(self.text_01)
self.layout_01.addWidget(self.input_text_01)
self.layout_top_01.addLayout(self.layout_01)
self.layout_02.addWidget(self.text_02)
self.layout_02.addWidget(self.input_text_02)
self.layout_03.addWidget(self.text_03)
self.layout_03.addWidget(self.input_text_03)
self.layout_04.addLayout(self.layout_02)
self.layout_04.addLayout(self.layout_03)
self.layout_04.addLayout(self.layout_05)
self.layout_top_01.addLayout(self.layout_04)
self.layout.addLayout(self.layout_top_01)
```

```
        self.layout.addWidget(self.button_01)
        self.widget = QWidget()
        self.widget.setLayout(self.layout)
        self.setCentralWidget(self.widget)


    def sizing(self):
        if self.input_text_01.text() != '' and self.input_text_02.text() !=
'':
            self.panel_power = int(self.input_text_01.text())
            self.panel_voltage = int(self.input_text_02.text())
            self.days_of_autonomy = int(self.input_text_03.text())


            total_panel_power = get_panel_power_total(self.load,
self.irradiation)
            panel_parallel = get_panel_parallel(self.panel_power,
total_panel_power)
            panel_current = get_panel_current(self.panel_power,
self.panel_voltage)
            battery = int(math.ceil(storage_block_sizing(self.load,
self.days_of_autonomy)))
            controller = int(charge_controller_sizing(panel_parallel,
panel_current))
            angle = int(get_panel_angle(abs(self.lat)))
            orientation = get_panel_orientation(self.lat)


            text_01 = QLabel('Quantidade de Painéis: ' + str(panel_parallel))
            text_02 = QLabel('Capacidade da Bateria: ' + str(battery) + ' Ah')
            text_03 = QLabel('Controlador de Carga: ' + str(controller) + '
A')
            text_04 = QLabel('Ângulo dos Painéis: ' + str(angle) + ' ° ' +
'Orientação: ' + orientation)


            text_01.setStyleSheet(Style.label)
            text_02.setStyleSheet(Style.label)
            text_03.setStyleSheet(Style.label)
            text_04.setStyleSheet(Style.label)
            font = QFont("Arial", 14)
            text_01.setFont(font)
            text_02.setFont(font)
            text_03.setFont(font)
            text_04.setFont(font)
```

```
        text = QLabel('Resultado')
        text.setStyleSheet(Style.label)
        font = QFont("Arial", 14)
        text.setFont(font)

        if self.index > 1:
            self.layout.itemAt(2).widget().deleteLater()
            self.layout.itemAt(3).widget().deleteLater()
            self.layout.itemAt(4).widget().deleteLater()
            self.layout.itemAt(5).widget().deleteLater()
            self.layout.itemAt(6).widget().deleteLater()
            self.index = 1

        self.layout.addWidget(text)
        self.layout.addWidget(text_01)
        self.layout.addWidget(text_02)
        self.layout.addWidget(text_03)
        self.layout.addWidget(text_04)
        self.index = 6
        self.widget = QWidget()
        self.widget.setLayout(self.layout)
        self.setCentralWidget(self.widget)

    else:
        msg = QMessageBox()
        msg.setIcon(QMessageBox.Critical)
        msg.setText("Atenção")
        msg.setInformativeText("É Necessário Preencher Todos os Campos")
        msg.setWindowTitle("Erro")
        msg.setDetailedText("")
        msg.show()
        msg.exec_()
```

Classe Styles

```
class Style:
    map = ("Map{"
           "background-color: #3b3b3b;"
           "border: 1px solid black;"
           "}"
           )
    window = ("Window{"
              "background-color: #3b3b3b;"
              "border: 1px solid black;"
              "padding: 50px;}"
              "}"
              )
    table = ("QTableWidget{"
              "background-color: #3b3b3b;"
             "color: white;"
              "}"
              )
    label= ("QLabel{"
             "color: white;"
             "text-align: center;"
             "}"
             )
    label_2 = ("QLabel{"
             "color: white;"
             "background-color: #1f1f1f;"
            "border: 1px  solid gray;"

            "border-radius: 6px;"
            "padding:  6px;"
             "}"
             )
    input_text = (
        "QLineEdit{"
        "background-color:#3b3b3b;"
        "border: 1px  solid gray;"
        "color: white;"
```

```
    "border-radius: 6px;"
    "padding: 8px;"
    "selection-background-color: darkgray;"
    "font-size: 25px;}"
    "QLineEdit:focus { "
    "border: 1px  solid #0d9adb;}"
    "}"
)

spinBox = (
    "QSpinBox{"
    "background-color:#3b3b3b;"
    "border: 1px  solid gray;"
    "color: white;"
    "border-radius: 6px;"
    "padding: 8px;"
    "selection-background-color: darkgray;"
    "font-size: 25px;}"
    "QLineEdit:focus { "
    "border: 1px  solid #0d9adb;}"
    "}"
)
button = ("QPushButton{"
          "text-align: left;"
          "color: #ffffff;"
          "padding-left: 10px;"
          "padding-right: 10px;"
          "border: 0px solid #4588f5;"
          "font-size: 20px;"
          "qproperty-iconSize: 35px;"
          "height : 50px;"
          "border-radius: 8px}"
          "QPushButton::hover"
          "{"
          "background-color : #525963;"
          "}"
          "QPushButton::pressed"
          "{"
```

```
            "background-color : #2b2b2b;"
            "}"
            )
    table_button = ("QPushButton{"
            "text-align: center;"
            "color: #ffffff;"
            "padding-left: 10px;"
            "padding-right: 10px;"
            "border: 0px solid #4588f5;"
            "font-size: 20px;"
            "qproperty-iconSize: 35px;"
            "height : 50px;"
            "border-radius: 8px}"
            "QPushButton::hover"
            "{"
            "background-color : #525963;"
            "}"
            "QPushButton::pressed"
            "{"
            "background-color : #2b2b2b;"
            "}"
            )
    right_widget = (
        '''QTabBar::tab{width: 0; \
                    height: 0; margin: 0; padding: 0; border: none;fill:
#3b3b3b;
                    }
                    QTabWidget::pane {
                    border-width: 2px;
                    border-style: solid;
                    border-color: #3b3b3b;
                    border-radius: 4px;
                }
                QTabWidget::tab-bar {
                    left: 5px;

                }
                    '''
    )
```