DESENVOLVIMENTO E AVALIAÇÃO DE UM SEGUIDOR SOLAR DE EIXO DUPLO

DANIEL G. ALMEIDA[1]

[1]*Graduando em Bacharelado em Engenharia Elétrica - Departamento de Engenharia Elétrica*
Universidade Federal de Viçosa - Av. P. H. Rolfs, s/n, Campus UFV, CEP: 36570-000, Viçosa, MG, Brasil

E-MAIL: daniel.g.almeida@ufv.br

Resumo—Este trabalho apresenta o desenvolvimento e a avaliação de um rastreador solar de eixo duplo, projetado para maximizar a captação de energia solar por meio do alinhamento do módulo fotovoltaico com o movimento do Sol. O sistema utiliza sensores de luminosidade e servomotores controlados por um ESP32 para ajustar os ângulos de inclinação e orientação. Em comparação a um sistema fixo, o rastreador demonstrou aumento de 21,09% na eficiência energética, destacando-se como uma solução eficaz para projetos de energia renovável.

Palavras-chave—Seguidor Solar de Dois Eixos, Módulo Fotovoltaico, Energia Renovável, Eficiência Energética.

I. INTRODUÇÃO

Segundo a Agência Internacional de Energia (IEA) a necessidade mundial por eletricidade está prevista para aumentar em um ritmo mais acelerado nos próximos três anos, impulsionada pelo avanço da transição para fontes de energia limpa. Nessa perspectiva, é projetado que toda a demanda adicional seja suprida por tecnologias que geram eletricidade com baixas emissões. No ano de 2023, houve um aumento de 50% na capacidade de energia renovável em comparação com 2022, e os anos subsequentes estão previstos para experimentar o crescimento mais acelerado já registrado.

Nesse contexto, o seguidor solar ou tracker apresenta vantagens notáveis, como a maximização da eficiência na geração de energia e uma produção mais consistente ao longo do dia. Além disso, contribui para a redução das emissões de gases de efeito estufa, alinhando-se com os esforços globais para combater as mudanças climáticas, como destacado por Bessa Neto et al. Portanto, trata-se de uma inovação crucial no campo da energia renovável, especialmente diante da crescente demanda por soluções mais eficientes e sustentáveis.

Um desses métodos é o sistema de rastreamento de eixo único. O sistema solar de rastreamento de eixo único é projetado com um grau de liberdade que permite o movimento dos módulos de leste para oeste, seguindo o movimento do sol ao longo do dia, desde o nascer até o pôr do sol, como descrito por Lewandoski et al.

Em contraste, os rastreadores solares de eixo duplo possuem dois graus de liberdade, permitindo a atuação de dois eixos de rotação, também descrito por Lewandoski et al. Com essa flexibilidade, eles conseguem também ajustar sua posição conforme as mudanças nos ângulos de incidência solar ao longo do ano, o que os torna ideais para regiões com variações significativas na posição do Sol.

As vantagens dos seguidores solares de eixo duplo incluem uma adaptação dinâmica que possibilita ajustes precisos, garantindo uma exposição ótima aos raios solares em diferentes condições atmosféricas. Isso resulta em uma eficiência aprimorada na captação de luz solar, gerando maior produção ao longo do tempo.

Fig. 1: Movimentação de um seguidor solar de dois eixos. (Imagem de autoria própria)

De acordo com Cortez, uma das principais desvantagens desse tipo de seguidor é o esforço a que está sujeito, especialmente devido ao peso e à necessidade de ser bloqueado em condições de vento forte. Além disso, em muitas aplicações, a complexidade mecânica envolvida torna esses seguidores não competitivos. Portanto, embora essas desvantagens sejam importantes, sua relevância pode variar dependendo das características específicas do projeto e das condições locais. Cada sistema solar deve ser cuidadosamente avaliado para garantir que atenda às necessidades específicas.

Este trabalho aborda a concepção, construção e implementação de um seguidor solar de eixo duplo, enfocando a escolha de mais de um eixo para otimizar a orientação do módulo fotovoltaico diariamente e sazonalmente. A implementação inclui o uso de uma interface web (Thingable) utilizada para monitorar continuamente os ângulos. Além

disso, o estudo realiza uma avaliação comparativa da eficiência energética do seguidor em relação aos sistemas fotovoltaicos fixos convencionais para uma compreensão do desempenho da tecnologia.

II. Metodologia

A. Construção Física

O projeto do seguidor solar (Fig. 2) apresenta uma construção simples, trata-se de painéis de fibra de madeira que são livres para girar nos 2 eixos. Centralizando o controle, o ESP32 atua como o cérebro do sistema, sendo um microcontrolador de 32 bits que integra Wi-Fi. Compacto e de baixo custo, ele permite a comunicação sem fio, ideal para aplicações em projetos de automação e controle remoto.

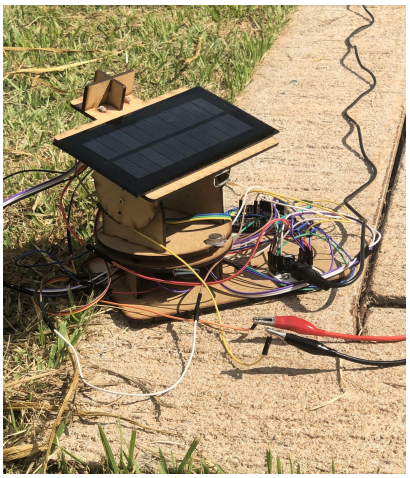
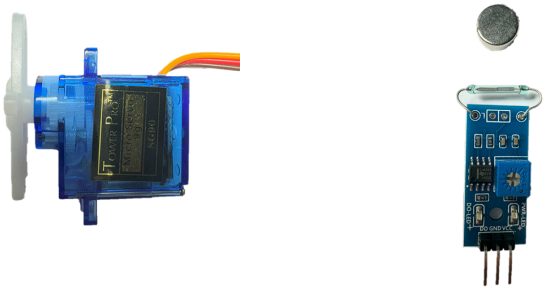


Fig. 2: Protótipo do seguidor solar. (Foto de autoria própria)

A movimentação precisa do módulo fotovoltaico é garantida por dois servos motores (Fig. 3 (a)), controlados pelo ESP32. Esses motores ajustam a posição horizontal e vertical, otimizando a exposição solar ao longo do dia. A presença de sensores de fim de curso assegura que os movimentos não ultrapassem limites físicos, promovendo a durabilidade do sistema.

Esses sensores de fim de curso ou *reed switch*, são sensores magnéticos (Fig. 3 (b)), posicionados estrategicamente para detectar o fim do alcance mecânico. Quando a estrutura se movimenta até o ponto limite em uma direção, um ímã de neodímio se aproxima do sensor, ativando-o e interrompendo o movimento nessa direção. Esse mecanismo evita movimentos excessivos que poderiam danificar o sistema, garantindo que o sistema permaneça dentro de uma faixa segura de operação.

Os quatro sensores de luz (Fig. 4), poscionados acima do módulo fotovoltaico, em conjunto com resistores em série, desempenham o papel da medição da intensidade luminosa do ambiente. Estrategicamente posicionados, esses sensores formam divisores de tensão (Fig.5), gerando sinais proporcionais à luz incidente que são captados pelas entradas analógicas do ESP32. Esses dados são essenciais para determinar a orientação do módulo fotovoltaico. Quando os dois sensores superiores recebem mais luz do que os dois inferiores, ocorre



(a) Servo motor. (b) Sensor magnético.

Fig. 3: (a) Servo motor e (b) Sensor magnético. (Foto de autoria própria)

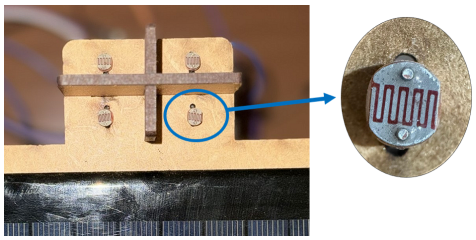


Fig. 4: Sensor LDR (Light Dependent Resistor) utilizado no sistema para detecção de luminosidade. (Foto de autoria própria)

uma disparidade na tensão captada pelos sensores. Isso sinaliza ao sistema que o módulo fotovoltaicos deve ser ajustados para cima, em direção à fonte de luz solar. O controle motorizado entra em ação, girando-o na direção apropriada até que um equilíbrio seja alcançado, mantendo uma orientação ideal em relação ao sol. O mesmo princípio se aplica ao movimento na direção oposta, se os sensores inferiores captarem mais luz do que os superiores, o sistema interpreta a necessidade de ajustar o módulo para baixo.

Essa lógica de controle é estendida para os movimentos horizontais. Se os sensores LDR do lado direito detectam

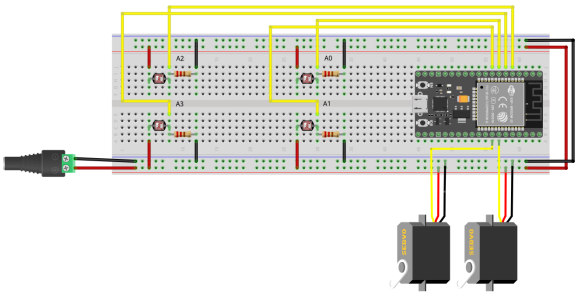


Fig. 5: Esquema do circuito com LDR formando divisores de tensão com resistores de 330 Ω, cujos sinais são captados pelo microcontrolador ESP32. (Imagem de autoria própria)

uma luminosidade desigual em comparação com os do lado esquerdo, o sistema realiza os ajustes necessários para manter o módulo fotovoltaico perpendiculares à fonte de luz solar. A lógica de segurança é integrada por meio da verificação dos estados dos sensores magnéticos para os limites inferior, superior e horizontal. Essa abordagem impede movimentos não autorizados.

B. Programação do seguidor solar

O código de programação do seguidor solar desempenha um papel central, possibilitando o rastreamento do sol e a comunicação externa via MQTT para monitoramento remoto. Ele pode ser consultado no final deste trabalho. A seguir, está a descrição do funcionamento do código:

1) **Configuração Inicial**
   - No início do código, são incluídas bibliotecas essenciais:
     - `ESP32Servo.h`: Controla os servomotores responsáveis pela movimentação do módulo fotovoltaico.
     - `WiFi.h`: Estabelece conexão com a rede Wi-Fi.
     - `PubSubClient.h`: Facilita a comunicação com o broker `MQTT`.
     - `ArduinoJson.h`: Formata os dados no padrão JSON para envio.
   - Os parâmetros de conexão com o servidor `MQTT` são definidos:
     - `const char* mqttServer = "mqtt.thingable.com";`
     - `const int mqttPort = 1883;`
     - `const char* mqttUser = "...";`
     - `const char* mqttPassword = "...";`
   - Também são configurados os pinos dos sensores de luz, dos sensores de fim de curso e dos servomotores.
2) **Conexão com Wi-Fi e MQTT**
   - No `setup()`, o ESP32 realiza a conexão com a rede Wi-Fi utilizando as credenciais fornecidas.
   - Em seguida, o dispositivo estabelece conexão com o broker `MQTT`.
   - Caso a conexão seja perdida durante a execução, a função `reconectabroker()` é responsável por restabelecê-la.
3) **Controle dos Servomotores**
   - Dois servomotores controlam a movimentação do módulo fotovoltaico:
     - **Servo Horizontal**: Controla o movimento para a direita e para a esquerda (`GPIO 18`).
     - **Servo Vertical**: Controla o movimento para cima e para baixo (`GPIO 19`).
   - Os limites físicos dos movimentos são definidos para proteger os componentes:
     - `int servoHorizontalMaxLimit = 180;`
     - `int servoHorizontalMinLimit = 0;`
     - `int servoVerticalMaxLimit = 110;`
     - `int servoVerticalMinLimit = 10;`
   - Os sensores de fim de curso também bloqueiam o movimento quando os limites são atingidos.
4) **Fim de Curso: `allowUpwardMovement` e `allowDownwardMovement`**
   - As variáveis `allowUpwardMovement` e `allowDownwardMovement` garantem que o módulo fotovoltaico não ultrapasse os limites verticais.
   - **Movimento para cima**: Controlado pelo *reed switch* superior:
     - Se o sensor superior for acionado, `allowUpwardMovement` é definido como `false`.
     - Caso contrário, `allowUpwardMovement` é `true`.
   - **Movimento para baixo**: Controlado pelo *reed switch* inferior:
     - Se o sensor inferior for acionado, `allowDownwardMovement` é definido como `false`.
     - Caso contrário, `allowDownwardMovement` é `true`.
   - **Movimento Horizontal:** A lógica se estente igualmente para o movimento horizontal.
5) **Leitura dos Sensores de Luz**
   - O sistema utiliza quatro sensores de luz para determinar a posição do sol:
     - `LDC`: Sensor de luz da direita, parte de cima.
     - `LDB`: Sensor de luz da direita, parte de baixo
     - `LEC`: Sensor de luz da esquerda, parte de cima.
     - `LEB`: Sensor de luz da esquerda, parte de baixo.
   - As leituras são combinadas para calcular as médias:
     - `int ValorSup = (LDC + LEC) / 2;`
     - `int ValorInf = (LDB + LEB) / 2;`
     - `int ValorDir = (LDC + LDB) / 2;`
     - `int ValorEsq = (LEC + LEB) / 2;`
6) **Tolerância (`tol`)**
   - O parâmetro `tol` determina a sensibilidade do seguidor em relação às diferenças entre as leituras:
     - `int tol = 200;`
   - Se a diferença for maior que `tol`, o movimento é acionado; caso contrário, o módulo fotovoltaico não se move.
7) **Lógica de Rastreamento Solar**
   - Se a parte superior estiver mais iluminada, o módulo se move para cima.
   - Se a parte inferior estiver mais iluminada, ele se move para baixo.
   - Se a direita estiver mais iluminada, o módulo se move para a direita.
   - Se a esquerda estiver mais iluminada, ele se move para a esquerda.

8) **Publicação de Dados via MQTT**

- Um `timer` é configurado para enviar os dados dos ângulos em formato `JSON` a cada 10 segundos:

C. Sistema de medição de potência

O microcontrolador ESP32, utilizado neste sistema, não possui capacidade direta para medir corrente em um circuito, limitando-se à medição de tensão. No entanto, a corrente pode ser determinada de forma indireta, utilizando a variação da tensão ao longo do tempo, o que é especialmente útil em sistemas fotovoltaicos, onde a medição da corrente é crucial para avaliar o desempenho e a eficiência do sistema de geração de energia.

A abordagem adotada neste trabalho visa calcular a corrente e, em seguida, a potência com o intuito de comparar a eficiência do sistema fixo e do seguidor solar, utilizando a técnica apresentada por Lopes et al.

No circuito integrado (Fig. 6), o IR2104 é o componente central responsável pelo controle de dois MOSFETs em uma configuração half-bridge. Essa configuração permite gerenciar o fluxo de energia durante o processo de carregamento descarga. O controle dos MOSFETs superior e inferior é feito com base nos sinais recebidos na entrada ”IN” do IR2104.

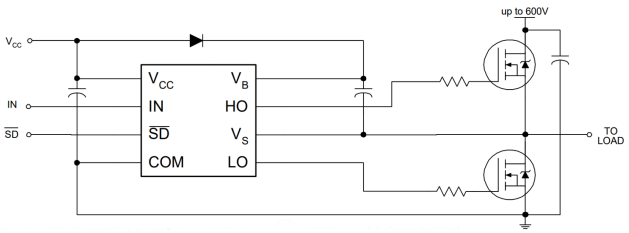


Fig. 6: Configuração típica do IR2104, conforme mostrado no Datasheet IR2104(S) Infineon

A entrada ”IN” é responsável por receber o sinal lógico do microcontrolador ESP32, que determina quando o MOSFET de alta (HO) ou de baixa (LO) tensão deve ser ativado. Quando o MOSFET superior é acionado (HO), a energia flui do módulo fotovoltaico para o capacitor, iniciando o processo de carregamento. Durante essa fase, o valor da tensão no capacitor é monitorado e, a partir disso, pode-se calcular a variação da tensão ao longo do tempo.

A corrente no capacitor pode ser determinada utilizando a equação:

$$I = C \times \frac{dv}{dt}, \tag{1}$$

- I é a corrente no capacitor,
- C é a capacitância do capacitor,
- $\frac{dv}{dt}$ é a taxa de variação da tensão em relação ao tempo.

Essa relação permite obter a curva da corrente em função do tempo. Além disso, a curva da potência no circuito pode ser calculada multiplicando a tensão pela corrente:

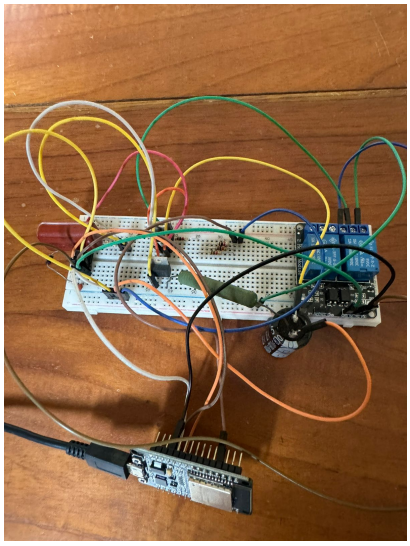


Fig. 7: Circuito utilizado para medir a potência dos módulos fotovoltaicos. (Foto de autoria própria).

$$P = V \times I \tag{2}$$

Ao inverter o estado dos MOSFETs, o capacitor se descarrega através do MOSFET inferior (LO) por meio de uma resistência, preparando o sistema para uma nova medição de potência.

Na imagem, também é possível observar a operação dos diodos de bootstrap, que fornecem a alimentação necessária ao MOSFET superior, além dos resistores que limitam a corrente nas portas dos MOSFETs, controlando a velocidade de comutação dos mesmos. O IR2104 proporciona um controle preciso dessas comutações, garantindo que o capacitor seja carregado e descarregado de maneira eficiente e segura, evitando a condução simultânea dos MOSFETs, o que poderia causar curtos-circuitos.

Como a tensão nominal do módulo fotovoltaico excede o limite máximo de 3,3V permitido para leitura analógica no ESP32, foi implementado um divisor de tensão que reduz a tensão pela metade, ajustando-a a níveis compatíveis com o microcontrolador.

Além disso, foi utilizado um relé para alternar entre as medições dos dois sistemas — o Tracker e o fixo. Primeiramente, os dados de tensão do capacitor são coletados enquanto ele está sendo carregado pelo sistema Tracker. Após o capacitor ser completamente descarregado, o relé é ativado, alterando a conexão para o sistema fixo. Em seguida, o capacitor é recarregado pelo sistema fixo, e o ESP32 coleta a curva de tensão desse processo. Esse procedimento foi realizado com o objetivo de avaliar o ganho proporcionado pelo seguidor solar em comparação ao sistema fixo.

O relé utilizado possui um módulo fotoacoplador, que garante o isolamento entre o circuito de controle e os sistemas de potência. Esse isolamento é importante para evitar interferências e ruídos elétricos, contribuindo para medições

mais precisas e consistentes. Dessa forma, o ESP32 consegue registrar as curvas de tensão de ambos os sistemas.

D. Programação do ESP utilizado para medição de potência.

Um outro ESP32, além do seguidor solar, foi utilizado para coletar dados de tensão durante o carregamento de um capacitor. O funcionamento do código é descrito abaixo e pode ser consultado no final desse trabalho:

1) **Configuração do timer e interrupção:**
   O código utiliza um timer, que define o fator de divisão do clock e ativa a contagem do timer. A interrupção do timer é configurada para disparar a cada intervalo específico, permitindo a coleta de dados com precisão.
2) **Leitura da tensão:**
   A cada interrupção do timer, o código realiza a medição da tensão no pino GPIO 34 utilizando a função `analogRead()`. Os valores de tensão lidos são armazenados no vetor `voltage`. A variável `i` controla o índice do vetor onde os dados são armazenados. A cada interrupção, a leitura é realizada e o índice do vetor é incrementado.
3) **Controle do Relé:**
   O código controla o estado do pino GPIO 12 do ESP32 responsável por acionar o relé. Se um caractere `'H'` for recebido via comunicação serial, esse pino é configurado como `HIGH` (ligado). Se um caractere `'L'` for recebido, o pino é configurado como `LOW` (desligados). Isso possibilita fazer a leitura do sistema fixo ou do tracker a depender do estado desse pino.
4) **Início da coleta de dados:**
   A coleta de dados inicia quando o caractere `'S'` é recebido pela comunicação serial. Nesse momento, o timer é configurado para disparar interrupções com um intervalo pré-estabelecido, começando assim a coleta das leituras de tensão. O processo continua até que todas leituras sejam realizadas.
5) **Processamento da leitura de tensão:**
   Quando todas leituras de tensão são coletadas, o código desativa a interrupção do timer utilizando `timerAlarmDisable()`. Em seguida, os dados armazenados no vetor `voltage` são impressos na comunicação serial e o processo de coleta é reiniciado. A variável `flag2` é usada para indicar que a coleta foi concluída e os dados estão prontos para serem transmitidos via serial.

O MATLAB foi utilizado para interagir com o ESP32, enviando os caracteres H, L, S através da comunicação serial. Uma vez que o processo de coleta foi iniciado, o MATLAB foi responsável por gerenciar a coleta dos dados de tensão provenientes do ESP32, recebendo os valores gerados pelo sistema. Esses dados foram então armazenados em um vetor dentro do MATLAB para posterior processamento.

Após a coleta e organização dos dados, o MATLAB formatou e salvou as informações em um arquivo Excel, permitindo uma análise mais detalhada dos dados coletados. Esse processo de exportação para o formato Excel possibilitou o armazenamento das leituras de forma estruturada, facilitando o acesso e a visualização dos resultados para futuras análises.

E. Validação do Sistema utilizado na medição de potência.

O circuito utilizando o IR2104 foi montado e os dados de tensão e corrente de uma fonte chaveada carregando um capacitor foram coletados tanto no osciloscópio quanto no ESP32 para comparação e validação. Durante o processo de carga do capacitor, a cada 2,5 milissegundos foi coletado um ponto de tensão pelo ESP32, totalizando 100 pontos no total.

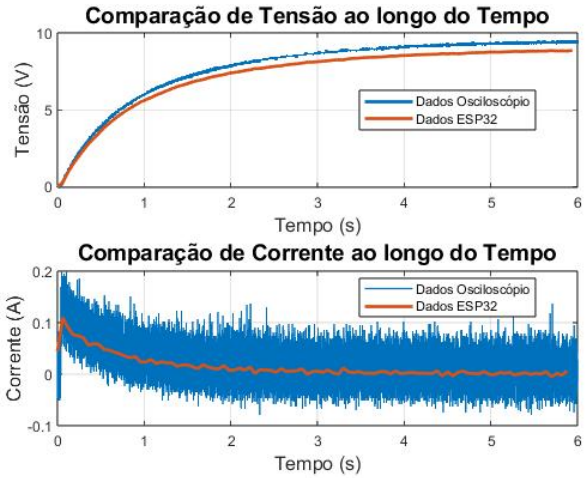


Fig. 8: Comparação dos dados do Osciloscópio e ESP32. (Imagem de autoria própria)

No gráfico de tensão, as medições do ESP32 mostraram uma curva muito próxima à do osciloscópio, com uma tendência de crescimento semelhante ao longo do tempo. No entanto, a tensão medida pelo ESP32 foi ligeiramente inferior à do osciloscópio. Essa diferença pode ser atribuída a calibração não muito precisa do divisor de tensão utilizado na entrada do ESP32.

Em relação ao gráfico de corrente, o ESP32 também mostrou um comportamento muito semelhante ao obtido no osciloscópio, com as curvas acompanhando a mesma tendência ao longo do tempo. Contudo, o osciloscópio apresentou mais ruído nas medições, devido ao uso de um alicate amperímetro, que apresenta pouca precisão em baixas correntes.

No caso do ESP32, ele forneceu leituras de tensão que, ao serem aplicadas à fórmula correspondente, permitiram determinar a corrente. Essas leituras do ESP32 foram mais estáveis, devido à sua baixa taxa de amostragem se comparado ao osciloscópio.

F. Validação dos módulos utilizados.

Foram realizadas medições de potência de dois módulos fotovoltaicos posicionados a 20 graus de inclinação utilizando um sensor digital (Fig. 9) e ambos foram voltados para a mesma direção com o auxílio de uma bússola, com o objetivo

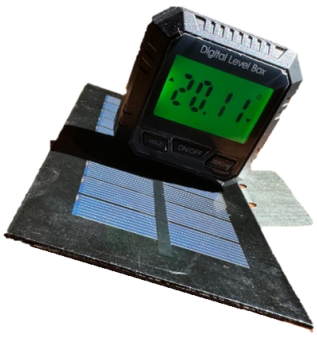


Fig. 9: Medição da inclinação do sistema fixo. (Foto de autoria própria)

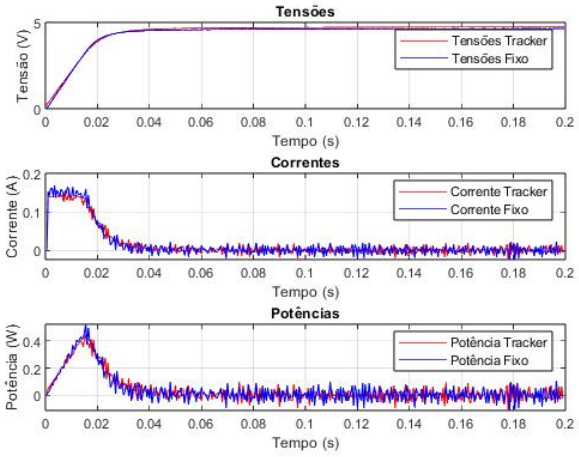


Fig. 10: Gráfico de comparação do módulo fotovoltaico do sistema fixo e do tracker, ambos na mesma orientação. (Imagem de autoria própria)

de verificar se estavam gerando a mesma potência. Os dados foram processados e analisados utilizando o MATLAB.

No primeiro gráfico (Fig.10), Tensões, as curvas indicam que ambos apresentaram comportamentos praticamente idênticos durante a medição, com a tensão estabilizando em valores próximos entre os dois sistemas. O segundo gráfico, Correntes, também confirma que ambos os sistemas seguiram um padrão similar, com a corrente decaindo na mesma proporção ao longo do tempo. Por fim, no terceiro gráfico de Potências, os dois evidenciam uma geração de potência bastante parecida, portanto, as curvas mostram que ambos geraram praticamente a mesma potência ao longo do período de medição.

Esses resultados validam que ambos os módulos fotovoltaicos estão funcionando de forma equivalente, gerando potências muito similares sob as mesmas condições de orientação e inclinação. Com essa validação inicial, o sistema pode agora ser utilizado para uma comparação mais abrangente da eficiência entre o sistema de rastreamento (Tracker) e o fixo.

G. Ajuste polinômial.

Em diversas áreas da ciência e da engenharia, é comum a necessidade de ajustar uma função analítica a um conjunto de dados experimentais ou observados. Segundo Boldrini et al. o processo de ajuste deve levar em consideração dois aspectos fundamentais:

i) Qualquer medida contém um erro, seja ele decorrente do aparelho de medição, falha do operador ou limitações experimentais.

ii) Pode existir um argumento teórico ou de bom senso que indique o formato ou aspecto analítico da função a ser utilizada.

O primeiro aspecto (*i*) nos alerta que exigir que a curva ajustada passe exatamente por todos os pontos pode não ser apenas desnecessário, mas também incorreto. Já o segundo aspecto (*ii*) reforça a importância de se considerar argumentos teóricos ou práticos que permitam prever o comportamento dos dados. Assim, o ajuste polinomial busca encontrar uma função que represente, de forma satisfatória, a tendência dos dados, minimizando os erros e garantindo previsões confiáveis.

Matematicamente, o ajuste polinomial busca determinar uma função $P(x)$ da forma:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \qquad (3)$$

onde n é o grau do polinômio e $a_n, a_{n-1}, \ldots, a_0$ são os coeficientes ajustados. Esses coeficientes são calculados de maneira a minimizar a diferença entre os valores originais e os valores previstos pela função polinomial. Geralmente, o método dos mínimos quadrados é utilizado para esse ajuste, minimizando a diferenças entre os valores observados e os ajustados.

Neste estudo, o software MATLAB foi utilizado para realizar o ajuste polinomial. O MATLAB dispõe de funções específicas para esse propósito, como `polyfit` e `fit`, que permitem ajustar os dados a um polinômio de grau n. O procedimento inclui os seguintes passos:

1) Os valores dos dados medidos são fornecidos como entrada para a função de ajuste.
2) A função calcula os coeficientes $a_n, a_{n-1}, \ldots, a_0$ do polinômio, utilizando o método dos mínimos quadrados.
3) Uma curva ajustada é gerada, representando os valores previstos pelo polinômio para os mesmos pontos dos dados originais.
4) Essa curva é então comparada com os dados reais, e a qualidade do ajuste é avaliada por meio do coeficiente de determinação (R^2).

O coeficiente de determinação (R^2) é utilizado para medir a qualidade do ajuste de um modelo aos dados observados. Sua definição matemática é dada por:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \qquad (4)$$

onde:

- SS_{res} é a soma dos quadrados dos resíduos, calculada como:

$$SS_{res} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5}$$

representando a soma das diferenças ao quadrado entre os valores observados (y_i) e os valores previstos ($\hat{y}_i$).

- SS_{tot} é a soma dos quadrados totais, definida como:

$$SS_{tot} = \sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{6}$$

representando a variação total nos dados em relação à média ($\bar{y}$).

A interpretação do R^2 é:

- $R^2 = 1$: Indica um ajuste perfeito, onde o modelo explica 100% da variação nos dados.
- $R^2 = 0$: O modelo não explica nenhuma variação nos dados; a previsão é equivalente à média dos valores observados.
- $R^2 < 0$: O modelo ajustado é pior do que simplesmente usar a média.

III. Resultados

A seguir (Fig. 11), apresentam-se gráficos comparativos das curvas originais e ajustadas durante um ciclo de carregamento do capacitor. Para esse exemplo foi utilizado um polinômio de grau 8, onde o coeficiente R^2 atingiu 0,99938. Esses gráficos permitem avaliar visualmente a proximidade entre os valores ajustados e os observados.

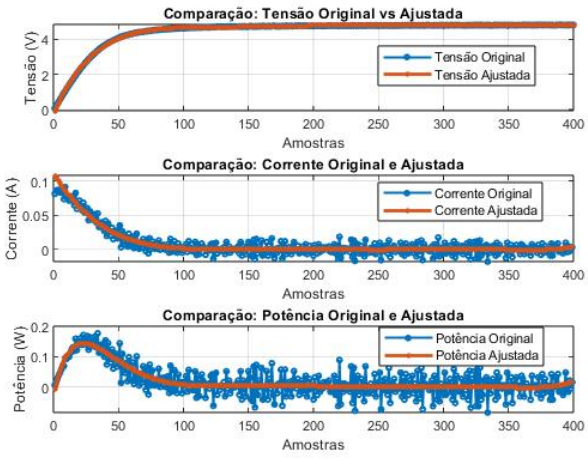


Fig. 11: Exemplo gráfico da comparação de uma função com o ajuste polinômial e sem o ajuste. (Imagem de autoria própria)

Após realizadas todas as validações, foram coletadas 80 curvas de carregamento do capacitor durante o dia, com um intervalo de 6 minutos entre cada medição. Este procedimento foi realizado tanto para o sistema fixo, direcionado para o norte com 20 graus de inclinação, quanto para o sistema tracker.

Os dados obtidos das curvas foram ajustados por meio do ajuste polinomial, permitindo identificar, desprezando o ruído, o ponto de máxima potência que corresponde à potência gerada pelo módulo naquele momento. Com isso, foi possível traçar um gráfico com os pontos de máxima potência, comparando o desempenho dos dois sistemas.

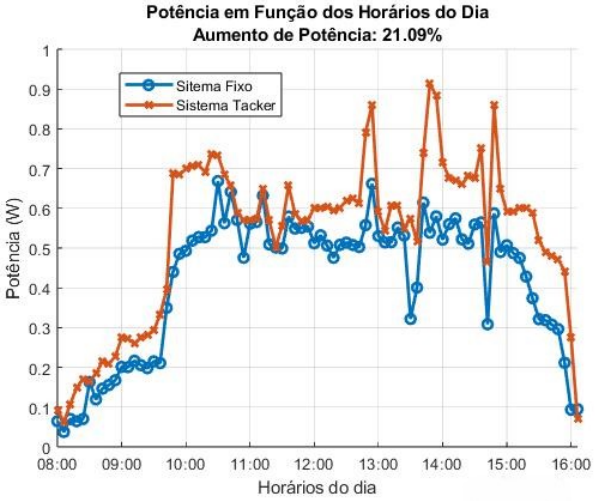


Fig. 12: Gráfico da comparação de potência do sistema fixo e do sistema tracker ao longo do dia. (Imagem de autoria própria)

O gráfico resultante (Fig. 12) evidencia os pontos de máxima potência coletados, tanto para o sistema fixo quanto para o sistema tracker, possibilitando uma análise comparativa do desempenho energético entre os dois métodos. Observa-se que o sistema com tracker (representado pela linha laranja) apresenta uma geração de potência significativamente superior ao sistema fixo (linha azul) em grande parte do dia. Isso ocorre porque o sistema com seguidor ajusta continuamente o ângulo do módulo fotovoltaico, acompanhando o movimento do sol e, assim, otimizando a captação de radiação solar.

Em contraste, o sistema fixo, que mantém o módulo em uma posição estática, exibe uma geração de potência mais constante ao longo do dia. Embora este sistema apresente picos de potência em horários próximos ao meio-dia, sua performance não atinge os valores máximos observados no seguidor solar, que se adapta ao longo das horas.

Através da área do gráfico (Fig. 12), foi possível constatar um **aumento de potência de 21,09%** proporcionado pelo sistema com tracker, conforme indicado no gráfico, reflete sua superioridade em termos de eficiência.

O gráfico de inclinação horizontal e vertical do seguidor solar (Fig. 13), revela a dinâmica de movimentação do sistema ao longo do dia. Os valores de inclinação horizontal (representados pela linha azul) e inclinação vertical (representada pela linha verde) demonstram como o seguidor ajusta a posição para maximizar a captação de luz solar.

Esses ajustes são fundamentais para garantir o desempenho otimizado do sistema com tracker, refletindo diretamente na maior geração de potência em comparação com o sistema fixo. É importante destacar que em nenhum momento o módulo fotovoltaico atingiu os fins de curso.

Fig. 13: Inclinação Vertical e Horizontal do Sistema Tracker. (Imagem de geração própria, através da interface web Thingable)

IV. Conclusão

A implementação de um seguidor solar de eixo duplo demonstrou ser uma solução eficiente para maximizar a captação de energia em sistemas fotovoltaicos. Por meio de um controle preciso dos ângulos de inclinação, foi possível registrar um aumento de 21,09% na geração de potência em comparação a sistemas fixos. Apesar de apresentar desafios relacionados a custos iniciais e manutenção, a superioridade do sistema com seguidor em termos de eficiência energética reforça sua relevância na busca por soluções sustentáveis para o setor de energia. Os resultados obtidos validam o potencial dessa tecnologia e abrem caminho para estudos futuros focados na otimização de desempenho e redução de custos.

REFERÊNCIAS

[1] International Energy Agency. (2024). Massive expansion of renewable power opens door to achieving global tripling goal set at COP28. Disponível em: https://www.iea.org/news/massive-expansion-of-renewable-power-opens-door-to-achieving-global-tripling-goal-set-at-cop28. Acesso em: 10 dez. 2024.

[2] L. J. de Bessa Neto, M. R. B. G. Vale, e F. K. O. M. Varella Guerra, ”Desenvolvimento de Rastreador Solar Didático de um Eixo para Painéis Fotovoltaicos,” em *VIII Congresso Brasileiro de Energia Solar (CBENS)*, Fortaleza, Brasil, jun. 2020.

[3] Lewandoski, C. F., Santos, R. F., Sio, J. P. M. K., Cruz, F. G. da, and Ikpehai, A. (2022). Sistema de rastreador solar de eixo simples baseado em intensidade solar e desempenho. *International Journal of Environmental Resilience Research and Science*, 4(2), 1–11. Disponível em: https://e-revista.unioeste.br/index.php/ijerrs/article/view/28751. Acesso em: 23 jan. 2025.

[4] Cortez, R. J. M. (2013). *Sistema de seguimento solar em produção de energia fotovoltaica*. Dissertação (Mestrado) – Curso de Engenharia Elétrica, Universidade de Porto.

[5] INFINEON TECHNOLOGIES. IR2104(S) & (PbF) Datasheet. El Segundo, CA: Infineon Technologies, 2004. Disponível em: www.irf.com. Acesso em: 10 dez. 2024.

[6] Boldrini, J. L., Costa, S. L. R., Figueiredo, V. L., & Wetzler, H. G. (1986). *Álgebra Linear* (3ª ed.). São Paulo: Editora Harbra.

[7] Villalva, M. G., & Gazoli, J. R. (2012). *Energia Solar Fotovoltaica: Conceitos e Aplicações*. São Paulo: Érica.

[8] Pinto Neto, A. F. C., Macagnan, M. H., Zilles, R., and Lehmann, J. B. (2010). *Descrição de seguidores solares e sua aplicação em centrais fotovoltaicas conectadas à rede*. In: III Congresso Brasileiro de Energia Solar, 21-24 set. 2010, Belém - PA.

[9] J. I. L. Seguel, “Projeto de um sistema fotovoltaico autônomo de suprimento de energia utlizando técnica MPPT e controle digital.,” Dissertação de Mestrado Apresentado na Universidade Federal de Minas Gerais, UFMG., Belo Horizonte-MG, 2009

[10] Lopes, L. A., Silva Junior, D. C., Dardengo, V. P., Pereira, H. A., Cupertino, A. F., & Brito, E. M. S. (2024). Desenvolvimento de um caracterizador de módulos fotovoltaicos utilizando plataforma WEB. In *XXV Congresso Brasileiro de Automática*, Rio de Janeiro, 2024.

```
#include <ESP32Servo.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

const char* mqttServer = "mqtt.thingable.com";
const int mqttPort = 1883;
const char* mqttUser = "...";
const char* mqttPassword = "...";
const char* clientID = "...";
const int oneWireBus = 4;

WiFiClient espClient;
PubSubClient client(espClient);
char mensagem[300];
const char* ssid = "...";
const char* password =  "...";
const int JSON_BUFFER_SIZE = 1024;
StaticJsonDocument<JSON_BUFFER_SIZE> doc;

hw_timer_t * timer = NULL;

Servo horizontalServo;
Servo verticalServo;
int servoHorizontalPosition = 95;
int servoVerticalPosition = 60;

int LDC;
int LDB;
int LEC;
int LEB;

float anguloH;
float anguloV;
float servoHorizontalAngle;
float servoVerticalAngle;
int flag1;

int servoHorizontalMaxLimit = 180;
int servoHorizontalMinLimit = 0;
int servoVerticalMaxLimit = 110;
int servoVerticalMinLimit = 10;

const int reedSwitchUpperPin = 14;
const int reedSwitchLowerPin = 27;
const int reedSwitchRightHorizontalPin = 25;
const int reedSwitchLeftHorizontalPin = 26;

bool allowUpwardMovement = true;
bool allowDownwardMovement = true;
bool allowRightMovement = true;
bool allowLeftMovement = true;

void onTimer() {
  flag1 = flag1 + 1;}

void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Iniciando conexao com a rede WiFi...");}
  Serial.println("Conectado na rede WiFi!");
  client.setServer(mqttServer, mqttPort);
  client.setCallback(callback);

  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &onTimer, true);
  timerAlarmWrite(timer, 10000000, true);
  timerAlarmEnable(timer);

  horizontalServo.attach(18);
  verticalServo.attach(19);

  pinMode(reedSwitchLowerPin, INPUT);
  pinMode(reedSwitchUpperPin, INPUT);
  pinMode(reedSwitchRightHorizontalPin, INPUT);
  pinMode(reedSwitchLeftHorizontalPin, INPUT);

  horizontalServo.write(servoHorizontalPosition);
  verticalServo.write(servoVerticalPosition);}

void loop() {
  if (flag1 >= 1) {
    anguloH = -(servoHorizontalAngle - 90);
    anguloV = -(servoVerticalAngle - 60);
    sprintf(mensagem, "{\"INC1\":%.3f,\"INC2\":%.3f}", anguloH, anguloV);
    client.publish("Thingable/delufv/Solar_tracker/solar_tracker/evt", mensagem
        );
    Serial.println(mensagem);
    flag1 = 0;}
  if (!client.connected()) {
    reconectabroker();}
  client.loop();
  allowDownwardMovement = digitalRead(reedSwitchLowerPin) != LOW;
  allowUpwardMovement = digitalRead(reedSwitchUpperPin) != LOW;
  allowRightMovement = digitalRead(reedSwitchRightHorizontalPin) != LOW;
  allowLeftMovement = digitalRead(reedSwitchLeftHorizontalPin) != LOW;
  servoHorizontalAngle = servoHorizontalPosition;
  servoVerticalAngle = servoVerticalPosition;

  LDC = analogRead(32);
  LDB = analogRead(33);
  LEC = analogRead(34);
  LEB = analogRead(35);

  int tol = 200;

  int ValorSup = (LDC + LEC) / 2;
  int ValorInf = (LDB + LEB) / 2;
  int ValorDir = (LDC + LDB) / 2;
  int ValorEsq = (LEC + LEB) / 2;
  int DifSupInf = ValorSup - ValorInf;
  int DifDirEsq = ValorDir - ValorEsq;

  if (abs(DifSupInf) > tol) {
    if (ValorSup > ValorInf && allowUpwardMovement) {
      servoVerticalPosition = min(servoVerticalPosition + 1,
          servoVerticalMaxLimit);
    } else if (ValorSup < ValorInf && allowDownwardMovement) {
      servoVerticalPosition = max(servoVerticalPosition - 1,
          servoVerticalMinLimit);}
    verticalServo.write(servoVerticalPosition);
    delay(100);}

  if (abs(DifDirEsq) > tol) {
    if (ValorDir > ValorEsq && allowRightMovement) {
      servoHorizontalPosition = max(servoHorizontalPosition - 1,
          servoHorizontalMinLimit);
    } else if (ValorDir < ValorEsq && allowLeftMovement) {
      servoHorizontalPosition = min(servoHorizontalPosition + 1,
          servoHorizontalMaxLimit);}
    horizontalServo.write(servoHorizontalPosition);
    delay(100);}
  delay(100);}

void reconectabroker() {
  while (!client.connected()) {
    Serial.println("Conectando ao broker MQTT...");
    if (client.connect(clientID, mqttUser, mqttPassword)) {
      client.subscribe("Thingable/delufv/Solar_tracker/solar_tracker/evt");
      Serial.println("Conectado ao broker!");
    } else {
      Serial.print("Falha na conexao ao broker - Estado: ");
      Serial.println(client.state());}}}
```

Programação 1: Código do rastreador solar utilizando ESP32

```
hw_timer_t *timer = NULL;
int i = 0;
int flag2;
int pinVc = 34;
int voltage[400] = {0};

void onTimer() {
  if (i < 400) {
    digitalWrite(15, HIGH);
    voltage[i] = analogRead(pinVc);
    i++;
  } else {
    digitalWrite(15, LOW);
    i = 0;
    flag2 = 1;
  }
}

void setup() {
  Serial.begin(9600);
  pinMode(15, OUTPUT);
  pinMode(12, OUTPUT);
  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &onTimer, true);
}

void loop() {
if (Serial.available() > 0) {
char receivedChar = Serial.read();

if (receivedChar == 'S') {
  timerAlarmWrite(timer, 2500, true);
  timerAlarmEnable(timer);
}

else if (receivedChar == 'H') {
  digitalWrite(12, HIGH); // Configurar GPIO como HIGH
}

else if (receivedChar == 'L') {
  digitalWrite(12, LOW); // Configurar GPIO como LOW
}
  }

  if (flag2 == 1) {
    timerAlarmDisable(timer);
    Serial.println("Leituras de tensao:");
    for (int i2 = 0; i2 < 400; i2++) {
      Serial.print(voltage[i2]);
      Serial.print(" ");
    }
    flag2 = 0;
  }
}
```

Programação 2: Código para leitura de valores de tensão usando ESP32