Relatório do Projeto: Solar Tracker

Igor Cleto e Matheus Galbiatti

Belo Horizonte, 10 de julho de 2025



**Universidade Federal de Minas Gerais**

**Escola de Engenharia**

**Disciplina: Projeto de Sistemas Embutidos**

**Relatório do Projeto: Solar Tracker**

**Autores:**

Igor Cleto

Matheus Galbiatti

Belo Horizonte

10 de julho de 2025

# Introdução

Este documento detalha o projeto e desenvolvimento de um protótipo de rastreador solar de eixo único (azimutal), concebido no âmbito da disciplina de Projeto de Sistemas Embutidos. O objetivo principal é criar uma solução de baixo custo e autossustentável para otimizar a captação de energia solar, abordando a limitação de eficiência dos painéis solares fixos. O projeto foi planejado utilizando a metodologia do Project Model Canvas para estruturar as ideias e garantir o alinhamento entre os objetivos, requisitos e restrições.

# Project Model Canvas

O Project Model Canvas é uma ferramenta de gerenciamento visual utilizada para descrever, projetar e analisar modelos de projetos de forma concisa e integrada. Ele é composto por 13 blocos que cobrem as áreas fundamentais de um projeto, desde suas justificativas e objetivos até os custos e cronograma. A seguir, cada um dos blocos é detalhado conforme definido para o projeto Solar Tracker.

## Justificativas

Painéis solares fixos apresentam uma limitação intrínseca na captação de energia, pois não acompanham o movimento aparente do sol, resultando em menor eficiência ao longo do dia.

## Produto

Sistema de otimização para a captação de energia solar, que opera por meio do acompanhamento do azimute solar, sendo complementado pela documentação técnica.

## Objetivo SMART

Desenvolver um protótipo funcional de um sistema de rastreamento solar de eixo único (azimutal) para aumentar a captação de energia de um painel solar de maneira autossustentável.

## Requisitos

* **Rastreamento Solar Autônomo:** Detectar e seguir autonomamente a luz solar.
* **Movimentação:** Em um eixo (azimutal).
* **Simplicidade:** Componentes de hardware e software simples, com uma interface de usuário (IU) intuitiva.
* **Desempenho Operacional:** Operação confiável e ágil.
* **Restrição Orçamentária:** Desenvolvimento e operação dentro de um orçamento limitado.

## Benefícios Futuros

O produto aprimora a eficiência energética e otimiza a utilização do espaço, enquanto o monitoramento remoto e uma solução de rastreamento solar de baixo custo complementam seus benefícios. O resultado é a otimização da eficiência, atendendo à necessidade de soluções energéticas práticas e eficazes.

## Stakeholders e Fatores Externos

O projeto é influenciado por fornecedores de componentes eletrônicos e mecânicos (disponibilidade e preço), o ambiente climático (condições para testes) e potenciais usuários/avaliadores de protótipos na área de energia solar.

## Equipe

* Igor Cleto: Validação do projeto.
* Matheus Galbiatti: Desenvolvimento do projeto.

## Premissas

* Disponibilidade contínua dos componentes críticos no mercado.
* Condições de insolação solar consistentes e adequadas durante a fase de testes.
* Viabilidade de um design de sistema que assegure um consumo energético inferior ao ganho obtido pela captação solar.

## Riscos

* Indisponibilidade de componentes críticos.
* Potenciais atrasos no cronograma estabelecido de 8 semanas.
* Condições climáticas desfavoráveis que prejudiquem os testes planejados.
* Consumo energético do sistema que supere os benefícios da captação solar.

## Grupo de Entregas

1. Design e planejamento detalhado.
2. Desenvolvimento do protótipo de hardware e software.
3. Testes e refinamentos.
4. Protótipo funcional e documentação final.

## Linha do Tempo

* **Semanas 1 e 2:** Design e planejamento.
* **Semanas 3 a 5:** Desenvolvimento do protótipo.
* **Semanas 6 e 7:** Testes e refinamentos.
* **Semana 8:** Finalização.

## Custos

O projeto conta com um orçamento fixo de R$300,00, destinado exclusivamente aos componentes eletrônicos e mecânicos necessários para o protótipo.

## Restrições

* Testes do protótipo excluem condições climáticas extremas.
* Os membros da equipe vão dedicar 5 horas por semana para o desenvolvimento e acompanhamento do projeto.

# Hardware e Componentes

# Backend

# Frontend com Streamlit

O frontend do projeto é um dashboard interativo desenvolvido integralmente em Python com a biblioteca Streamlit. A escolha desta tecnologia foi estratégica, visando acelerar o ciclo de desenvolvimento ao eliminar a necessidade de linguagens de frontend tradicionais (HTML, CSS, JavaScript) e facilitar a integração direta com a lógica de controle e análise de dados do projeto.

## Arquitetura e Fluxo de Dados

A aplicação Streamlit atua como o centro de controle e monitoramento do Solar Tracker. Sua arquitetura é baseada em um fluxo de dados reativo e em tempo real, sustentado por três tecnologias principais.

* **Streamlit:** É o framework que renderiza a interface web. A cada interação do usuário ou recebimento de novos dados, o Streamlit re-executa o script Python para atualizar a interface, garantindo que as visualizações reflitam sempre o estado mais recente do sistema.
* **Paho-MQTT:** A comunicação com o hardware (ESP32) é desacoplada e realizada através do protocolo MQTT. A aplicação se conecta a um broker EMQX Cloud e opera de duas formas:
  + **Subscrição:** Inscreve-se no tópico esp32/angulo para receber, de forma assíncrona, as atualizações de posição do painel solar.
  + **Publicação:** Publica mensagens no tópico esp32/comando para enviar instruções ao microcontrolador, como a mudança de modo de operação ou a definição de um ângulo manual.
* **Pandas:** É utilizado para a gestão dos dados históricos. Cada ângulo recebido via MQTT é adicionado a um DataFrame do Pandas, que é então salvo em um arquivo CSV (dados\_historicos.csv). Isso garante a persistência dos dados entre sessões da aplicação.

## Gestão de Estado e Comunicação Assíncrona

Um dos desafios em aplicações Streamlit que mantêm conexões persistentes é a gestão de estado. O Streamlit re-executa o script a cada interação, o que poderia levar à perda de estado e múltiplas reconexões. Para contornar isso, a aplicação utiliza o objeto st.session\_state:

* st.session\_state.mqtt\_client: Armazena a instância do cliente MQTT, garantindo que a conexão com o broker seja estabelecida apenas uma vez e persista durante toda a sessão do usuário.
* st.session\_state.dados\_atuais: Um dicionário que guarda o valor mais recente do ângulo recebido, permitindo que seja exibido na interface.
* st.session\_state.historico: Mantém o DataFrame do Pandas com todos os dados históricos carregados ou coletados durante a sessão.

A comunicação MQTT é tratada de forma assíncrona. O cliente MQTT é iniciado com client.loop\_start(), que cria um thread em segundo plano para lidar com a rede, permitindo que a aplicação Streamlit continue responsiva. A função de callback on\_message é acionada sempre que um novo dado de ângulo chega. Dentro desta função, o estado da sessão é atualizado e o comando st.experimental\_rerun() é chamado para forçar a re-execução do script e, consequentemente, a atualização da interface com o novo dado.

## Detalhes da Interface de Usuário

A interface foi projetada para ser clara e funcional, dividida em um painel de controle e um dashboard principal.

### Painel de Controle (Sidebar)

A barra lateral (st.sidebar) agrupa os controles interativos:

* **Modo de Operação:** Utiliza o componente st.radio com a opção horizontal=True para uma seleção compacta. A função de callback on\_change é associada a este componente, garantindo que a mudança de modo seja enviada ao hardware imediatamente após a seleção do usuário, proporcionando feedback instantâneo.
* **Controle Manual:** Condicionalmente visível apenas no modo "Manual", este controle utiliza um st.slider para uma seleção de ângulo intuitiva. O comando só é enviado ao clicar no st.button, evitando o envio contínuo de mensagens MQTT enquanto o usuário arrasta o controle.

### Dashboard Principal

A área principal foca na visualização dos dados:

* **Status Atual:** O componente st.metric oferece uma visualização proeminente e clara do ângulo atual, formatado para duas casas decimais.
* **Gráfico Histórico:** O st.line\_chart renderiza o histórico de ângulos de forma nativa e otimizada, utilizando o timestamp como índice para o eixo X.
* **Tabela de Dados:** O uso do st.expander permite que o usuário escolha se deseja ver a tabela completa de dados históricos (st.dataframe), evitando sobrecarregar a interface principal, mas mantendo os dados detalhados acessíveis.

# Conclusão