Painel de Configurações PID - Plataforma de Stewart

Visão Geral

`settings.html` oferece uma tela enxuta para visualizar o estado da conexão e ajustar rapidamente os parâmetros de controle PID aplicados no firmware (ganhos Kp/Ki/Kd por pistão, valores globais, deadband em mm e PWM mínimo). O script `settings.js` consome os endpoints `/pid/gains` e `/pid/settings` do backend FastAPI e fornece feedback imediato via Toastify.

Características

- Status da porta serial compartilhado com o restante da interface: o painel mostra LED, texto e porta detectada sem duplicar lógica (consulta periódica a `/serial/status`).
- Grade responsiva com cartões para cada pistão (Kp/Ki/Kd + botão “Aplicar”).
- Bloco “Todos os Pistões” para replicar um conjunto de ganhos em uma única chamada.
- Sessão “Parâmetros Globais” com campos para `dbmm` (deadband) e `minpwm`.
- Toasts coloridos para sucesso/erro e validação básica (parseFloat/parseInt).
- Carregamento automático dos valores atuais ao abrir a página.

Arquitetura

Backend (`interface/backend/app.py`)

- `GET /serial/status`: informa `connected` e `port`. O painel usa para exibir o LED e sincronizar `localStorage`.
- `GET /pid/gains`: devolve um dicionário `{1: {kp,ki,kd}, ...}` usado para preencher todos os inputs.
- `POST /pid/gains`: recebe `{ piston, kp, ki, kd }` para atualizar apenas um eixo.
- `POST /pid/gains/all`: aceita query parameters `kp`, `ki`, `kd` e replica para os seis pistões.
- `GET /pid/settings` / `POST /pid/settings`: leitura/escrita de `dbmm` e `minpwm`.

Frontend (`settings.html` + `settings.js`)

- `updateConnectionStatus()`:
  - Chama `/serial/status` a cada 2 s.
  - Atualiza `status-indicator`, texto e porta.
  - Replica o estado no `localStorage` (para outras páginas usarem).
- `loadPIDGains()`:
  - `fetch('/pid/gains')`.
  - Preenche inputs `kp-x`, `ki-x`, `kd-x`.
  - Também usa os valores do pistão 1 para popular a seção “Todos os Pistões”.
- `loadPIDSettings()`:
  - `fetch('/pid/settings')`.
  - Atualiza inputs `dbmm` e `minpwm`.
- `sendGainsInd(piston)` / `sendGainsAll()`:
  - Realizam POST com JSON ou query string.

    - Mostram toasts de sucesso/erro conforme retorno.
- `sendSettings()`:
    - Envia JSON `{"dbmm": ..., "minpwm": ...}` e confirma com toast.

Fluxo de Funcionamento

1. `DOMContentLoaded` → chama `updateConnectionStatus`, `loadPIDGains`, `loadPIDSettings`.
2. Usuário altera um ganho e pressiona “Aplicar” → `sendGainsInd` faz POST e aguarda resposta.
3. Toasts informam se o backend aceitou ou rejeitou valores.
4. A cada 2 s o painel checa o status da porta (mantém LED atualizado mesmo que outra página conecte/desconecte).

Passo a Passo de Uso

1. Certifique-se de que o backend está em execução e o ESP32 conectado em alguma página (controller, actuators etc.) – o status será refletido aqui.
2. Abra `settings.html`.
3. Revise os valores carregados automaticamente em cada cartão.
4. Para atualizar um pistão:
    - Edite Kp/Ki/Kd.
    - Clique em “Aplicar”.
5. Para copiar ganhos iguais para todos:
    - Ajuste o bloco “Todos os Pistões”.
    - Clique em “Aplicar em Todos”.
6. Ajuste `dbmm` ou `minpwm` conforme a necessidade e clique em “Aplicar Ajustes”.

Configurações Avançadas

- **Intervalo de polling**: mude o `setInterval(updateConnectionStatus, 2000)` para deixar o painel mais responsivo (ou economizar CPU).
- **Valores padrão**: os inputs começam com valores “dummy” (2.0/0.0). Pode-se alterar no HTML para refletir presets de fábrica.
- **Validação extra**: hoje o script apenas converte para `parseFloat/parseInt`. Adicione checagens e máscaras se desejar restringir faixas específicas.
- **Internacionalização**: a formatação de números usa o padrão americano (`.` como separador decimal). Caso queira vírgula, inclua um conversor no `sendGains*`.

Troubleshooting

- **Toasts de erro “Erro ao aplicar ganhos”**:
    - Verifique se algum campo ficou vazio (retorna `NaN` no JSON).
    - Veja o terminal do backend para mensagens adicionais (ex.: pistão inexistente).
- **Valores não atualizam após salvar**:
    - Recarregue a página ou clique em algum botão “Aplicar” novamente – o backend pode ter ignorado por validação.
    - Confirme se o backend caiu (HTTP 500/422).
- **LED sempre vermelho**:

    - Mesmo com outra página conectada, se o backend estiver fora do ar o fetch falha → o script assume desconectado. Verifique `app.py`.
- **Inputs resetando sozinhos**:
    - `loadPIDGains` roda apenas no load inicial; se o backend enviar strings, podem aparecer valores `undefined`. Ajuste o endpoint para sempre mandar números.

Autor

Guilherme Miyata - Instituto Federal de São Paulo (IFSP)
Trabalho de Conclusão de Curso - 2025

Github
Linkedin
Portfólio

Última atualização: Novembro 2025