Controller 2 - Controle por Joystick Minimalista

Visão Geral

`joystick.html` oferece uma versão leve do controle por joystick da plataforma de Stewart. A tela prioriza leitura rápida dos eixos, status do gamepad e comprimentos calculados dos pistões, sem carregar a cena 3D. O script `joystick.js` conversa diretamente com `joystick-control.js` e com os utilitários globais de serial (`common.js`) para publicar poses na API FastAPI.

Características

- Interface focada em operação: cards para dobrar/ativar joystick e aplicar no hardware com avisos claros.
- Atualização automática dos valores de X/Y/Z e Roll/Pitch/Yaw conforme o movimento dos sticks.
- Pré-verificação de curso através do endpoint `/calculate` para colorir os cartões dos pistões.
- Indicadores ativos de porta serial, gamepad conectado e modo hardware.
- Reuso do backend do joystick completo (`/joystick/pose`), porém sem renderização em Three.js → ideal para notebooks modestos.

Arquitetura

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

- `POST /serial/ports`, `/serial/open`, `/serial/close`, `/serial/status`: gestão da porta USB do ESP32.
- `POST /joystick/pose`: recebe eixos normalizados, converte para pose física, valida cinemática e envia comando serial quando `apply=true`.
- `POST /calculate`: serviço reutilizado para obter os comprimentos absolutos dos atuadores e indicar se a pose está dentro do curso 500–680 mm.

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

- Importa `initJoystickControl` de `joystick-control.js` para encapsular toda a lógica Gamepad API.
- Usa `common.js` para carregar portas, conectar/desconectar a serial, controlar toasts e manter `serialConnected`.
- Funções relevantes:
  - `initJoystick()`: inicia o controlador, define callbacks `onPoseChange` e `onError`.
  - `updatePoseUI(pose)`: sincroniza displays numéricos + sliders (somente leitura).
  - `calculateAndUpdatePistons(pose)`: faz `POST /calculate` e colore os cartões (verde = válido, vermelho = fora do curso).
  - `registerEventListeners()`: liga checkboxes de joystick/hardware, listeners de gamepad e controles seriais.
  - `checkGamepads()`: consulta `navigator.getGamepads()` para atualizar o selo “Conectado”.

Componentes da Página

1. **Status da Conexão** – LED e porta exibidos via `setSerialStatus`.
2. **Cartão "Controle por Joystick"** – mostra gamepad ativo, toggles "Ativar Joystick" e "Aplicar no Hardware".
3. **Checklist Operacional** – lembretes de segurança (conectar serial, testar sem hardware, etc.).
4. **Valores Atuais da Pose** – grade de seis sliders desabilitados exibindo `pose` calculada.
5. **Comprimentos dos Pistões** – seis cartões com `Pistão N` e valor em mm + borda colorida conforme `valid`.

Fluxo de Funcionamento

1. Ao carregar a página:
   - `loadSerialPorts()` popula o select.
   - `checkExistingConnection()` restaura sessões abertas.
   - `initJoystick()` cria um `JoystickController` e liga callbacks.
2. Quando um gamepad é detectado e o usuário marca "Ativar Joystick":
   - `setEnabled(true)` inicia loops de leitura (20 Hz para API, ~60 FPS para preview).
   - Cada variação de eixo dispara `onPoseChange`.
3. `onPoseChange` → `updatePoseUI` + `calculateAndUpdatePistons`.
4. Caso "Aplicar no Hardware" esteja marcado, `joystick-control.js` envia `apply=true` e os comandos `spmm6x` chegam ao ESP32.
5. Saiu da página? `beforeunload` chama `destroy()` e limpa checkboxes para evitar drift.

Passo a Passo de Uso

1. Rodar o backend (`python app.py`) e abrir `joystick.html`.
2. Conectar o ESP32:
   - Clique em "Atualizar", escolha a porta e aperte "Conectar".
3. Conectar o joystick e movimentar um botão para o navegador reconhecê-lo.
4. Marcar "Ativar Joystick" e observar os valores de pose mudarem.
5. Se desejar mexer o hardware:
   - Marcar "Aplicar no Hardware".
   - Monitorar os cartões dos pistões → se algum ficar vermelho, ajuste os sticks para voltar às faixas seguras.
6. Desmarcar os toggles ou desconectar o controle ao finalizar.

Configurações Avançadas

- **Limites de sliders**: altere `min/max` dos `input type="range"` em `joystick.html` para refletir novos envelopes (e.g., ±15 mm).
- **Parâmetros do joystick**: `joystick-control.js` define `MAX_TRANS_MM`, `MAX_ANGLE_DEG`, `DEADZONE` e `UPDATE_RATE_MS`.
- **Z base**: passado para `initJoystickControl({ zBase: 500 })`. Mude para adequar a altura da bancada.
- **Polling serial**: `updateConnectionStatus` roda a cada 2 s; ajuste o `setInterval` se precisar de refresh mais rápido/lento.

Troubleshooting

- **“Nenhum gamepad detectado”**:
  - Confirme se algum botão foi pressionado (necessário para o navegador listar).
  - Verifique o console → `navigator.getGamepads()` pode estar retornando vazio em alguns browsers (prefira Chrome).
- **Cartões dos pistões permanecem “--”**:
  - O endpoint `/calculate` pode ter falhado; veja logs no backend.
  - Certifique-se de que os valores de pose não são `NaN`.
- **Não envia comandos ao hardware**:
  - Garanta que a serial esteja conectada (LED verde).
  - Marque “Aplicar no Hardware”.
  - Olhe o terminal do backend para ver se o comando `spmm6x` foi transmitido.
- **Checkbox “Ativar Joystick” desmarca sozinho**:
  - O `JoystickController` força `false` quando `setEnabled` detecta falta de gamepad.
  - Reconecte o controle ou teste outro navegador.

Autor

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

Github
Linkedin
Portfólio

Última atualização: Novembro 2025