

# FlightGear Simulation Bridge

---

## Visão Geral

Este guia descreve a ponte de simulação entre o FlightGear e a Plataforma de Stewart. O componente `interface/simulation/fg-bridge.py` lê roll/pitch do simulador via Telnet, limita amplitudes para manter a geometria segura e repassa poses ao backend FastAPI. O backend, por sua vez, calcula a cinemática inversa e envia os setpoints aos atuadores da plataforma real.

## Arquitetura e Fluxo de Dados

- **FlightGear (Telnet 5050):** expõe propriedades `/orientation/roll-deg` e `/orientation/pitch-deg`.
- **fg-bridge (Python + httpx + flightgear\_python):**
  - Conecta ao Telnet do FlightGear (`FG_TELNET_HOST`, `FG_TELNET_PORT`).
  - Lê roll/pitch, aplica `clamp_angle` com limite configurável (`FG_ANGLE_LIMIT`, padrão 15°).
  - Monta pose segura com `z` e `yaw` fixos (`SAFE_Z`, `SAFE_YAW`).
  - Chama `POST /calculate` para validar workspace e obter pré-visualização.
  - Publica a pré-visualização em `POST /flight-simulation/preview` para que o frontend possa renderizar mesmo sem telemetria física.
  - Verifica a flag `GET /flight-simulation/status` (`enabled`) para saber se deve ou não controlar a plataforma.
  - Envia a pose final para `POST /apply_pose`, que encaminha os comprimentos aos atuadores via serial.
- **Backend (FastAPI):** recebe poses, calcula cinemática inversa e repassa comandos ao ESP32-S3. Também alimenta o WebSocket `/ws/telemetry` usado pelo frontend para exibir a pré-visualização e o estado da simulação.

## Configuração

Variáveis de ambiente consumidas por `fg-bridge.py`:

- `STEWARD_API_BASE` (padrão `http://localhost:8001`) – base da API FastAPI.
- `STEWARD_APPLY_PATH` (padrão `/apply_pose`) – rota de aplicação de pose.
- `FG_TELNET_HOST` / `FG_TELNET_PORT` (padrão `localhost:5050`) – endereço do FlightGear.
- `FG_POLL_INTERVAL` (padrão 0.1 s) – período de leitura.
- `FG_RECONNECT_DELAY` (padrão 2.0 s) – tempo para tentar reconectar ao Telnet.
- `FG_ANGLE_LIMIT` (padrão 15.0 graus) – saturação de roll/pitch.
- `SAFE_Z` (padrão 540.0) e `SAFE_YAW` (padrão 0.0) – componentes fixos da pose enviada.

## Execução

1. No FlightGear, habilite o servidor Telnet: `--telnet=socket,in,30,localhost,5050,udp`.
2. Suba o backend (`uvicorn app:app --host 0.0.0.0 --port 8001`) e habilite a flag de simulação via endpoint ou UI.
3. Ajuste variáveis de ambiente se necessário e execute:

```
cd interface/simulation
python fg-bridge.py
```

4. Acompanhe os logs: conexões Telnet, ângulos lidos, preview aceito ou rejeitado, e aplicação da pose.

## Segurança Operacional

- A ponte só envia poses se **enabled** em `/flight-simulation/status` estiver ativo.
- Roll/pitch são saturados em `FG_ANGLE_LIMIT`; poses fora do workspace são descartadas ao cair na validação de `/calculate`.
- Reconexões ao Telnet são automáticas, minimizando perda de dados se o FlightGear reiniciar.

## Troubleshooting

- **Conexão recusada:** confirme porta `5050` e parâmetro `--telnet` no FlightGear.
- **Preview inválido:** reduza `FG_ANGLE_LIMIT` ou ajuste `SAFE_Z` para manter a plataforma dentro do curso mecânico.
- **Sem movimento na plataforma:** verifique se `/flight-simulation/status` retorna `{"enabled": true}` e se o backend está com a serial aberta.



### Autor

**Guilherme Miyata** - Instituto Federal de São Paulo (IFSP)

Trabalho de Conclusão de Curso - 2025

---

[Github](#)

[Linkedin](#)

[Portfólio](#)

**Última atualização:** Novembro 2025