s²innovation
Software solutions

AI-driven Tango device driver generator

Software development and control systems integration services

www.s2innovation.com

AI-driven Tango device driver generator
The goal of the project

Web-based application that **automates the generation** of Tango Controls device servers using Large Language Models (LLMs).

This application **surpasses the functionality of existing tools** (like POGO) by also implementing device-specific logic based on documentation, and not just generating templates.

AI-driven Tango device driver generator
Key objectives

1. **Automate** the generation of **device server** code.
2. **Simplify the process** through a GUI and eliminate dependency on desktop software.
3. Use documentation-based code generation via **RAG** (Retrieval-Augmented Generation).
4. Enable **multi-model support** (GPT, Claude, Gemini) for comparative code generation.

AI-driven Tango device driver generator
Technological Foundation

Tango Controls

1. A SCADA/DCS framework widely used in research and industry.
2. Device servers handle communication with physical devices using C++, Python, or Java.
3. Servers consist of Attributes, Commands, and Device Properties.

AI-driven Tango device driver generator
Technological Foundation

Large Language Models

1. Models such as GPT-4o, Gemini 1.5 Pro, and Claude 3 Opus are used.
2. The system leverages prompting strategies (e.g., role prompting, one-shot prompting).
3. JSON-formatted schema inputs are converted into rich prompts.

AI-driven Tango device driver generator
Technological Foundation

LangChain and RAG

1. **LangChain** is used to orchestrate interactions, manage prompts, and handle embeddings.
2. **RAG** enriches the prompt with device-specific documentation stored in a vector database (FAISS), improving contextual relevance and precision.

AI-driven Tango device driver generator
Prompt Engineering

The generated prompt for the LLM includes:

1. A **role-defining statement instructing** the model to generate Tango device servers.
2. **Example implementation** (one-shot learning).
3. Embedded **documentation snippets**.
4. **User-defined** attributes/commands/properties formatted as numbered lists.
5. **Chat history** for context-aware conversation continuity.

Special care is taken to ensure prompt formatting complies with LLM best practices for accuracy and conciseness.

AI-driven Tango device driver generator
GPT-4o (by OpenAI)

Why it's good:

1. Produces very **clear and short code**.
2. **Follows user instructions** very well.
3. **Rarely** makes mistakes.
4. Great when you need **precise results**.

AI-driven Tango device driver generator
Gemini 1.5 Pro (by Google)

Why it's good:

1. Adds things like **error checking** and **input validation** automatically.
2. Can **guess** what **might be needed**, even if not fully described.

Issues:

1. Sometimes makes mistakes (e.g., uses wrong functions or skips needed parts).
2. The code often needed fixing before it could run.
3. May “hallucinate” — add things not asked for.

AI-driven Tango device driver generator
Claude 3 Opus (by Anthropic)

Why it's good:

1. Handles **long inputs** well.
2. Often **accurate** and **clean** in code generation.
3. Needs **fewer prompts** to understand context.

AI-driven Tango device driver generator
Model Comparison - Simple Summary

Feature	GPT-4o	Gemini 1.5 Pro	Claude 3 Opus
Best for	Clean, simple code	Creativity & features	Complex inputs
Accuracy	✅✅✅ (high)	❌✅ (mixed)	✅✅ (good)
Code quality	Very high	Needs fixing	Good
Adds extra features	No (just what you ask)	Yes	Sometimes
Needs manual changes	Rarely	Often	Sometimes
Good with long context	Moderate	Moderate	Very good

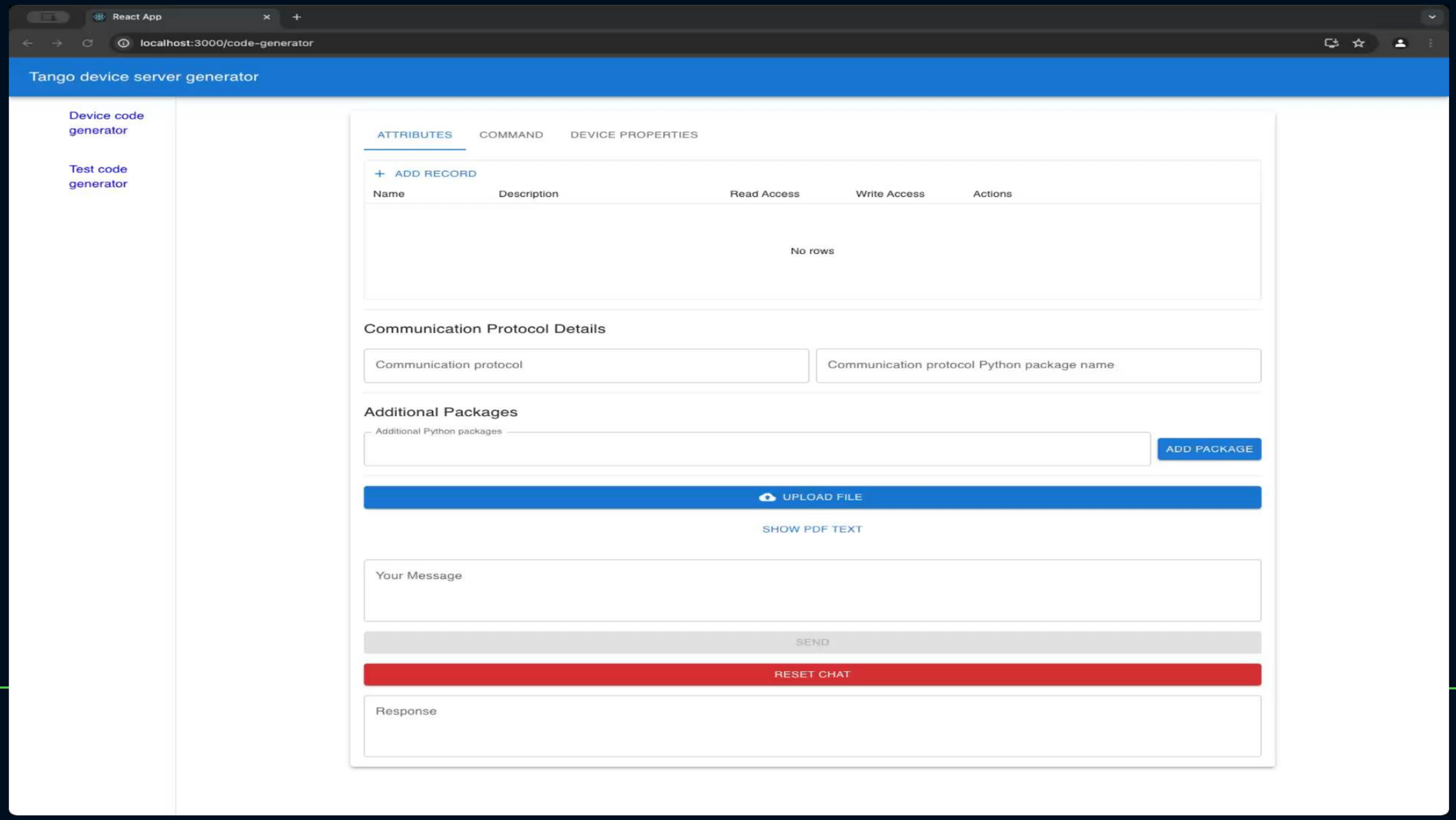
AI-driven Tango device driver generator
Real-world scenarios

Tests:

1. Danfysik 8000 power supply.
2. Metrolab PT2026 teslameter.
3. HPLC pump with PI 872 controller.
4. Eurotherm 3508 temperature controller.
5. Andor Newton CCD camera

About ~20% of time saving

AI-driven Tango device driver generator



AI-driven Tango device driver generator
Proposed Enhancements

Future development directions include:

1. Automated **test code generation**.
2. Integration with **open-source LLMs**.
3. **Enhanced search** and indexing techniques.
4. Support for additional programming languages beyond **Python**.

AI-driven Tango device driver generator

Real-world scenarios

We are looking for **partners** for further testing!

Please contact:

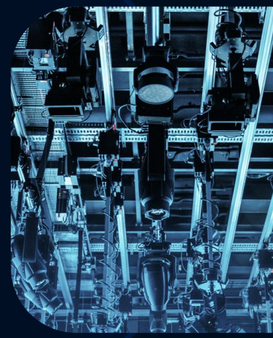
Lukasz Zytniak

Email: lukasz.zytniak@s2innovation.com

Mobile: (+48) 789 339 875

Thank You

For Your Attention



Lukasz Zytniak – COO of S2Innovation
Email: lukasz.zytniak@s2innovation.com
Mobile: (+48) 789 339 875