Rust for embedded devices

EchoKit with MCPs

EchoKit

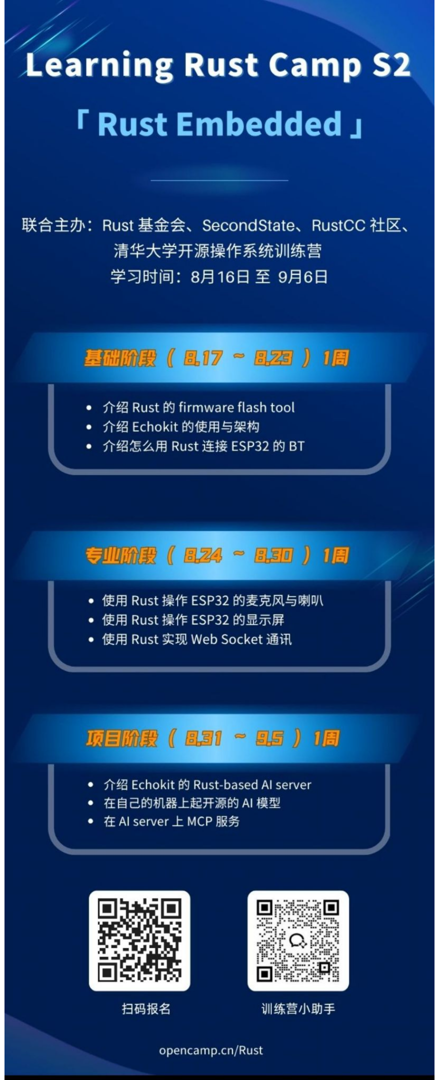
Star, clone and fork 👍

EchoKit devices: https://github.com/second-state/echokit_box

EchoKit server: https://github.com/second-state/echokit_server

Introduction:
https://opencamp.cn/Rust/camp/S02

Sign up here:
https://opencamp.cn/Rust/camp/S02/register?code=cHsXpIq2vGdaM



Learning Rust Camp S2

「Rust Embedded」

联合主办：Rust 基金会、SecondState、RustCC 社区、
清华大学开源操作系统训练营
学习时间：8月16日 至 9月6日

基础阶段（ 8.17 ~ 8.23 ）1周

- 介绍 Rust 的 firmware flash tool
- 介绍 Echokit 的使用与架构
- 介绍怎么用 Rust 连接 ESP32 的 BT

专业阶段（ 8.24 ~ 8.30 ）1周

- 使用 Rust 操作 ESP32 的麦克风与喇叭
- 使用 Rust 操作 ESP32 的显示屏
- 使用 Rust 实现 Web Socket 通讯

项目阶段（ 8.31 ~ 9.5 ）1周

- 介绍 Echokit 的 Rust-based AI server
- 在自己的机器上起开源的 AI 模型
- 在 AI server 上 MCP 服务

扫码报名　　训练营小助手

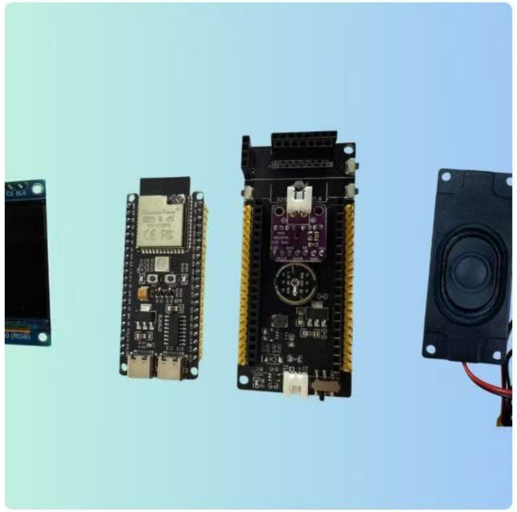
opencamp.cn/Rust

The EchoKit device

An ESP32-S3 SoC + audio processor + microphone + speaker + buttons + USB

https://opencamp.ai/Rust/bbs/2

Echokit
08/04 16:37:59



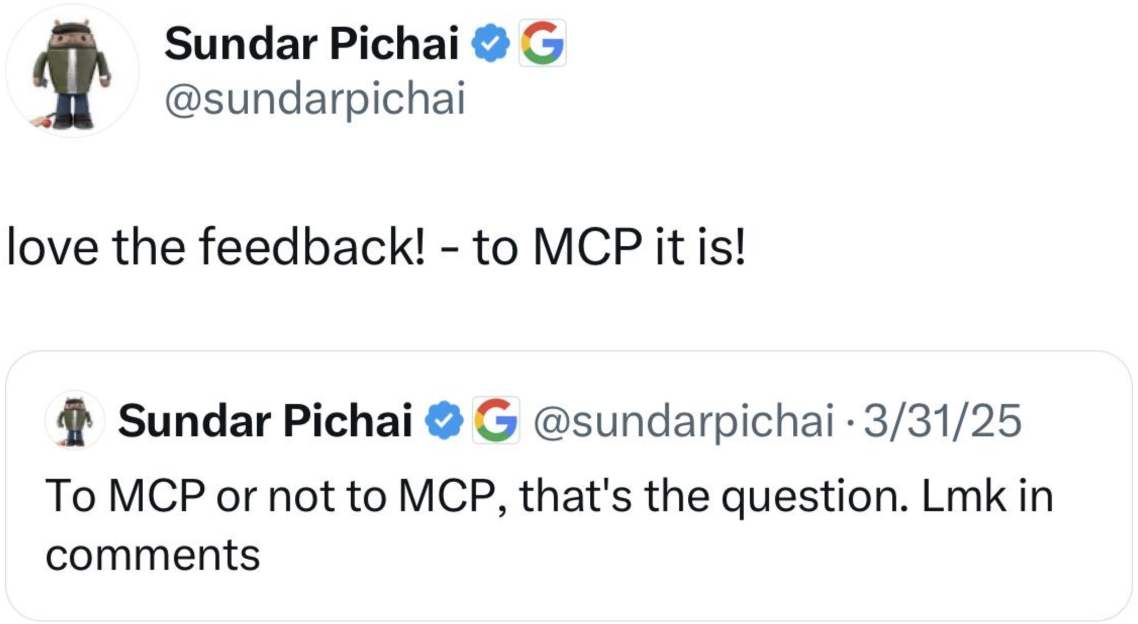
嵌入式Rust训练营专用设备 EchoKit

🎓【训练营简介】嵌入式 Rust 训练营是一门面向初学者的项目制学习课程，涵盖嵌入式...

¥168

长按识别小程序 跟团购买

MCP is the hands and feet of LLMs



Sundar Pichai
@sundarpichai

love the feedback! - to MCP it is!

Sundar Pichai @sundarpichai · 3/31/25

To MCP or not to MCP, that's the question. Lmk in comments

Who are the real users of computers?

Humans

The web and mobile app era

Computers

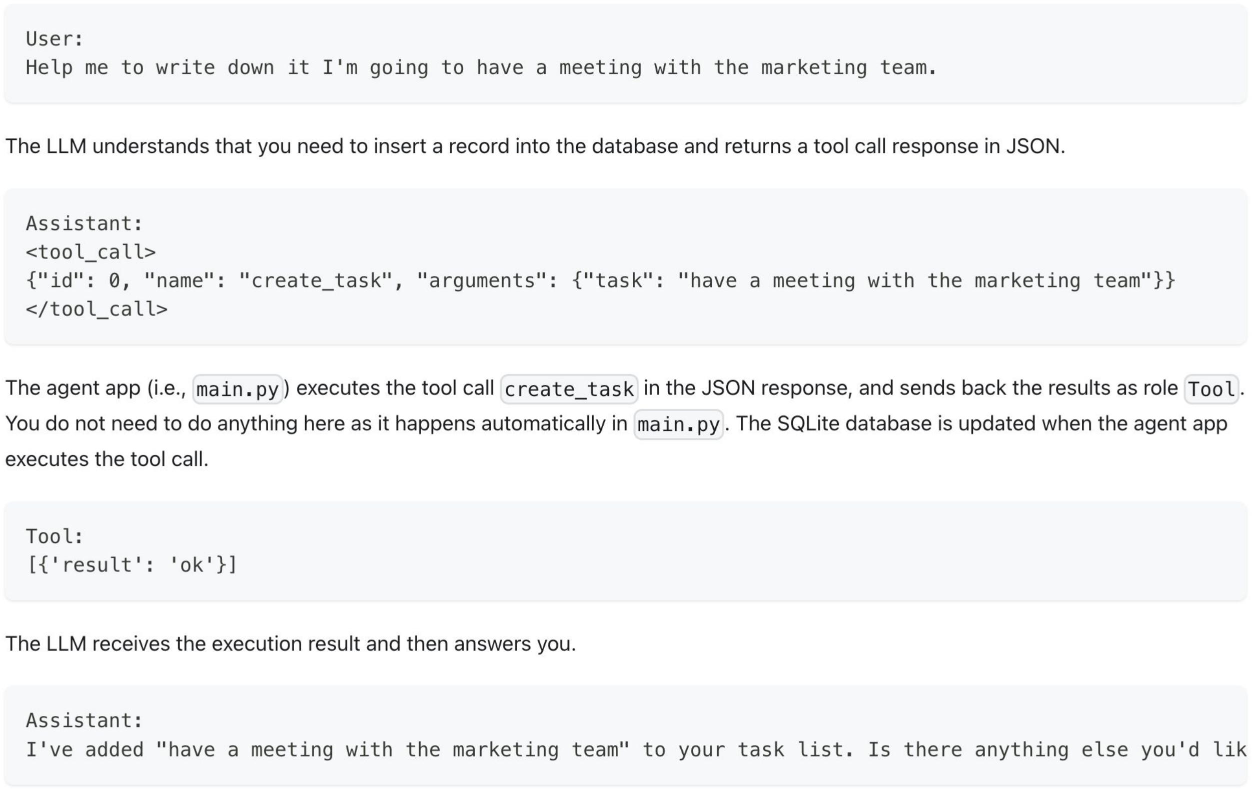
The “API first” era

AIs

The MCP era

https://llamaedge.com/docs/user-guide/llm/tool-call

LLM tool calls



```
User:
Help me to write down it I'm going to have a meeting with the marketing team.
```

The LLM understands that you need to insert a record into the database and returns a tool call response in JSON.

```
Assistant:
<tool_call>
{"id": 0, "name": "create_task", "arguments": {"task": "have a meeting with the marketing team"}}
</tool_call>
```

The agent app (i.e., `main.py`) executes the tool call `create_task` in the JSON response, and sends back the results as role `Tool`. You do not need to do anything here as it happens automatically in `main.py`. The SQLite database is updated when the agent app executes the tool call.

```
Tool:
[{'result': 'ok'}]
```

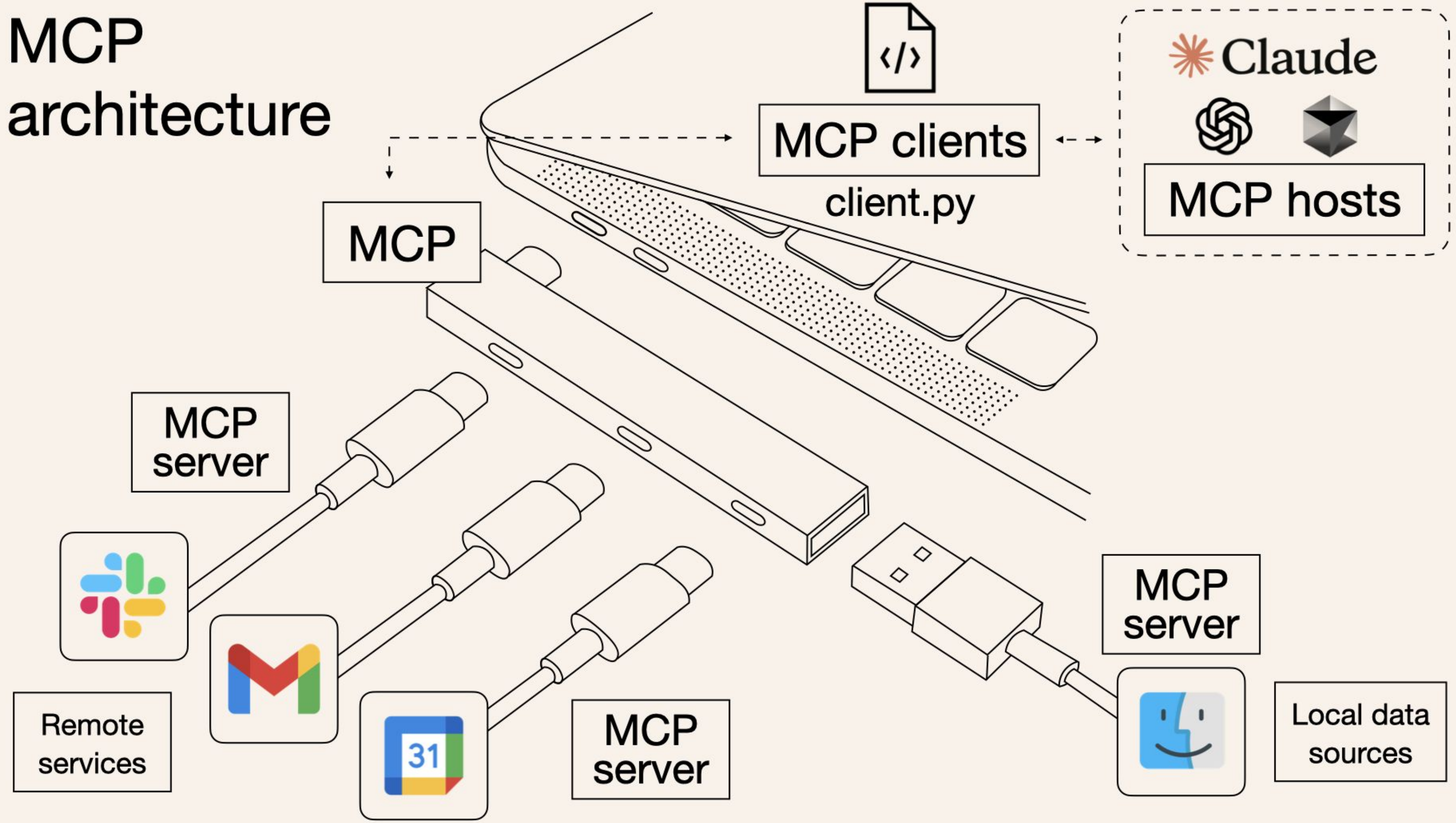
The LLM receives the execution result and then answers you.

```
Assistant:
I've added "have a meeting with the marketing team" to your task list. Is there anything else you'd lik
```

Use cases

- Supplement knowledge on-demand
  - Search the Internet
  - Search private knowledge bases
- Internet actions
  - Send email or messages
  - Pay money
  - Update calendar
  - Make phone calls
  - Any Zapier or n8n workflows
- Local actions
  - Turn on the light
  - Close the curtain
  - etc.

MCP architecture

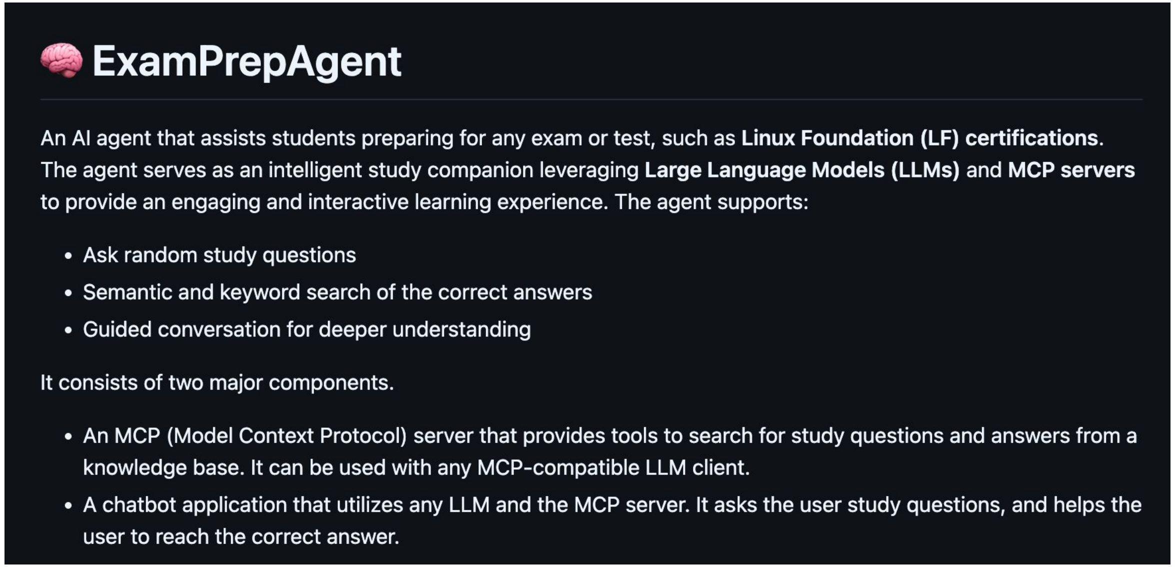


Architecture

- MCP servers
  - https://hub.docker.com/mcp
  - https://mcp.so/
  - https://mcpmarket.com/
  - Create your own … https://github.com/modelcontextprotocol
- EchoKit Server is an MCP client
  - https://github.com/modelcontextprotocol/rust-sdk

Demo: Civics test

The MCP server



🧠 ExamPrepAgent

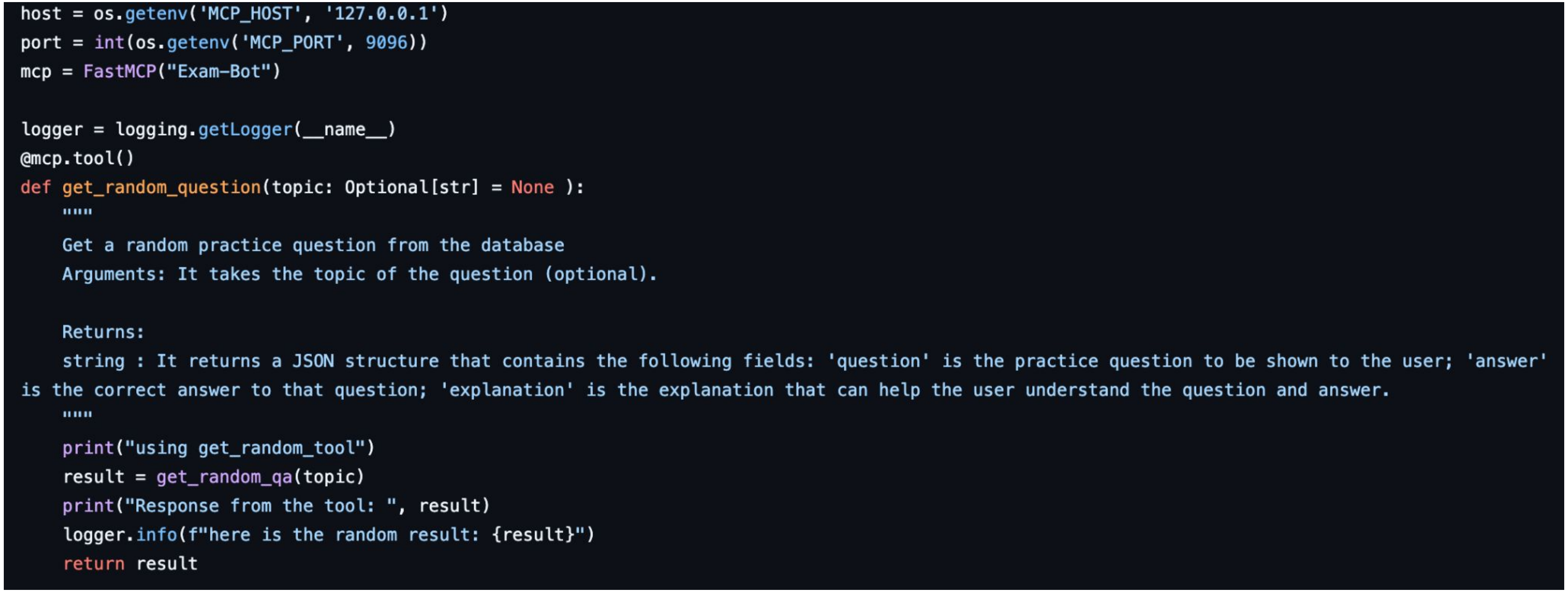
An AI agent that assists students preparing for any exam or test, such as **Linux Foundation (LF) certifications**. The agent serves as an intelligent study companion leveraging **Large Language Models (LLMs)** and **MCP servers** to provide an engaging and interactive learning experience. The agent supports:

- Ask random study questions
- Semantic and keyword search of the correct answers
- Guided conversation for deeper understanding

It consists of two major components.

- An MCP (Model Context Protocol) server that provides tools to search for study questions and answers from a knowledge base. It can be used with any MCP-compatible LLM client.
- A chatbot application that utilizes any LLM and the MCP server. It asks the user study questions, and helps the user to reach the correct answer.

https://github.com/cardea-mcp/ExamPrepAgent



```python
host = os.getenv('MCP_HOST', '127.0.0.1')
port = int(os.getenv('MCP_PORT', 9096))
mcp = FastMCP("Exam-Bot")

logger = logging.getLogger(__name__)
@mcp.tool()
def get_random_question(topic: Optional[str] = None ):
    """
    Get a random practice question from the database
    Arguments: It takes the topic of the question (optional).

    Returns:
    string : It returns a JSON structure that contains the following fields: 'question' is the practice question to be shown to the user; 'answer'
is the correct answer to that question; 'explanation' is the explanation that can help the user understand the question and answer.
    """
    print("using get_random_tool")
    result = get_random_qa(topic)
    print("Response from the tool: ", result)
    logger.info(f"here is the random result: {result}")
    return result
```

main.py

Configure EchoKit Server

To start the MCP server

```
nohup python main.py &
```

In the `config.toml` file

```
[[llm.mcp_server]]
server = "http://localhost:9096/mcp"
type = "http_streamable"
```

Prompt to use the tool

You are a helpful test prep expert that asks the user mock exam questions and then helps the user understand the correct answer.

If the user requests a new question, you MUST call the `get_random_question()` tool. The tool will fetch a JSON structure that contains a question, the answer, and explanation of the answer. You MUST only return the question in the JSON to the user. Respond with the question text only. Do NOT add text before or after the question. Do NOT re-write the question in any way!

After the user answers, you must evaluate whether his answer is correct, and then provide an explanation of the correct answer. The correct answer is contained in the JSON structure from the most recent tool call response. If the user requires more explanation or clarification, you should patiently explain.

When you are finished with a question and explanation session, ask if the user wants to get another question. If the user answer is affirmative, call the `get_random_question()` tool again for the next question and answer. You could pass a keyword to the get_random_question() tool call to get a new question that is similar to a past one if the user needs improvement on that subject.

Send LLM tool call results directly back to EchoKit device and let the device handle it

Next step

Assignments

Fork the repo: https://github.com/second-state/echokit_server

Make code changes in your own repo:

- Capture error or empty value in the LLM response. 抓住错误或者空的大模型回答。
- Send a “standard” response, such as “Sorry, I cannot understand. Can you say it in another way?”, to TTS and then to EchoKit device. 把一个“标准”的回复，送去 TTS 然后发还给 EchoKit 设备：比如“抱歉，我没能理解您的回复。请您换种表达方式重新说一下”。

Task 1/3: Improve LLM support

Fork the repo: https://github.com/second-state/echokit_box

Make code changes in your own repo:

- Detect a wake words at idle state using the ESP32 API. 在 idle 状态下，用 ESP32 的 API 检测自定义的唤醒词。
- Switch to listening mode when this wake word is detected. 检测到唤醒词之后，切换到 listening 状态。
- Detect a second wake word at listening state using the ESP32 API. 在 listening 状态下，用 ESP32 的 API 检测自定义的唤醒词。
- Switch to idle mode when this wake word is detected. 检测到唤醒词之后，切换到 idle 状态。

Task 2/3: Wake words

Create a demo video and upload to a public video sharing service. 录制一个演示视频，并且上传到 B 站等视频分享网站之一。

The video must include the following elements.

- Use Bluetooth to configure WiFi, EchoKit server, and a background image. 用蓝牙配置设备的 Wi-Fi，EchoKit 服务器，以及屏幕背景图。
- Demonstrate the wake words. 演示用唤醒词切换状态。
- Converse with the device through multiple turns. 与设备进行多轮对话。

Task 3/3: Create video demo

Have fun coding!

EchoKit

Star, clone and fork 👍

EchoKit server: https://github.com/second-state/echokit_server

VAD server: https://github.com/second-state/silero_vad_server

TTS server: https://github.com/second-state/gsv_tts