Structured Outputs 的 `strict:true` 模式让 GPT-4o 系列模型 *必定* 输出符合你提供的 JSON-Schema；因此你可以把返回值锁定为诸如 `"LED" / "WEATHER" / "MOTOR" / "TEMP"` 这些枚举关键词，再在后台按关键词路由到相应硬件或服务。下面给出一份完整、可直接运行的示例，并解释每一步需要注意的细节。

示例项目目录

```
assistant_demo/
├── main.py          # 调用 OpenAI API 并解析结果
└── schema.json      # 枚举约束（自动加载）
```

`schema.json` —— 定义允许的枚举值

```json
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "cmd": {
      "type": "string",
      "description": "Which subsystem to activate",
      "enum": ["LED", "WEATHER", "MOTOR", "TEMP"]
    }
  },
  "required": ["cmd"],
  "additionalProperties": false
}
```

> 只要把 `"enum"` 数组改成你需要支持的关键词列表即可。
> `additionalProperties:false` 防止模型偷偷添加别的字段。 (Structured Outputs - OpenAI API, Structured Outputs - OpenAI API)

`main.py` —— Python 调用与后端路由

PYTHON

```python
from openai import OpenAI
import json, pathlib, sys, subprocess

MODEL = "gpt-4o-mini-2024-07-18"   # 支持 strict:true 的模型版本
client = OpenAI()                  # 需设置 OPENAI_API_KEY 环境变量

# 1. 读取 JSON-Schema
schema = json.loads(pathlib.Path("schema.json").read_text())

# 2. 状态机：1) 提问 → 2) 返回关键词 → 3) 后端执行
while True:
    user_msg = input("User> ").strip()
    if not user_msg:
        break

    chat = client.chat.completions.create(
        model=MODEL,
        messages=[{"role": "system",
                   "content": "你是硬件路由助手，只能返回 JSON，且字段
cmd 必须是 LED/WEATHER/MOTOR/TEMP 中之一"},
                  {"role": "user", "content": user_msg}],
        response_format={
            "type": "json_object",
            "schema": schema,
            "strict": True           # ★ 关键：强制符合 schema
        }
    )

    # 3. 解析输出
    try:
        payload = json.loads(chat.choices[0].message.content)
        cmd = payload["cmd"]
    except (json.JSONDecodeError, KeyError):
        print("⚠️  模型输出不符 schema：",
chat.choices[0].message.content)
        continue
```

```
    # 4. 路由表—在此处调用你的硬件 / 服务
    match cmd:
        case "LED":
            print("👉 切换 LED")      # 例如 gpiozero.LED(17).toggle()
        case "WEATHER":
            print("👉 查询天气")      # 调用气象 API 并朗读
        case "MOTOR":
            print("👉 驱动电机")      # pigpio set_PWM_dutycycle()
        case "TEMP":
            print("👉 读取温度传感器") # 读取 DHT11
```

> 当你在终端输入 **打开主灯**，模型会返回：
>
> ```
> {"cmd":"LED"}
> ```
>
> 程序随即进入 `LED` 分支执行实际动作。

代码分步解析

1. 选择支持 Strict 模式的模型

只有 GPT-4o (及 mini) 的 2024-08-06 / 2024-07-18 及之后版本保证与 `response_format` 搭配 `strict:true` 工作。 (Introducing Structured Outputs in the API - OpenAI)

2. 在 `response_format` 里启用 `strict:true`

- **JSON-mode** 与 **Structured Outputs** 使用同一字段 `response_format`；当你同时提供 `schema`并设置 `strict:true`，OpenAI 网关会先验证你提交的 Schema，再强制模型输出严格匹配的 JSON。 (Structured Outputs - OpenAI API, Introducing Structured Outputs in the API - OpenAI)
- 若模型生成了无效 JSON，会自动重试最多 3 次，否则返回 `invalid_response_format` 错误；你可以捕获后降级为普通文字回答或提示用户重试。 (Introduction to Structured Outputs | OpenAI Cookbook, Client side JSON Schema in response_format validation for ... - GitHub)

3. 把枚举放在 Schema，而非 prompt

- 模型会把 `enum` 视作唯一合法 token 集；这比在系统提示里写“只能回答 LED/WEATHER...”更稳定，且升级模型后仍有效。 (Structured outputs `response_format`

requires `strict` function calling …, DOC: with_structured_output ⇒ example of setting enum #27139)

- 若想一次请求多值（例如返回 `cmd` + `argument`），可把 `argument` 定义为字符串或数值字段并移除 `additionalProperties:false`。

4. 后端安全校验

尽管有 `strict:true`，仍推荐在业务层按白名单二次校验，以防未来 Schema 变更或你误改模型版本。（Strict=True and Required Fields! - OpenAI Developer Forum, Structured outputs `response_format` requires `strict` function calling …）

5. 与 LangChain / LlamaIndex 等框架结合

这些框架已提供 helper（如 `with_structured_output`）自动生成 `response_format`，并可把枚举封成 Python Enum；用法与裸 OpenAI SDK 相同。（DOC: with_structured_output ⇒ example of setting enum #27139, Automate your bookkeeping with LangChain & GPT-4 | Medium）

常见错误与调试

现象	原因	解决
`openai.BadRequestError: invalid_response_format`	Schema 语法错误或模型不支持	用在线 JSON-Schema Linter 先校验；或升级到支持 strict 的 4o 模型。（Client side JSON Schema in response_format validation for … - GitHub）
模型输出 `{"cmd":"LED","foo":1}`	忘记 `additionalProperties:false`	加上即可阻止多余字段。（Structured Outputs - OpenAI API）
输出 `"led"` 小写	未列入 enum（区分大小写）	要么改小写，要么把两种大小写都加进 enum。

延伸阅读

- 官方文档 — Structured Outputs 指南、`response_format` 字段说明。（Structured Outputs - OpenAI API, Structured Outputs - OpenAI API）
- **OpenAI Cookbook** — 《Introduction to Structured Outputs》《Leveraging structured outputs for classification》。（Introduction to Structured Outputs | OpenAI Cookbook,

Leveraging model distillation to fine-tune a model | OpenAI Cookbook)

- 社区讨论 — 关于 strict:true 行为、schema 校验与错误处理的实战经验。 (Strict=True and Required Fields! - OpenAI Developer Forum, Using response_format in chat completion throws error - API)
- 公告博客 — OpenAI 发布 Structured Outputs 与示例请求/响应。 (Introducing Structured Outputs in the API - OpenAI)

小结

使用 `response_format → schema + strict:true`，你可以让 GPT-4o 只返回受约束的 **JSON**；在 Schema 的 `enum` 中列出关键词，例如 `"LED" / "WEATHER" / "MOTOR" / "TEMP"`，就能把模型回复当作 安全、可靠的指令路由键，后台无需复杂解析即可调用相应硬件或 API。