

VIBE CODING 方法论

—— SKILL 与 MCP SERVER 实战

LESSON 02 · 本节课

把开发板 API 写成 MCP Server

MCP 给 agent 装上 =

手 + 眼

WRITE PINS — READ BACK. 没有读回与日志，闭环就是瞎的。

“

tool 的 docstring 和 schema，
就是给模型看的说明书。

学期 2026 秋季

时长 45 分钟 · 含实验

前置 第 1 节课的 skill

课程地图

COURSE MAP

三节递进：方法论 →

能力 → 融合

01

写 Skill

方法论 · 闭环 SOP

02

写 MCP Server

能力 · agent 的手脚

本节

03

Skill × MCP 协同

融合 · 故障下不崩

MCP = agent 的手脚

一个跑在旁边的进程
把外部能力暴露成 tools

三个认识

WHAT MCP IS

1 一份独立进程，暴露 tools / resources

tool = agent 可调用的函数；resource = 按需读取的资料（如引脚表）

2 传输走 stdio

agent CLI 以子进程拉起 server，JSON-RPC 过 stdin / stdout——不开端口、不碰网络

3 说明书写给模型看

tool 的 docstring 与 schema 就是模型读的使用手册：参数含义、范围、单位

一条链路，两段协议

AGENT ⇌ SERVER ⇌ BOARD

agent CLI Claude Code · OpenCode · Cursor

决定什么时候调哪个 tool

↑ ↓ stdio · JSON-RPC
子进程拉起 · 本课唯一传输

MCP server devboard · mcp_server/server.py

把板子 API 翻译成 tools；错误返回可读文本

gpio_write

gpio_read

pwm_set TODO

adc_read TODO

board_reset

get_logs

firmware_flash

7 个 tools · 红虚线 2 个 = 实验二 L1 等你补的桩

↑ ↓ HTTP 127.0.0.1:8765
或真板串口 · 接口一致

板子 / 模拟器 GPIO · PWM · ADC

LED = GPIO2 · 光敏 = 引脚 34

最小 Server 解剖

主稿：server.py · 16 行节选

边注：四个强调点，按行号对位

mcp_server/server.py

PYTHON · FASTMCP · STDIO · EXCERPT

```
01 mcp = FastMCP("devboard")
02 board = BoardClient()      # 模拟器 HTTP / 真板串口
03
04 @mcp.tool()
05 def gpio_write(pin: int, level: int) -> str:
06     """设置 GPIO 引脚电平。pin: 0-33; level: 0/1。"""
07     if not 0 <= pin <= 33: return "ERROR: pin 非法"
08     try: board.gpio_write(pin, level)
09     except BoardError as e: return f"ERROR: {e}" # 不抛裸异常
10     return f"OK: GPIO{pin} 已置为电平 {level}"
11
12 @mcp.tool(annotations={"destructiveHint": True})
13 def firmware_flash(path: str, confirm: bool = False) -> str:
14     if not confirm: return "ERROR: 烧录需用户明确确认..."
15
16 # TODO(student): pwm_set / adc_read — 实验二 L1 等你补
```

— 1
— 2
— 3
— 4

四个强调点

FOUR EMPHASES · 边注对位

- 1

docstring 即说明书

L06

参数含义 / 范围 / 单位写给模型看——它只读得到这个。
- 2

schema 校验挡非法参数

L07

越界 pin 返回 ERROR: 文本，模型能自纠而不是烧一轮。
- 3

错误返回可读文本

L09

串口断开、超时——绝不向模型抛裸 traceback。
- 4

危险操作加确认

L12-14

安全边界：只暴露可逆操作；flash 类 confirm=false 一律拒绝。

课程地图：轮到「能力」

方法论 → 能力 → 融合
每节课的作业都是下节课的输入

递进结构 · 正 → 反 → 合

课次	主题	一句话定位	实验
01	写 Skill ✓ 已完成	把「写→查→验」固化成 agent 的 SOP	修复坏 skill（纯软件闭环）
02	写 MCP Server 本节	把开发板 API 变成 agent 的手脚	补全 tools + 点灯
03	Skill × MCP 协同	纪律 + 手 = 可靠闭环，故障下不崩	协同改造 + 故障注入

HANDOFF 上节课留下纪律，这节课补上能力	1 纪律已就位 → 遗产：本组 SKILL.md 草稿——不验证不算完、失败有限重试、超限升级给人	2 能力还缺：闭环的「验」停在软件层 → 测试全绿，真板不亮它不知道——本节课补 MCP tools = 手 + 眼
----------------------------	--	---

今日议程

页码	环节	内容
P02-03	开场 · 先看 MCP	定义（是什么） · 最小 server 长什么样
P06	回顾衔接	上节课的闭环，断在看不见硬件
P07-08	MCP 概念	server 五层结构 · 与 skill 的分工
P09	讲「写」	API→tool 映射表（解剖已在 P03 见过）
P10-15	讲「用」 · 安装与使用	三步总览 · 环境安装 · 注册 · 一次调用旅程 · 翻车排错
P16-18	实验二讲解	智能台灯 · 三层任务 · rubric
P17 驻屏	实验二 动手	补全 pwm_set / adc_read + 点灯读回
P19	展示 + 小结 + 作业	skill 改固件版草稿 → 第 3 节课协同强化

闭环断在哪

上节课的 skill 跑真板任务
测试全绿，真板不亮

回顾脚本

RECAP · SCRIPT

拿上节课修好的 skill，换一个真板任务再跑一遍。

1 skill 已会跑闭环
写 → 查 → 验，三步都在**软件层**

2 换真板任务：感光调光
对象 firmware/light_led.py

3 结果：全绿，但灯不亮
curl /state →
led_brightness = 0

上节课的闭环，停在软件层

WRITE → CHECK → VERIFY



—— 这节课要装的，就是让 agent 「摸到」板子的那双手 —— MCP Server

一个 server 由什么构成

一个进程 · 五层

本课 scaffold 只注册 tools

agent CLI 以子进程拉起 server · 会话结束即回收

| stdio · JSON-RPC 请求 ↓ 文本 ↑
进程 = mcp_server/server.py · 一个 Python 文件

1 传输入口

main(): mcp.run(transport="stdio") —— 不开端口、不碰网络, stdin/stdout 就是总线

2 注册名

mcp = FastMCP("devboard") —— 注册进 agent 时用的就是这个名字

3 tool 层 · 7 个

每个 @mcp.tool() 函数 = 一个 tool: 签名 → schema、docstring → 说明书、 annotations → 元数据 (如 destructiveHint)

4 返回值契约

成功 "OK: ..." / 失败 "ERROR: ..." 可读文本 —— 模型只读得到文本, 读不到异常栈

5 底层 client

BoardClient(): HTTP 127.0.0.1:8765 或真板串口 —— 接口一致, 上层零改动

↓ HTTP / 串口 板子细节止步于此

模拟器 / 真板 GPIO · PWM · ADC · 日志环

—— MCP server = 把「板子操作」包成模型能按名字调用、读说明书、拿文本回执的进程

三个边界说明

SCOPE NOTES

tools ≠ resources

tools = 模型可调用的函数; resources = 按需读取的资料 (如引脚表)。本课 server 只注册 tools; resources 的协同用法在第 3 节课。

进程生命周期跟着 agent

agent 启动时拉起、会话结束回收; 同一条 command 可注册进多家 harness, 各拉各的进程。

为什么是独立进程

语言与权限和 agent 隔离: 串口/HTTP 细节、板子状态都不进模型上下文——模型只看见 7 个名字和说明书。

分工：一个管流程，一个管能力

skill 告诉你该验什么
MCP 让你验得到

DIVISION OF LABOR

维度	Skill · 纪律	MCP Server · 能力
GOVERNS 管什么	流程与纪律：先查后验、有限重试、超限报告	能力与感知：写引脚、读回、看日志、读 ADC
FORM 形态	Markdown 文件：SKILL.md + references/	一个进程：FastMCP + 若干 @mcp.tool()
LOADING 何时生效	description 触发、用时才加载（渐进式披露）	注册后常驻可调用，list_tools 可见
IN THIS COURSE 本课例子	验证段写「调 gpio_write 后读回比对」	gpio_write 真的去把 GPIO2 置高
IF MISSING 缺一个会怎样	✗ 有手无纪律： 没读回就宣布成功 、失败无限重试	✗ 有纪律无手： 验不到硬件 ，测试全绿真板不亮

Skill ✗ MCP = 可靠闭环

第 3 节课的主题 · 本节课先把「能力」这半边装好

API → Tool 映射表

本课核心方法论
一张表抄完作业

BOARD API → MCP TOOLS

板子 API	MCP TOOL	设计要点
GPIO 读写 digital 0/1	gpio_read / gpio_write	参数 枚举校验 ：合法引脚号 0-33
PWM analog out	pwm_set(pin, duty)	数值 范围约束 写进 schema: duty 0-100
ADC analog in	adc_read(pin)	返回 原始值 + 换算电压 ：raw 0-4095 / 0-3.3V
烧录固件 flash	firmware_flash 危险	需用户确认 ；不可逆要标注 (destructiveHint)
复位 / 日志 reset · logs	board_reset / get_logs	调试闭环的关键——agent 靠它「看见」板子

映射三原则

MAPPING RULES

- 1

一个 tool 一件事
动词 + 对象命名：gpio_write, 不做大而全的 do_stuff
- 2

schema 把非法参数挡在门外
越界返回 ERROR: 文本, 让模型**能自纠**
- 3

感知类工具优先
read / logs / adc 是闭环的前提——先有眼, 再有手

危险操作：只暴露可逆操作；flash 类加确认

confirm=false → 一律拒绝执行

讲「用」总览：三步

写只是一半 · 装上才算
安装 / 机制 / 排错细节见 P11-15

01

注册 REGISTER

agent CLI 以**子进程**拉起 server；三家 harness 同一条 command。

```
$ claude mcp add devboard -- \
    .venv/bin/python
    mcp_server/server.py
```

OpenCode 写进 opencode.json · Cursor 在 MCP 设置里加——**command 完全相同**。

02

验证 LIST TOOLS

让 agent 自报工具清单，**7 个**一个都不能少。

> 你现在有哪些 devboard 工具？

应列出 gpio_write / gpio_read / pwm_set / adc_read / board_reset / get_logs / firmware_flash。**列不全** → 回 01 查注册。

03

排错 DEBUG · 三板斧

调用出错时，按顺序问三个问题：

- 1

server 单独跑通吗
python mcp_server/server.py 看 stderr
- 2

agent 里 list 到工具吗
list 不到 = 注册路径 / venv 写错
- 3

调用报错看 stderr
MCP 日志里有**原始异常**第一手证据

判据 agent 调一次 gpio_write(2, 1) 后，curl /state 里 **led_brightness > 0** —— 手装上了

安装 ① 环境：一次 setup

课前一天装完
课上只验不装

SETUP · 三步

- 1

make setup

建 .venv，装 mcp / pytest / ruff 三个依赖——只装进 venv，不污染系统 Python。
- 2

make doctor 自检

逐项打勾：Python 版本 / mcp / pytest / 模拟器连通性。全 ✓ 才算环境就绪。
- 3

（选做）切真板

有真板的组：把 simulator/client.py 底层换成串口实现——接口一致，上层零改动。

```
$ make doctor
✓ Python: Python 3.13.11
✓ mcp 已安装
✓ pytest 已安装
⚠ 模拟器未启动 (做 L0 前运行 make sim; make check 不需要)
doctor 完成
```

真实输出 · 2026-10-13 自验记录 — ⚠ 不是错：make check 自带进程内模拟器

两个真实的坑

KNOWN PITFALLS

mcp 版本必须钉 >=1.2, <2

mcp 2.x 把 FastMCP 改名 MCPServer 并移除旧导入路径——默认装 2.x 会直接 import 失败。requirements.txt 已钉死（实测装 1.30.0）。

教室网络受限的兜底

- ① 教师机提前 pip download -r requirements.txt -d wheels/，学生 --no-index --find-links wheels/ 离线装；
- ② 再不行索取 mcp_stub：只保 tests 可跑，L0 的 agent 点灯换有网环境或看录屏。

环境没就绪就别进 L0——doctor 的 ⚠ 与 ✓ 都是机器可判信号

make doctor exit code = 0 且无 x

安装 ② 注册进 agent

三家 harness · 同一条 command
路径一律写绝对路径

Claude Code CLI

一条命令写进项目配置：

```
$ claude mcp add devboard -- \
  <实验目录>/venv/bin/python \
  <实验目录>/mcp_server/server.py
```

devboard 是 server 名字；-- 后面整段是**启动命令**。

OpenCode OPENCODE.JSON

在项目根 opencode.json 加一段：

```
{
  "mcp": {
    "devboard": {
      "type": "local",
      "command":
[".../venv/bin/python",

".../mcp_server/server.py"],
      "enabled": true
    }
  }
}
```

同时兼容读 .claude/ 与 .agents/ 的项目配置。

Cursor SETTINGS

设置 → MCP → 新增 server，填**同一条 command**：

```
<实验目录>/venv/bin/python
<实验目录>/mcp_server/server.py
```

GUI 不同、**语义相同**：告诉 harness 「用这个命令把 server 拉起来」。

注册到底做了什么

MECHANICS

- 1 把**启动命令**写进配置（不是拷贝代码）
- 2 agent 启动时以**子进程**拉起 server（stdio 接通）
- 3 自动 list_tools 拿走 7 个工具的名字 + **docstring**——模型从此「看见」它们

安装 ② · 附：各家 harness 注册一览

11 家 · 同一条 command
CLI / JSON / YAML / GUI 四种壳

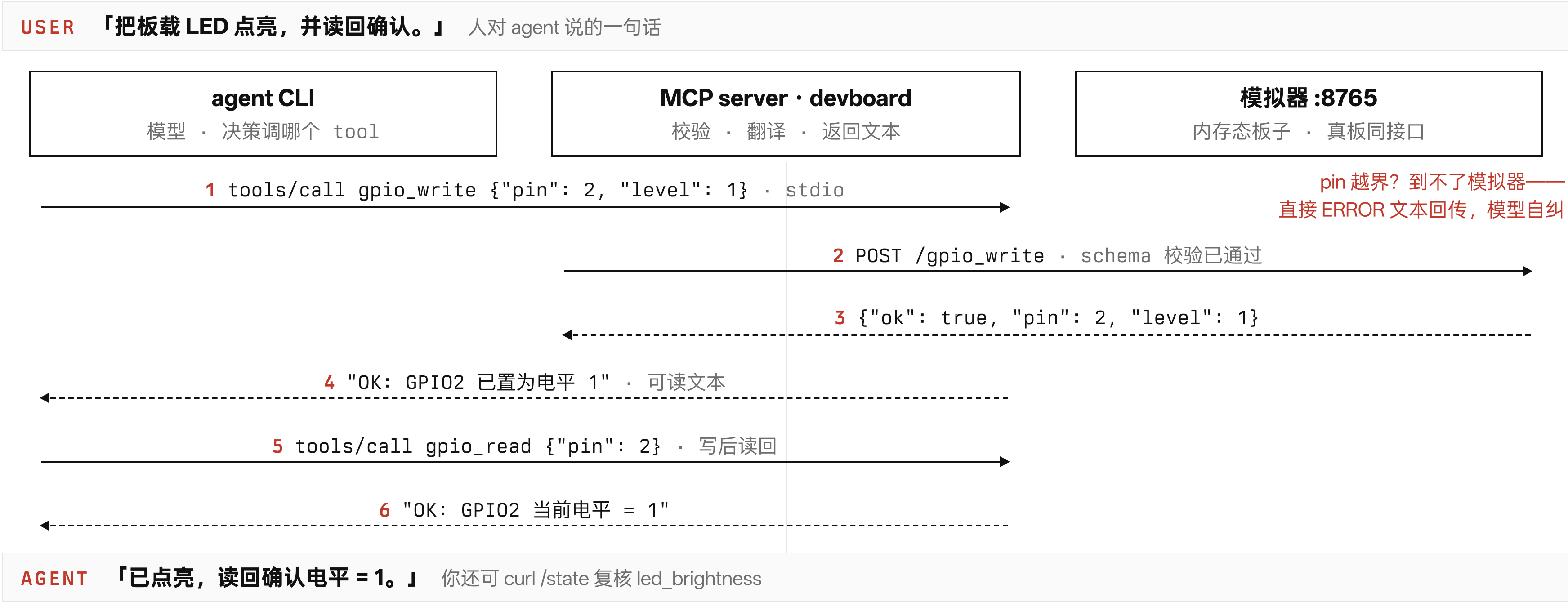
HARNESS	写在哪	写什么 (... = 实验目录，一律绝对路径)
Claude Code CLI	终端一行，写进项目配置	\$ claude mcp add devboard -- ../.venv/bin/python ../mcp_server/server.py
Codex CLI · TOML	终端一行；或 ~/.codex/config.toml	\$ codex mcp add devboard -- ../.venv/bin/python ../mcp_server/server.py [mcp_servers.devboard] command = "...python" args = ["...server.py"]
OpenCode JSON	项目根 opencode.json	"mcp" → "devboard": {"type": "local", "command": ["...python", "...server.py"]}
Qwen Code JSON	~/.qwen/settings.json	"mcpServers" → "devboard": {"command": "...python", "args": ["...server.py"]}
Pi JSON	~/.config/mcp/mcp.json (先 pi install npm:pi-mcp-adapter)	"mcpServers" 同 Qwen 一族；也可在 Pi 内 /mcp setup 交互添加
WorkBuddy JSON	~/.workbuddy/mcp.json	"mcpServers" 同上；保存后到「连接器管理」点信任并新开会话
Trae JSON · GUI	设置 → MCP → 手动添加；或 .trae/mcp.json	"mcpServers" 同上；支持 \${workspaceFolder} 变量
Cursor GUI · JSON	Settings → MCP 新增；或 .cursor/mcp.json	GUI 填同一条 command；JSON 也是 "mcpServers" 一族
VS Code 键名特例	命令面板 MCP: Add Server；或 .vscode/mcp.json	顶层键 "servers" → "devboard": {"type": "stdio", "command": "...python", "args": ["...server.py"]}
DeepSeek Harness YAML	~/.dsh/settings.yaml	mcpServers: → devboard: { command: ..python, args: ["...server.py"] }
豆包 GUI	「插件·技能·伙伴」→ 新建自定义插件	传输类型选 STDIO: 运行命令 = ...python, 参数 = ...server.py

殊途同归

11 家写的都是同一条启动命令: ../.venv/bin/python ../mcp_server/server.py——CLI / JSON / YAML / GUI 只是壳不同。

一次调用的完整旅程

「怎么回事」一图看完
实线请求 · 虚线返回



—— 模型全程只读文本：docstring 决定它会不会用对，返回值决定它知不知道成功

翻车现场排错

三种真实翻车 · 课堂现场演
症状 → 原因 → 处置

LIVE DEBUG · 三案

翻车	症状（真实文本）	原因 → 处置
A 模拟器没起	TOOL 返回 ERROR: 板子连接失败 (connection refused) · 建议动作：先运行 make sim	server 活着，但 8765 端口没人。 FIX make sim 后重试；make check 不需要 sim（自带进程内模拟器）。
B 注册路径写错	AGENT 答复 「我目前没有任何 devboard 工具。」 · list_tools 为空	command 用了相对路径或系统 python——agent 从别的目录拉起时找不到 venv。 FIX 改绝对路径 + .venv/bin/python，重启 agent 再 list。
C 参数越界（schema 立功）	TOOL 返回 ERROR: pin 非法 (40)。必须是 0-33 的整数，请对照 pins.md	模型猜错引脚。不需要人介入—— 自纠 模型读文本后改调 pin=2。这就是「错误返回可读文本」的意义。

三板斧 server 单独跑通吗 → agent 里 list 到吗 → 调用报错看 stderr P10 · 顺序不变

实验二：智能台灯

模拟器兜底 · 无硬件可上课

14 分钟 · 三层任务

场景 · SCENARIO

光敏电阻读数**低**时点亮 LED，亮度随环境光**反向调节**——一盏会自己看天色的台灯。

任务 · TASK

补全 `pwm_set` / `adc_read` 两个 TODO tool，让 **agent**——而不是你手工——点灯并**读回验证**。

验收 · ACCEPTANCE

`make check` 全绿——含 **HIL 断言**：查模拟器真实板载状态，不看你的代码脸色。

初始状态

BEFORE

18 passed + 8 failed · 全指向两个 TODO 桩

验收标准

AFTER

26 passed —— `make check` 全绿

实验材料

SCAFFOLD · 目录树

lesson2-mcp-lab/

- ├─ README.md 实验讲义 + FAQ
- ├─ simulator/ 板子模拟器
 - 内存态 GPIO/PWM/ADC · ADC 带噪声
- ├─ mcp_server/
 - └─ server.py
- 提示 ✖ `pwm_set` 桩 ✖ `adc_read` 桩
- ├─ firmware/
 - ├─ light_led.py 埋 2 bug
 - └─ pins.md 引脚约定表
- ├─ tests/
 - └─ test_hil.py 硬件在环 HIL
- └─ Makefile
 - `make sim / check / doctor`
- └─ teacher/ 学生勿看

三层任务

先会用，再会写，最后合起来
每层都有**机器可判**的验收

L0

约 4'
必做 · 用

注册 MCP，点亮 LED USE MCP

- 1 `make sim` 起模拟器 → `curl /state` 看到 `led_brightness` 字段
- 2 把 scaffold server 注册进你的 agent CLI
- 3 对 agent 说「把板载 LED 点亮，并读回确认」

验收——它调 `gpio_write(2, 1)`，且 `curl /state` 里 **led_brightness > 0**。

L1

约 6'
必做 · 写

补全两个 TODO tool WRITE MCP

- + **schema 校验**：`pwm_set` pin 0-33 / `duty` 0-100；`adc_read` 只认引脚 34
- + **错误处理**：`try/except BoardError` → 可读 **ERROR**：文本，不抛裸异常
- + **docstring**：参数含义 + 范围 + 单位（`duty %` · `raw` 0-4095 · 0-3.3V）

验收——`make check` **26 passed 全绿**（含 HIL 物理耦合断言）。

L2

4' + 课后
挑战 · 综合

接上你第 1 节课的 skill SKILL × MCP

把 skill 验证段换成 `flash` + 读回比对；修掉 `light_led.py` 的 2 个 bug（引脚号 + 类型）；让 agent 端到端完成「感光调光」。
验收——会话记录呈现 **写 → 查 → 验** 闭环，`led_brightness` 与固件设定一致；加分：`adc` 读数区间自动判亮度。

TIP · 卡住时 先查 README 讲义 FAQ 与**排错三板斧**（P10-15），再举手。

评分细则见下页 →

怎么评分

100 分制
每个得分点都有机器可判证据

评分项	验收标准	分值
L0 注册并点灯	make doctor 全√ · 会话有 gpio_write(2,1) 调用 · 读回 led_brightness > 0	20
L1 补全两个 tool	每个 tool 20: schema 10 + 错误处理 5 + docstring 5 · make check 全绿	40
L2 skill × MCP 端到端	固件版改造 10 + 修 2 bug 8 + 端到端 12 + 机器可判标准 5	40
+ 感光调光	adc 读数低→调高、高→调低，用 读数区间 自动判定	+5

合计 TOTAL · 封顶 100

100

L1 基本盘当场**机器可判**：make check 全绿 = 30 分到手；其余按证据核。

判定证据三件套

EVIDENCE

- 1

make check 输出
18+8 → 26 passed · 失败清单即修复输入
- 2

curl /state
板载真实状态：led_brightness / pwm / firmware
- 3

agent 会话记录
tool call 序列呈现 写 → 查 → 验 闭环

ZERO TOLERANCE

改测试凑通过 =

L1 清零

第 1 节课负面清单的延续——开场就声明。

带走的：一双手，一双眼

下节课：纪律 + 手
协同强化

TAKEAWAY · 核心主张

MCP 给 agent 装上 =

手 + 眼

WRITE PINS - READ BACK. 没有读回与日志，闭环就是瞎的。

- 1

docstring 即说明书 含义 / 范围 / 单位，写给模型看
- 2

错误返回可读文本 ERROR: 开头，不抛裸异常
- 3

感知类工具是闭环前提 read / logs / adc —— 先有眼，再有手
- 4

危险操作加确认 只暴露可逆操作；flash 类 confirm 必真

作业 · 下节课的输入

HOMEWORK

- 1

把本组 skill 改成固件版草稿
验证段换成 flash + 读回比对；只写 tool 名 + 判定标准，不复述参数细节。
- 2

保留补全后的 server.py
第 3 节课复用它做协同强化与故障注入——不重写。

两份作业都是下节课的输入 —— 全程迭代同一份作品。

NEXT · LESSON 03

Skill × MCP 协同 —— 纪律 + 手 = 可靠闭环，故障下也不崩

协同五模式 · 故障注入挑战 · 混沌开关 CHAOS=pin_stuck / serial_timeout / adc_drift

模拟器与命令速查

端口 8765 · 纯 stdlib
与 lessons/lesson2-mcp-lab 实测一致

命令 / ENDPOINT	作用	备注
MAKE · 验收链		
\$ make setup / doctor	建 venv 装依赖 / 环境自检	mcp 钉在 >=1.2, <2 (2.x 已移除 FastMCP)
\$ make sim / stop	后台起 / 停模拟器	日志 .sim.log · make check 不需要先 sim
\$ make check	一键验收: 26 passed 全绿	自带进程内模拟器 · 含 HIL 断言
HTTP · 直接 CURL		
GET /state	板载真实状态	led_brightness / pwm / firmware
POST /gpio_write	点灯: {"pin":2,"level":1}	L0 判据 = 之后 /state 亮度 > 0
GET /adc_read?pin=34	光敏读数	raw 0-4095 ↔ 电压 0-3.3V
GET /logs?limit=N	事件环形缓冲	跨复位保留 —— 故障诊断第一手证据

Pins

LED = GPIO2 (PWM 调光 duty 0-100) · 光敏 = 引脚 34 (唯一 ADC) · 暗环境基础读数约 800、满亮约 3000, 含 ±40 噪声 —— 判定用区间, 不做精确相等比较。