

作业 4（学期单人作业）：对战 AI

① 一个简单游戏：人对战AI

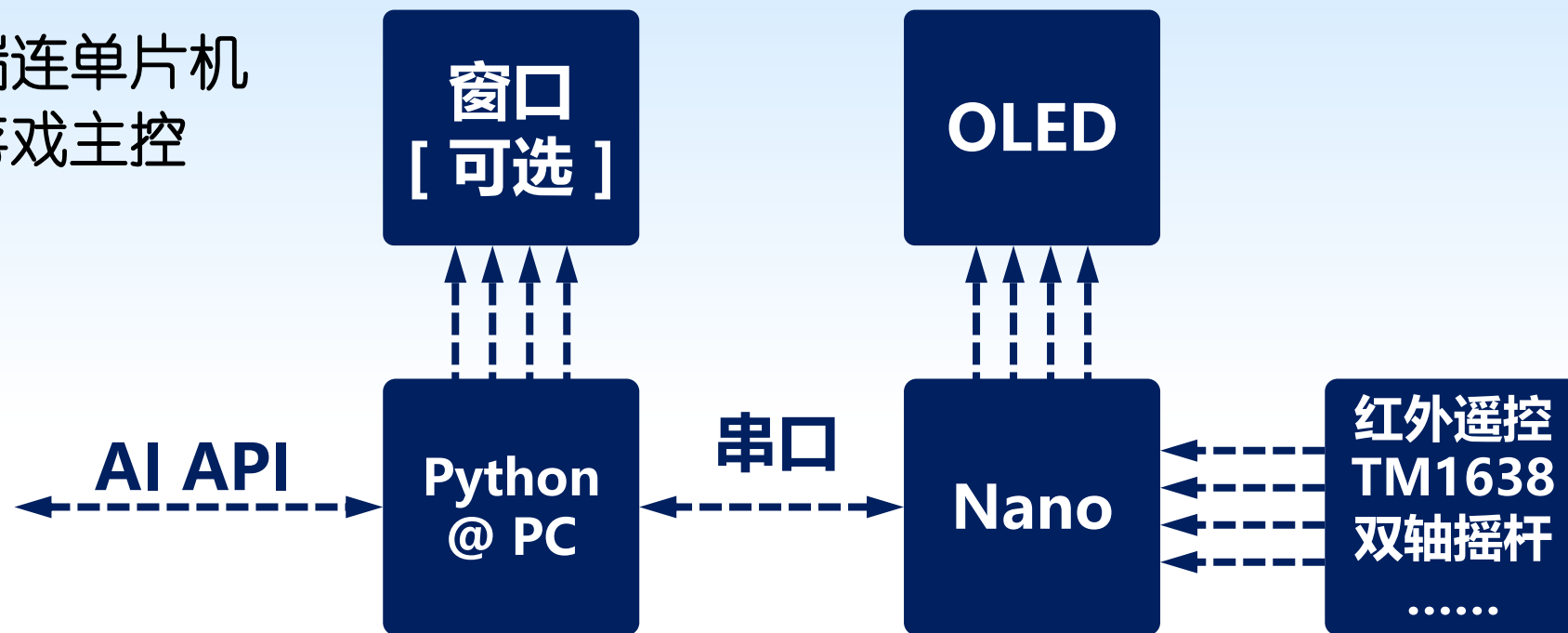
▶ Tik-tak-tok、扑克、猜数字、其它（可自定义游戏）

② Python:

▶ 一端连AI，一端连单片机
▶ 【可选】承担游戏主控

③ 单片机

▶ 玩家输入
▶ 输出到屏幕



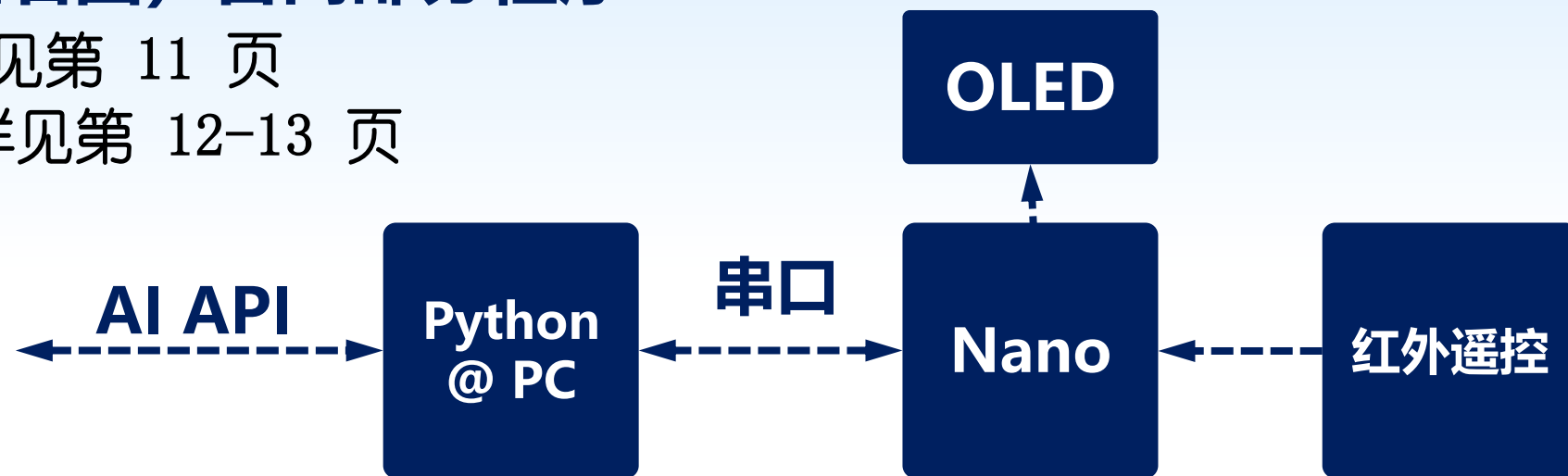
Demo: 与 AI 对战的井字棋

① 需要预先设定 AI 的 API: 详见第 4~7 页, 并提供3个调试程序

- ▶ API_Demo1: 查询 API 提供的大模型——说明见本文档第 8 页
- ▶ API_Demo2: 单次对话——说明见本文档第 9 页
- ▶ API_Demo3: 多轮对话——说明见本文档第 10 页

② 本示例 (如右图) 含两部分程序

- ▶ Python: 详见第 11 页
- ▶ Arduino: 详见第 12-13 页



Application
Programming
Interface

基于 API 的 AI 运用

注册
账号

申请
Key

安装
库

测试
脚本

多轮
对话



注册
账号

申请
Key

安装
库

测试
脚本

多轮
对话

Kimi.moonshot.cn

注册|登录

文件 拖进来；网址，发出来

快来解锁下这些用法~

注册
账号

申请
Key

安装
库

测试
脚本

多轮
对话

[Platform.moonshot.cn/console/api-keys](https://platform.moonshot.cn/console/api-keys)

浏览器地址栏显示: platform.moonshot.cn/console/api-keys

页面标题: Moonshot AI 文档 用户中心

左侧导航栏:

- 账户信息
 - 基本信息
 - 实名认证
 - API Key 管理
 - 用量限制
- 充值信息
 - 账户总览
 - 账户充值
 - 充值明细
 - 计费明细
 - 发票管理

API Key 管理 已保存

新建

您的密钥

请将此密钥保存在安全且可访问的地方。出于安全原因，您将无法通过您的 Moonshot AI 账户再次查看他。如果您丢失了此密钥，请删除它后生成一个新的。

sk-[redacted]

操作

编辑 删除

1

保存下来，遗失不补

注册
账号

申
K

Kimi.ai - 帮你更大的世界 × Moons

platform.moonshot.cn/conso

Moonshot AI 文档 用户中心

- 账户信息
 - 基本信息
 - 实名认证
 - API Key 管理
 - 用量限制
- 充值信息
 - 账户总览
 - 账户充值
 - 充值明细
 - 计费明细
 - 发票管理

用户等级	累计充值金额	并发	RPM	TPM	TPD
Free	¥ 0	1	3	32,000	1,500,000
Tier1	¥ 50	50	200	128,000	10,000,000
Tier2	¥ 100	100	500	128,000	20,000,000
Tier3	¥ 500	200	5,000	384,000	Unlimited
Tier4	¥ 5,000	400	5,000	768,000	Unlimited
Tier5	¥ 20,000	1,000	10,000	2,000,000	Unlimited

并发数 = 1

32000 Token / min

3 Requist / min

不限 Token / day

Key	操作
sk-Cs...7C2bc	编辑 删除

< 1 >

注册
账号

申请
Key

安装
库

测试
脚本

多轮
对话

兼容 OpenAI SDK

pip install openai

```
C:\Users\Administrator>pip install openai
Collecting openai
  Using cached openai-1.31.1-py3-none-any.whl (324 kB)
Requirement already satisfied: anyio<5,>=3.5.0 in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from openai) (4.3.0)
Requirement already satisfied: distro<2.0.0, >=1.7.0 in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from openai) (1.9.0)
Requirement already satisfied: httpx<1, >=0.23.0 in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from openai) (0.25.2)
Requirement already satisfied: pydantic<3, >=1.9.0 in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from openai) (2.6.2)
Requirement already satisfied: sniffio in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from openai) (1.3.0)
Requirement already satisfied: tqdm>4 in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from openai) (4.66.2)
Requirement already satisfied: typing-extensions in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from openai) (4.9.0)
Requirement already satisfied: idna<=2.10, >=2.0 in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from anyio<5,>=3.5.0->openai) (3.6)
Requirement already satisfied: exceptiongroup>=1.0.2 in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from anyio<5,>=3.5.0->openai) (1.2.0)
Requirement already satisfied: certifi in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from httpx<1, >=0.23.0->openai) (2024.2.2)
Requirement already satisfied: httpcore==1.* in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from httpx<1, >=0.23.0->openai) (1.0.4)
Requirement already satisfied: h11<0.15, >=0.13 in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from httpcore==1.*->httpx<1, >=0.23.0->openai) (0.14.0)
Requirement already satisfied: annotated-types>=0.4.0 in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from pydantic<3, >=1.9.0->openai) (0.6.0)
Requirement already satisfied: pydantic-core==2.16.3 in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from pydantic<3, >=1.9.0->openai) (2.16.3)
Requirement already satisfied: colorama in c:\users\administrator\appdata\local\programs\python\python310\lib\site-packages (from tqdm>4->openai) (0.4.6)
Using cached openai-1.31.1-py3-none-any.whl (324 kB)
Installing collected packages: openai
Successfully installed openai-1.31.1
```

注册
账号

申请
Key

安装
库

测试
脚本

实用
程序

前面保存的Key

API_Demo1.py

```
from openai import OpenAI

client = OpenAI(
    api_key = "sk-*****",
    base_url = "https://api.moonshot.cn/v1",
)

model_list = client.models.list()
model_data = model_list.data

for i, model in enumerate(model_data):
    print(f"model[{i}]:", model.id)
```

base url
不同公司的服务器

注册
账号

申请
Key

安装
库

测试
脚本

实用
程序

API_Demo1.py

```
from openai import OpenAI
client = OpenAI(
    api_key="sk-*****",
    base_url="https://api.moonshot.cn/v1",
)
completion = client.chat.completions.create(
    model="moonshot-v1-auto",
    messages=[
        {"role": "user", "content": "用8个字回答：天是什么色？"},
    ],
    temperature=0.3,
    stream=False
)
answer = completion.choices[0].message
print(answer.content)
```

聊天对话内容

思维发散程度 [0,1]
建议取 0.3 左右

并非“流模式”：后者指
文字逐渐一点一点地呈现

AI 的回答在此处

多轮对话 API_Demo3.py

```
from openai import OpenAI
client = OpenAI(
    api_key="sk-*****",
    base_url="https://api.moonshot.cn/v1",
)
history = [
    {"role": "system", "content": "你是个聪明玩家。"}
]

def chat(query, history):
    history.append({
        "role": "user", "content": query
    })
    completion = client.chat.completions.create(
        model="moonshot-v1-auto",
        messages= history,
        temperature=0.3,
        stream=False
    )
    answer = completion.choices[0].message.content
    history.append({
        "role": "assistant", "content": result
    })
    return answer
```

```
print(chat("地球有多大?", history))
print(chat("月球呢? 哪个大?", history))
... ..
```

总体说来，是在 `API_Demo3` 上添加了串口通信的部分

- 在 `SystemPrompt` 中约定了和 `AI` 的规则（游戏规则，输入和输出文字协议）
- 在 `While` 循环中检查是否收到串口数据
 - ▶ 如果收到复位：则复位
 - ▶ 如果收到串口数据：先打印显示（调试），然后调用 `Chat` 函数，其中
 - 1) 先把用户提示词记录入 `History` (role为用户)
 - 2) 调用 `AI`
 - 3) 获得 `AI` 的回答后，记录入 `History` (role为assistant)
 - ▶ 在`chat`返回后，把 `AI` 的走棋数据通过串口提交给单片机

在最后一页的 `Demo_TTT.ino` 下载后，和单片机一起合作完成任务

读红外遥控器 (例程)

工具 > 管理库: 搜索 IRLremote > 安装 IRLremote by Nicohood

示例 > IRLremote > Receive

- 开启 Arduino 的串口监视器
- 遥控器对准接受模块后, 按下遥控器每个按钮的信息形如:

Address: 0xFF00

Command: 0x52

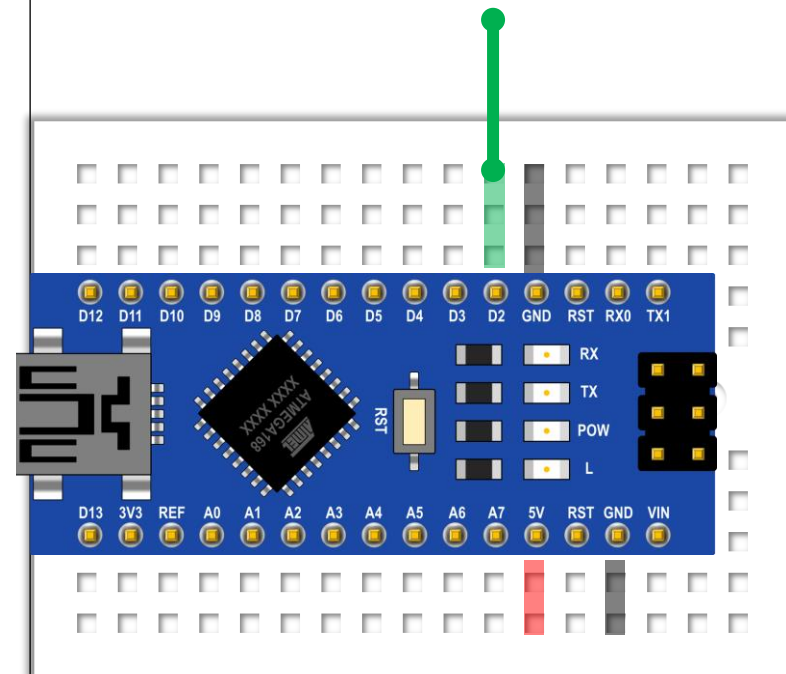
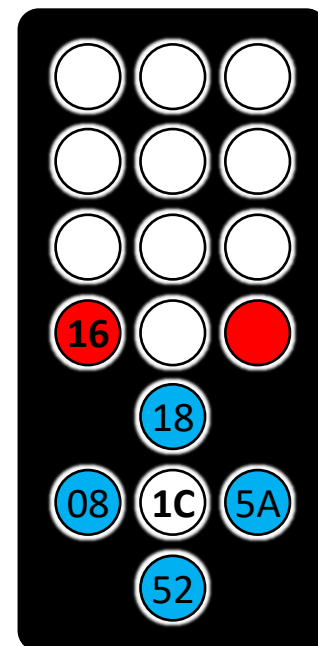
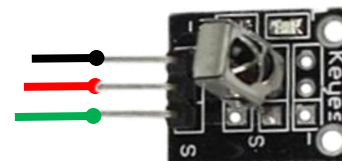
Address: 0xFFFF

Command: 0x0

记录第一个 Command 即可

- 本 demo 中总共记录了六个键:
请核实自己的遥控器是否相同。
如不相同, 则需修改 Demo 中
switch 的入口数值

● *	: 0x16
● 上	: 0x18
● 下	: 0x52
● 左	: 0x08
● 右	: 0x5A
● OK	: 0x1C



SSD1306_128x64_i2c

工具 > 管理库：搜索 SSD1306 > 安装 Adafruit_ssd1306

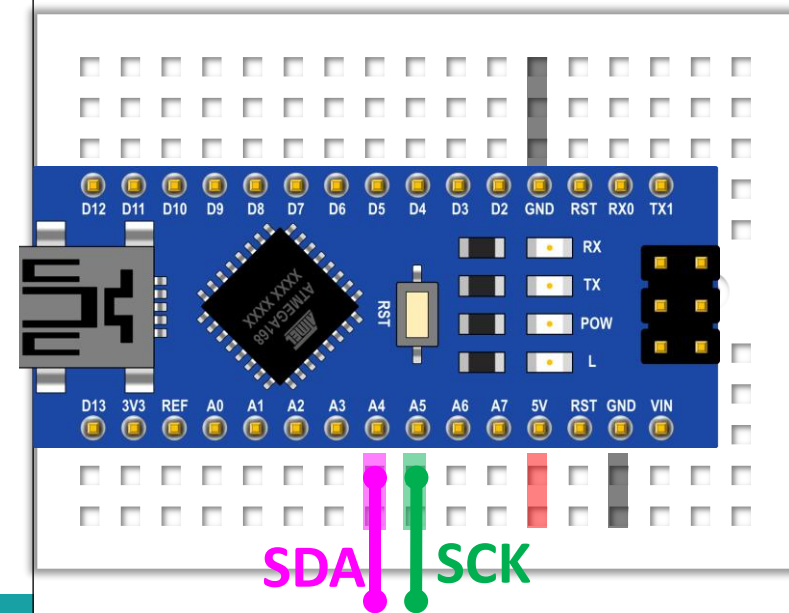
提示相关库时，选择需要安装 Adafruit-GFX-Library

载入示例： Adafruit SSD1306 > SSD1306_128x64_i2c

.....

```
#define SCREEN_ADDRESS 0x3C // 0x3D
```

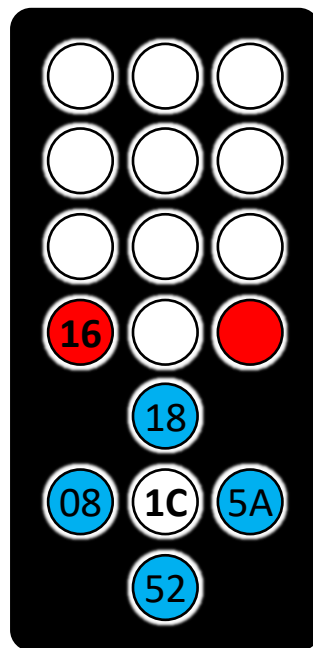
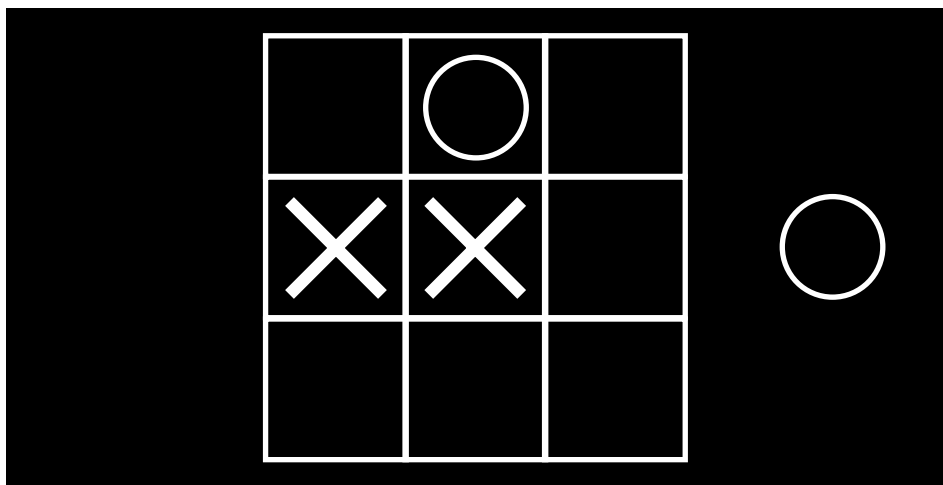
.....



OLED 128*64

运行 Demo_TTT

- 单机可以玩耍（不运行 API_DemoAgent）
- 可以用上下左右移动光标
- 用 OK 落子
- 用 * 复位游戏



运行 API_DemoAgent.py 后，合作完成本 demo

