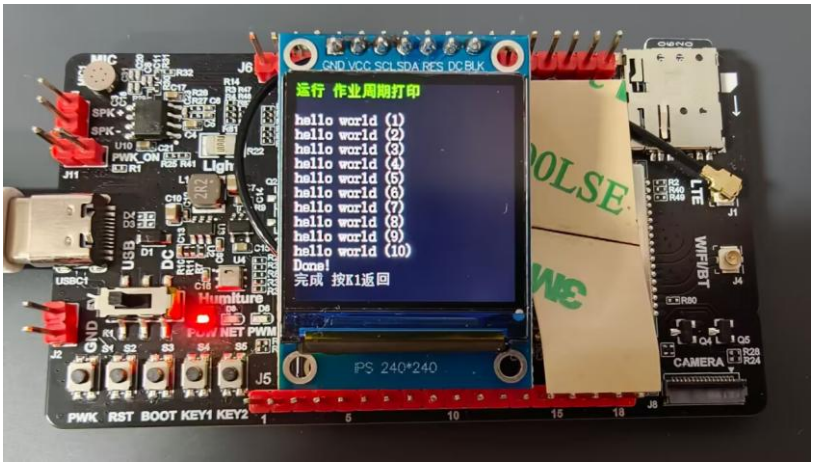
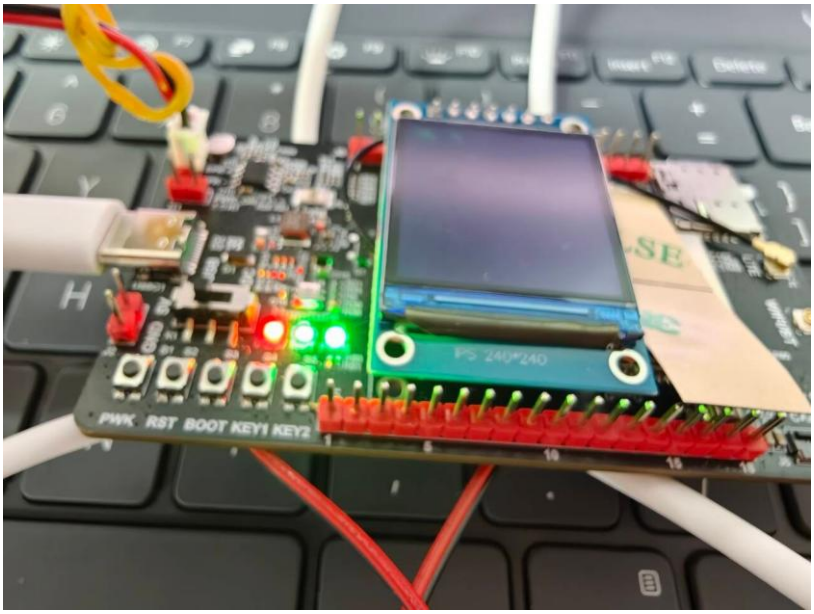
移远通信培训周报

姓名：陈文涛　　学号：2601401737　　班级：集电 26401　　时间：2026.11.19

任务完成情况	**任务 1：周期性打印 HelloWorld** 完成度：100% 我的工作成果：完成 VSCode 与 QPYcom 环境配置，打通 AI 辅助开发流程并生成基础代码。在 EC600M 上完成循环打印：每 2 秒输出一次、共 10 次，最后打印“Done!”。代码按 main.py / config.py / driver.py 拆分，经 QPYcom 上传到 /usr 运行。 **任务 2：按键触发定时 LED 闪烁** 完成度：100% 我的工作成果：按键采用上拉与下降沿触发；中断内仅关闭中断并启动 50ms 消抖定时器，确认按下后 Timer0 每 200ms 翻转 LED，Timer1 单次 10 秒超时停止并重新使能中断，ISR 保持最短路径。 **任务 3：板载程序菜单与中文界面（自选扩展）** 完成度：100% 我的工作成果：开机扫描 /usr 生成菜单，K1 切换、K2 运行，运行时把 print 重定向到屏幕并可按 K1 返回。界面全中文：自制 16×16 宋体字模 92 个，按 UTF-8 字节切分实现中英混排，菜单为“中文备注 + 文件名”双列显示，并自动隐藏库文件与入口。 **任务 4：4G 联网信息屏（自选扩展）** 完成度：100% 我的工作成果：完成 4G 拨号获取 IP、信号强度查询与网络授时。实测 dataCall 获取内网 IP，CSQ 22~27；nitzTime 的时间基准只下发一次，改用 ticks_ms 自行累加走秒避免时间冻结，屏幕每 10 秒刷新 IP / CSQ / 本地与 UTC 时间。 **任务 5：板载小游戏（贪吃蛇 / 打砖块 / 离线小恐龙）** 完成度：100% 我的工作成果：在 240×240 屏幕与双按键约束下实现三个可玩小游戏。贪吃蛇 15×15 网格双键转向；打砖块用脏区域擦除重绘实现挡板与球体动画，含六色砖墙、三段反弹与三条命；离线小恐龙实现跳跃物理、随机障碍、计时加速与彩色天空/云/远山。音效统一用 audio.Audio.aud_tone_play(tone, ms)。 **任务 6：五个实验作业的落地实现** 完成度：80%（小智受固件限制） 我的工作成果：完成环境监控、光控窗帘、温湿度记录仪、LVGL 联网天气屏四个实验的实机运行，并封装 sensors.py 抽象层（自动探测 AHT20 / ADC / PWM，无硬件时

	降级模拟值）。实测 AHT20 24.2℃、湿度 38~41%；光敏 ADC 51mV 约 14 lux；天气经 HTTP 取自 open-meteo（武汉室外 14.9℃ / 46%）。小智 AI 因固件缺 Opus 与唤醒接口无法运行，已出依赖自检与协议骨架。  图 1 任务 1 串口输出：hello world (1)~(10) + Done!  图 2 任务 2 实物调试：按键触发后 LED 点亮

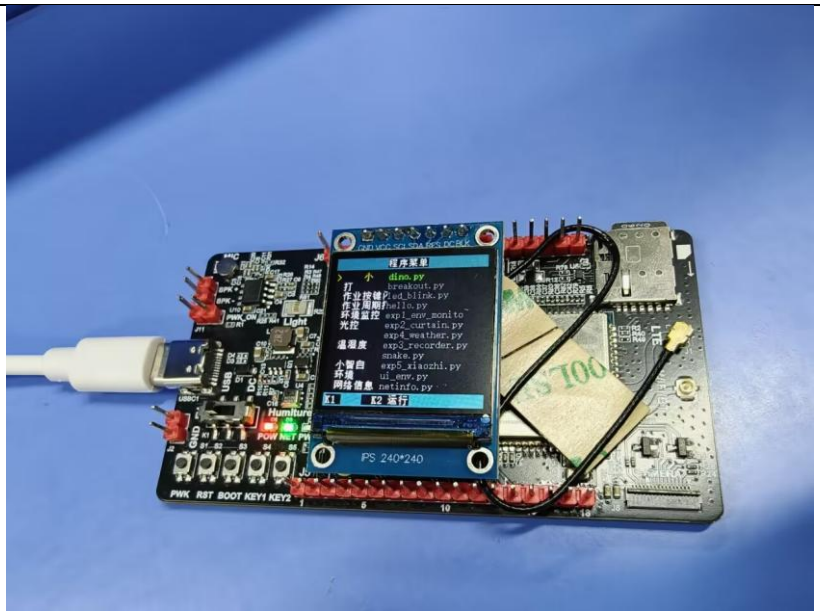


图 3 任务 3 中文程序菜单

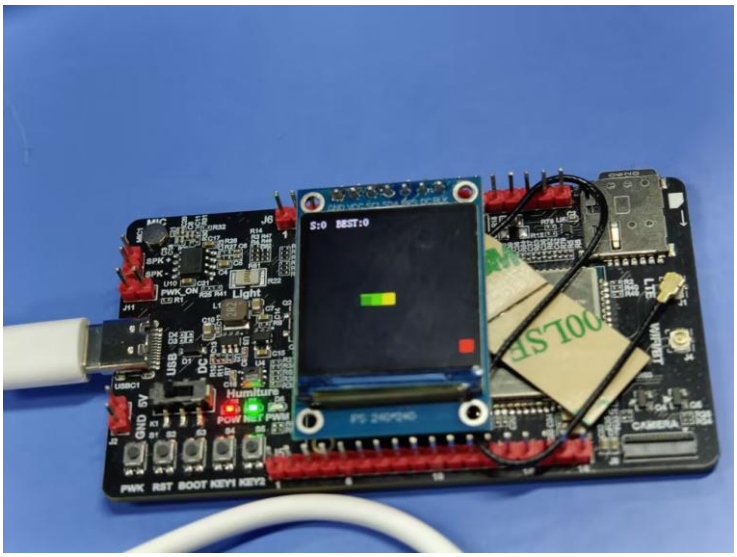


图 4 任务 5 贪吃蛇

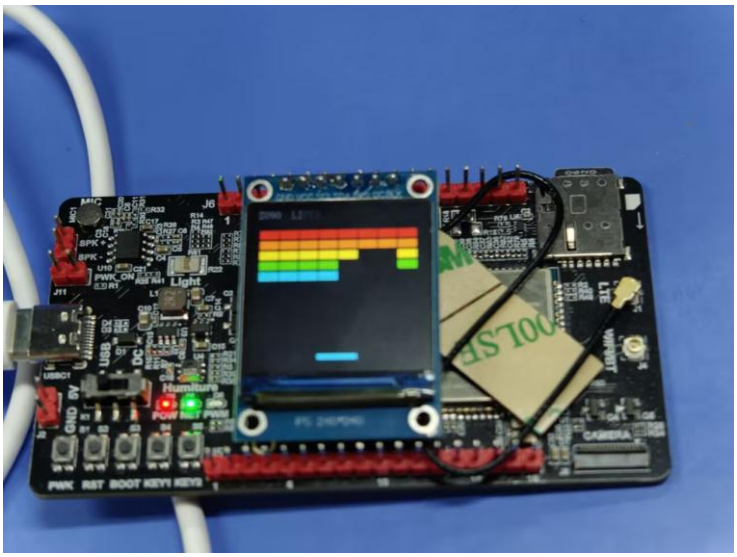
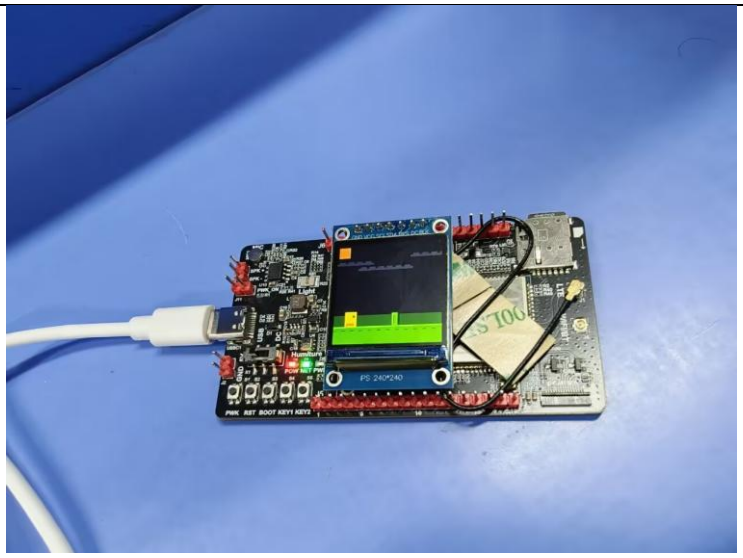


图 5 任务 5 打砖块

	 图 6 任务 5 离线小恐龙（彩色背景）
知识总结	**1.1 开发环境与 AI 辅助工具链搭建** 完成 VSCode 与 QPYcom 配置，打通“需求描述—代码生成—设备联调—问题分析”的流程。AI 能提高编码效率，但生成代码必须对照官方文档核对引脚、参数与固件限制，最终以串口日志和硬件现象为准。 **1.2 QuecPython 基础编程与设备联调** 首个项目用 utime.sleep() 实现 2 秒间隔打印，按模块拆分为 main.py / config.py / driver.py 上传到 /usr 运行。按键实验使用 GPIO12 按键与 GPIO24 LED，中断内关闭 ExtInt 并启动 50ms 消抖定时器，确认后 Timer0 周期翻转、Timer1 超时停止并重新使能中断。 **1.3 中文显示与字模制作** 官方 LCDPublic 字库仅收录 6 个汉字，界面文案无法直接显示。用 PC 端宋体渲染生成 16×16 点阵，按行扫描转为 32 字节一组的字模字典，共制作 92 个常用汉字；绘制时按 UTF-8 字节宽度切分实现中英混排，并做“扩展字库→官方字库→ASCII”三级回退。 **1.4 4G 联网与网络授时** dataCall.getInfo(1, 0) 获取拨号 IP，net.csqQueryPoll() 读取信号等级，net.nitzTime() 返回时间字符串与 epoch 秒。实测该 epoch 已是本地时间，UTC 需减时区偏移；时间基准只下发一次，故用 ticks_ms() 累加走秒。
学习心得	这周主要在折腾开发环境和跑通第一批代码。虽然大部分时间在装软件、配环境，但实打实踩了几个坑之后收获很大。 **2.1 错误总结** 最大的认知变化是“纯软件”和“搞硬件”的区别：写完 Hello World 直接按编辑器运行会疯狂报错，因为代码是给 EC600M 跑的，必须用 QPYcom 传到板子 /usr 再执行。LED 这题又踩一坑——周期定时器 stop() 之后不重新 start()，下次回调根本

	不触发，最后靠消抖回调里加打印才定位到定时器状态没复位。 **2.2 学习总结** 就为了两秒打印一句话，居然要建好几个文件；但看完规范才明白这是在给后面接传感器、写复杂逻辑打基础。把配置、驱动和主程序分开的习惯要早点养成。 **2.3 排障记录（本周踩坑）** ① 交互区多行粘贴只执行首行，必须逐行输入；② 该固件 LCD 仅 8×16 ASCII 字号生效，16/24 号静默失败；③ 中文经串口传输被替换为问号，改走文件传输并按 UTF-8 字节切分；④ 单条上传命令过长（720 字符）会被 REPL 静默截断，文件写成 0 字节，改 120 字节分块后正常；⑤ 修改模块文件后必须软复位，否则 import 命中内存中的旧模块。  图 7 调试过程：LED 闪烁程序运行中
课后思考（作业）	**思考题 1：为什么 ISR 中要先 exti.disable() 再启动 osTimer，而不是直接执行业务逻辑？** 答：按键触点会在几毫秒内反复通断，产生一串边沿，中断被连续触发多次。ISR 里直接跑业务会被抖动打乱状态机，且耗时操作阻塞系统、丢掉后续中断。所以 ISR 只做两件最轻的事：关闭中断防重复触发、启动 osTimer 做 50ms 延时；真正判断与业务放到定时器回调这一普通上下文中执行。 **思考题 2：debounce_cb 中为什么要重新读取引脚电平？** 答：中断只说明“此刻出现了下降沿”，不代表 50ms 后按键仍被按下。抖动、松手、干扰都可能让电平回到高电平。因此消抖回调必须重新读一次引脚（上拉模式下 0 = 真按下），确认后才执行业务；不确认则重新使能中断等待下一次有效按键。 **思考题 3：如何实现“第二次按键提前停止闪烁”？** 答：在消抖回调中加分支：若电平仍为低且 blinking == True，即按“第二次按键”处理——先 blink_timer.stop()、timeout_timer.stop()，再 led.write(0)、blinking = False 并打印提前停止信息，最后重新使能按键中断。两个定时器都要 stop()，否则残留的周期回调会继续翻转 LED。