



**《嵌入式系统》**

**课程实验报告**

姓名：苏一涵

学院：信息学院

系：软件工程

专业：软件工程

学号：36720232204041

2025年9月X日

**第1次实验 STM32实验**

1. **实验设备**

**（1）PC微机**

**（2）嵌入式系统综合实验箱（FS3399M4）**

1. **实验内容**
   1. **在实验箱上运行STM32实验的15个样例。请采用屏蔽拷贝（或拍照）的方式，将实验结果黏贴到实验报告中，并对有关实验结果进行文字说明。**

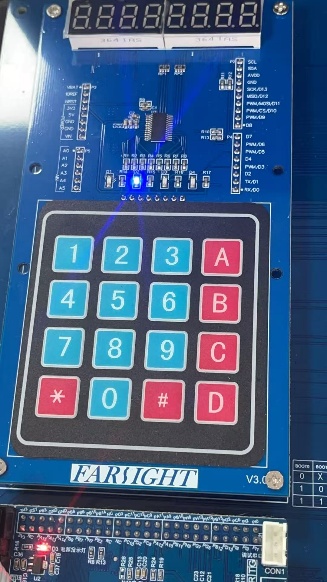
* **实验样例1 LED灯**

**实验原理：**

**LED 灯的控制核心基于 STM32 的GPIO（通用输入输出口）外设。实验中，首先将控制 4 个 LED 灯的 STM32 GPIO 引脚配置为推挽输出模式（根据 LED 与 GPIO 的电路连接，若为共阳极 LED，GPIO 输出低电平时 LED 点亮；若为共阴极，GPIO 输出高电平时 LED 点亮）。程序运行后，通过延时函数（如HAL\_Delay()）控制 GPIO 引脚周期性切换高低电平，使 4 个 LED 灯按设定顺序循环亮灭；按下 Reset 键时，MCU 重启并重新执行 LED 控制程序，从而观察到循环亮灭现象。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；观看实验箱底板上的4个LED灯（位于数码管与小 键盘之间）是不是循环亮/灭**

****

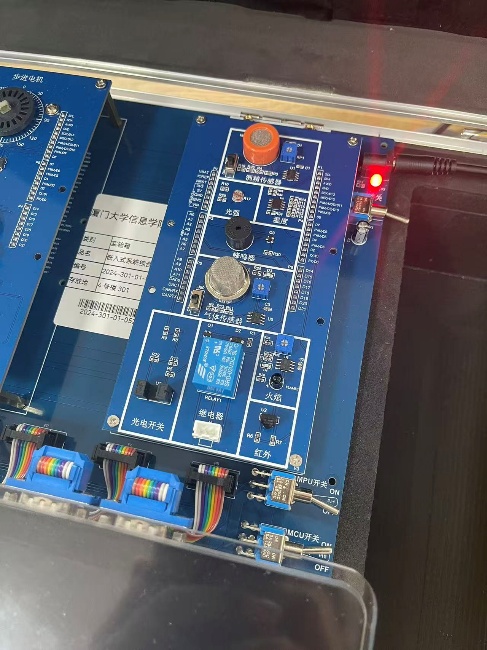
* **实验样例2 蜂鸣器**

**实验原理：**

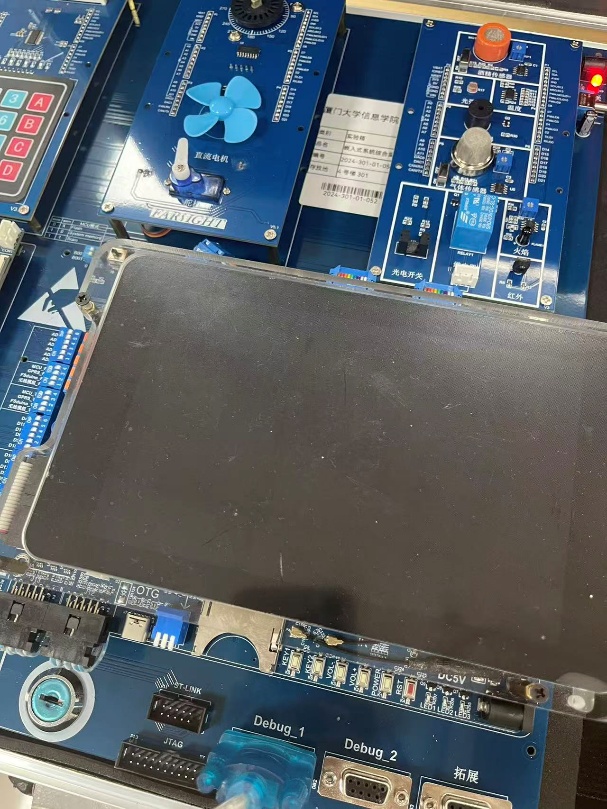
**蜂鸣器控制依赖 STM32 的 GPIO 输出功能，实验中使用的为有源蜂鸣器（无需外部方波驱动，通电能直接发声）。首先将连接蜂鸣器的 GPIO 引脚配置为推挽输出模式，程序通过定时切换 GPIO 引脚的高低电平实现蜂鸣器的 “响 / 停” 循环：当 GPIO 输出高电平时，蜂鸣器线圈通电发声；输出低电平时，线圈断电停止发声，配合 1 秒延时函数（HAL\_Delay(1000)），实现 “响 1 秒、停 1 秒” 的循环效果。若运行其他程序（如 LED 灯程序），该 GPIO 引脚不再输出蜂鸣器控制信号，故蜂鸣器不响。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；观看实验箱底板上蜂鸣器是不是响，1秒后，又不 响，循环反复**

****

**要执行其它的程序，蜂鸣器不响（如LED灯程序）**

****

* **实验样例3 小键盘控制直流电机**

**实验原理：**

**实验涉及小键盘输入检测与直流电机驱动两大核心逻辑：**

**（1）小键盘检测：小键盘的每个按键对应 STM32 的一个 GPIO 输入引脚，引脚配置为上拉输入模式（防止引脚浮空导致误触发），程序通过循环查询GPIO 引脚电平判断按键是否按下（按下时引脚电平变为低电平），识别 “1、2、3” 及其他按键；**

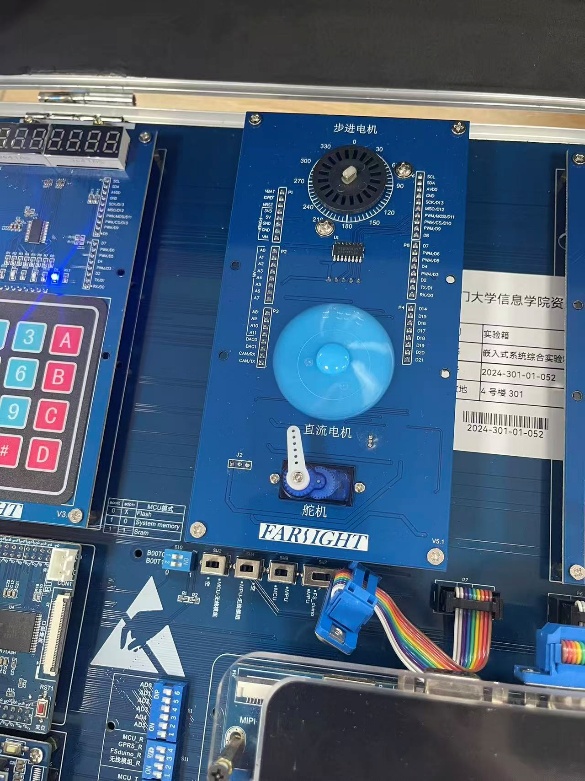
**（2）直流电机驱动：直流电机需通过电机驱动模块（如 L298N）连接 STM32（避免 MCU 引脚直接驱动电机），STM32 通过 2 个 GPIO 引脚输出 “方向控制信号”（如 GPIO1 高、GPIO2 低对应顺时针，反之逆时针），1 个 GPIO 引脚输出 “使能信号”（高电平电机运转，低电平停止）；**

**（3）逻辑联动：按下 “1” 键时，程序控制 GPIO 输出顺时针转动信号，电机顺时针转；“2” 键对应逆时针信号；“3” 键及其他键对应使能信号置低，电机停止；同时通过USART 串口将按键对应的控制状态（如flag=1）发送到串口调试助手，实现状态可视化。**

**实验结果：**

**MCU开发板上的Reset键，运行程序；按小键盘上的数字键“1、2、3”，以及其它13个键， 观看直流电机的转动情况，以及串口调试助手接收框中的显示内容**

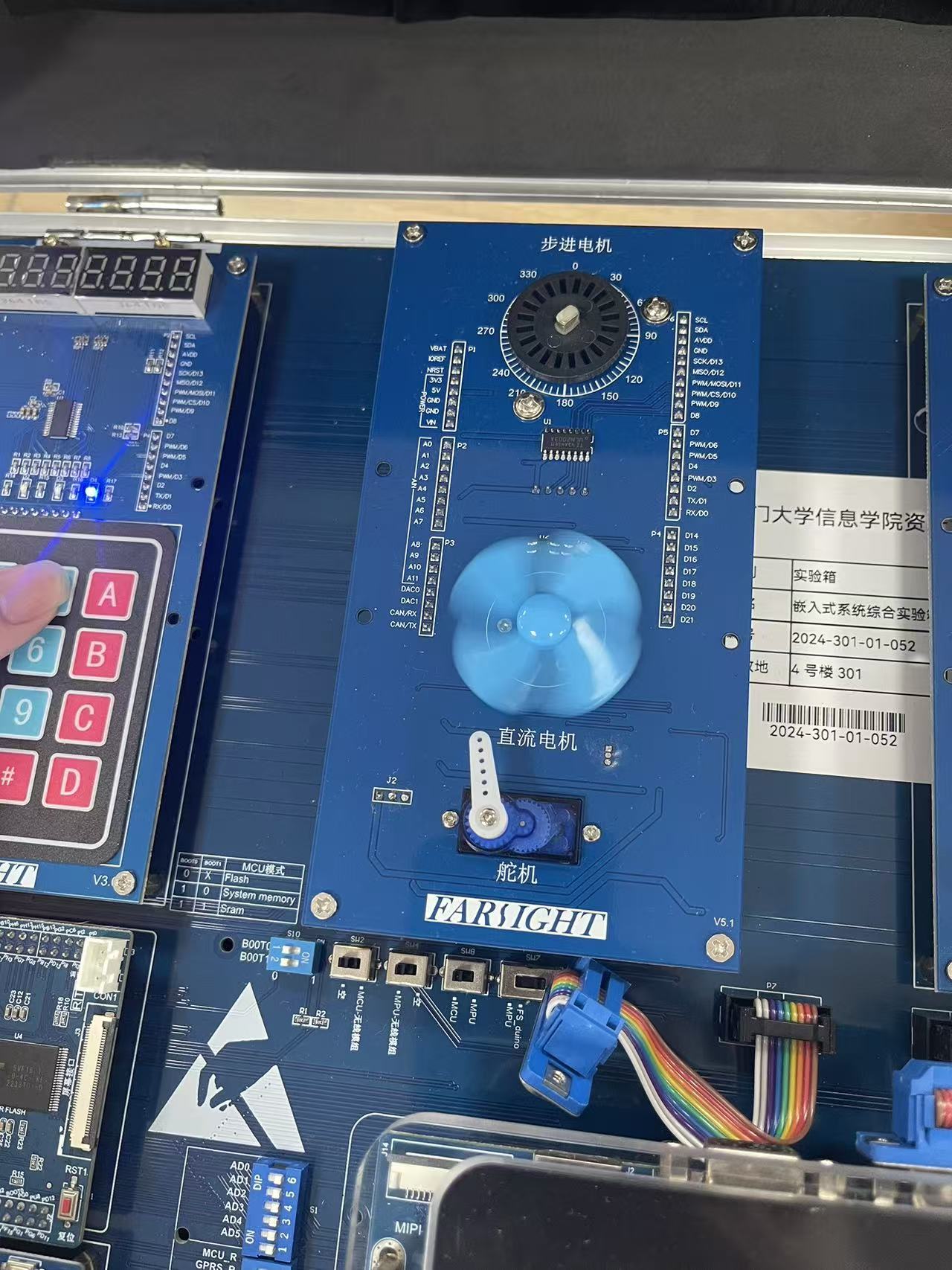
**1键：顺时针转**

****

**2键：逆时针转**

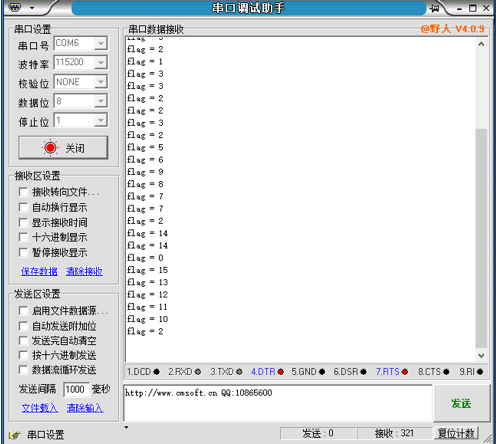
****

**3键：停止转**

****

**其余键：停止转**

****

****

* **实验样例4 陀机**

**实验原理：**

**舵机的核心控制信号为PWM（脉冲宽度调制）信号，实验通过 STM32 的定时器外设（如 TIM2）输出 PWM 实现舵机转动：**

**（1）舵机原理：舵机需 50Hz（周期 20ms）的 PWM 信号驱动，PWM 高电平时间决定转动角度（如 0.5ms 高电平对应 0°，1.5ms 对应 90°，2.5ms 对应 180°）；**

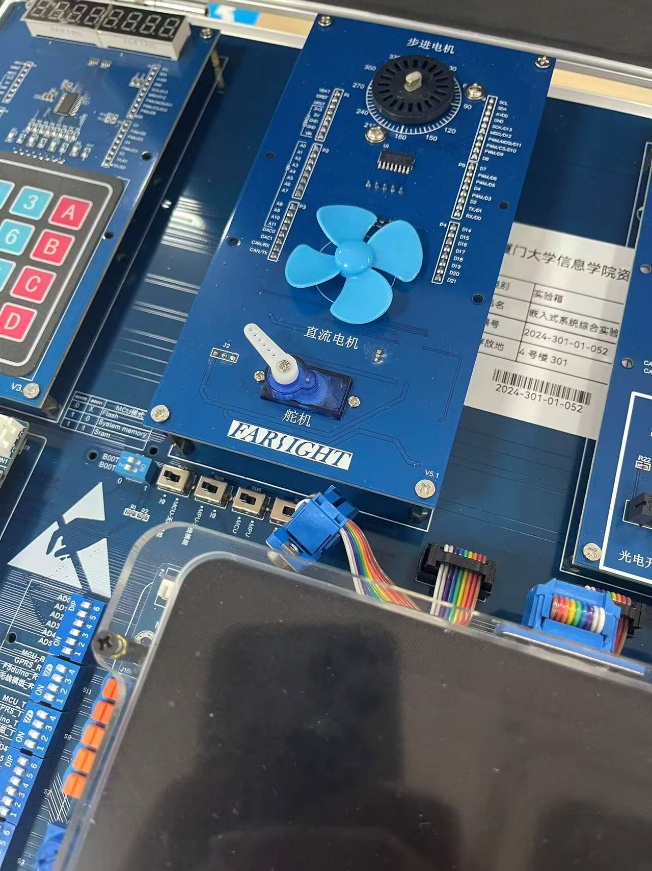
**（2）STM32 配置：将定时器配置为 “PWM 输出模式”，设置定时器预分频器和自动重装载值，使 PWM 周期固定为 20ms；通过调整比较寄存器值改变 PWM 高电平时间，从而控制舵机转动角度；**

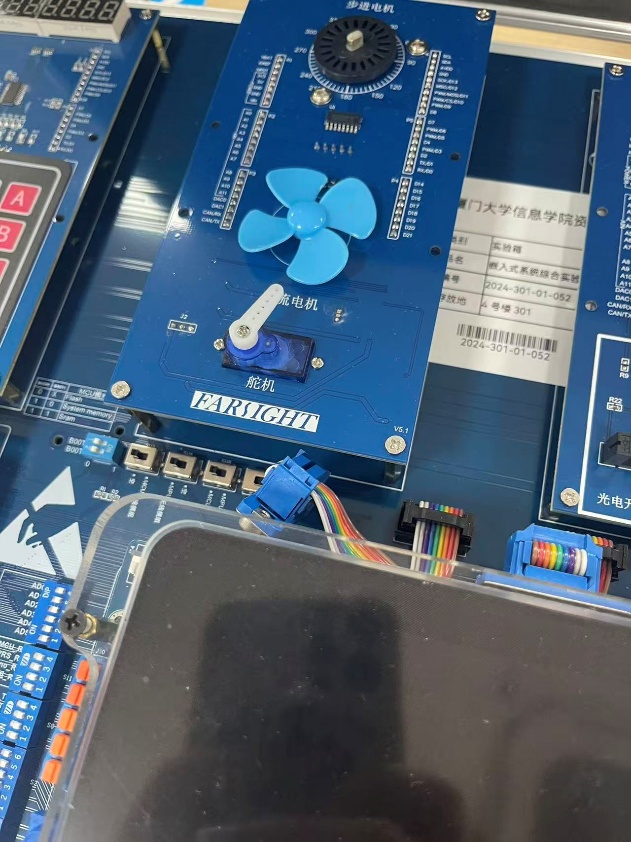
**（3）程序逻辑：按下 Reset 键后，程序循环调整定时器比较寄存器值，使 PWM 高电平时间在 0.5ms~2.5ms 之间切换，舵机随之左右转动（如从 0° 转到 180°，再转回 0°）。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，实验箱上的陀机会左右转动**

****

****

****

* **实验样例5 步进电机**

**实验原理：**

**步进电机通过 “绕组依次通电” 实现分步转动，实验核心是相序控制与 GPIO 输出：**

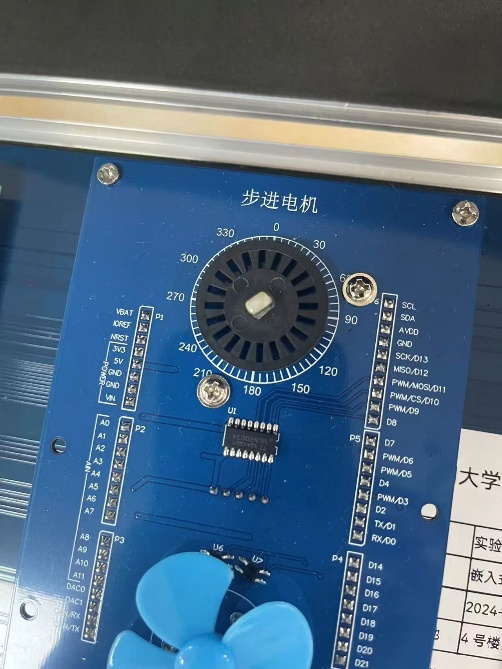
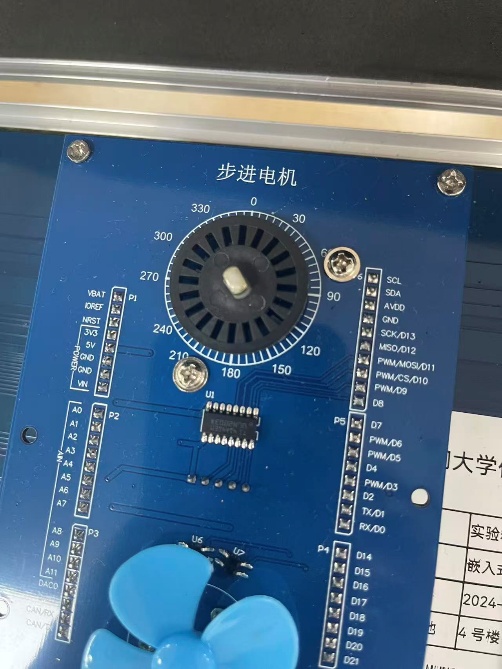
**（1）步进电机特性：实验用步进电机为四相电机（如 A、B、C、D 相），需按固定相序（如 “单四拍：A→B→C→D” 或 “双四拍：AB→BC→CD→DA”）给绕组通电，每通一次电电机转动一个固定步距角；**

**（2）STM32 控制：将连接电机四相绕组的 4 个 GPIO 引脚配置为推挽输出模式，程序中定义预设的相序数组，通过循环遍历数组，按顺序给对应 GPIO 引脚输出高电平（其余引脚低电平），驱动绕组依次通电；**

**（3）转动方向：程序固定输出 “顺时针相序”（如 A→B→C→D），故步进电机通电后持续顺时针转动，按下 Reset 键后 MCU 重启，重新执行相序控制程序。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，实验箱上的步进电机会顺时针转动**

****

* **实验样例6 串口通信**

**实验原理：**

**实验基于 STM32 的USART（通用同步异步收发器）外设实现双向数据传输，核心是串口参数配置与数据收发：**

**（1）串口参数配置：USART 配置为 “波特率 115200bps、数据位 8 位、停止位 1 位、无校验位（NONE）”（与串口调试助手参数一致，避免数据传输错误），GPIO 引脚分配为 “TX（发送）” 和 “RX（接收）”；**

**（2）数据收发逻辑：程序初始化 USART 后，启用 “接收中断” 或 “查询式接收”；当串口调试助手发送 “12345” 等字符时，STM32 通过 RX 引脚接收数据，再通过 TX 引脚将接收的数据回传至调试助手，最终在接收框中显示发送的字符；若串口端口未正确打开（如 COM6 被占用），则会提示 “Cannot open COM port” 错误。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，在串口调试助手的发送框中输入“12345” （也可以是其他字符），按“发送”按钮，观看接收框中的内容**

****

* **实验样例7 2个按键（查询方式）0**

**实验原理：**

**实验通过GPIO 查询式输入检测按键状态，并通过串口反馈结果，原理如下：**

**（1）按键硬件与 GPIO 配置：K3、K4 按键分别连接 STM32 的 2 个 GPIO 引脚，引脚配置为上拉输入模式（按键未按下时，引脚由内部上拉电阻拉为高电平；按下时，引脚接地变为低电平）；**

**（2）查询检测逻辑：程序通过无限循环持续读取 2 个 GPIO 引脚的电平，当检测到 K3 引脚变为低电平时，判定 “K3 按下”，通过 USART 串口发送 “KEY3 is pressed”；当电平恢复高电平时，发送 “KEY3 is released”；K4 按键检测逻辑同理；**

**（3）串口反馈：按键状态通过串口实时传输到调试助手，故能在接收框中观察到 “按下 / 释放” 的文字提示。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，按实验箱上K3、K4键，观看串口调试助手 接收框中显示的内容**

****

* **实验样例8 2个按键（中断方式）**

**实验原理：**

**与查询方式的核心区别是 “通过中断检测按键”，避免循环查询占用 CPU 资源，原理如下：**

**（1）外部中断配置：将 K3、K4 对应的 GPIO 引脚配置为外部中断模式，触发方式设为 “下降沿触发”（按键按下时电平从高变低，触发中断）；同时配置中断优先级（确保按键中断能被正常响应）；**

**（2）中断服务函数：当按键按下触发中断时，MCU 暂停当前任务，跳转到中断服务函数；在函数中通过 “读取 GPIO 电平” 或 “中断标志位” 判断是 K3 还是 K4 触发中断，再通过 USART 串口发送对应的 “按下 / 释放” 信息；**

**（3）中断退出：发送完成后，清除中断标志位，MCU 返回继续执行原任务，实现按键状态的实时检测与反馈。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，按实验箱上K3、K4键，观看串口调试助手 接收框中显示的内容**

****

* **实验样例9 继电器**

**实验原理：**

**继电器控制依赖 STM32 GPIO 输出与串口指令解析，原理如下：**

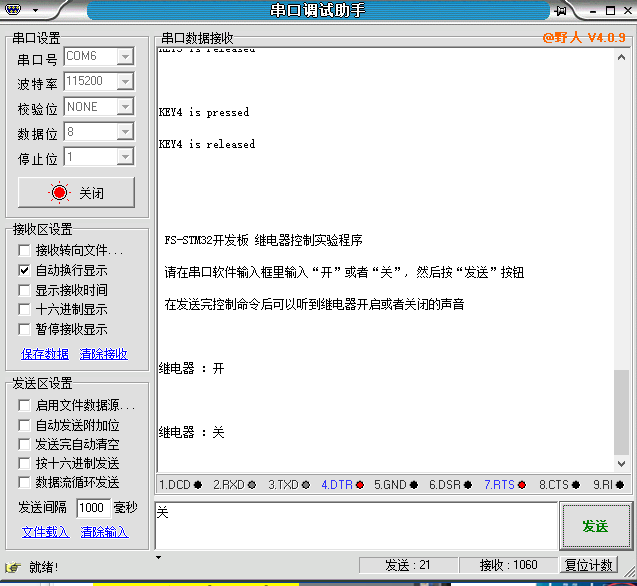
**（1）继电器驱动：继电器线圈需较大电流（MCU 引脚无法直接驱动），故通过三极管放大电路连接 GPIO 引脚；GPIO 输出高电平时，三极管导通，线圈通电，继电器触点吸合（发出 “咔哒” 声）；输出低电平时，线圈断电，触点断开；**

**（2）串口指令解析：程序初始化 USART 后，持续接收串口调试助手发送的指令（“开” 或 “关”）；通过字符串匹配判断指令类型：接收到 “开” 时，控制 GPIO 输出高电平，继电器打开；接收到 “关” 时，GPIO 输出低电平，继电器关闭；**

**（3）状态反馈：继电器动作时的 “响声” 是触点吸合 / 断开的物理反馈，配合串口指令实现精准控制。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，在串口调试助手的发送框中输入“开”，按发 送，则打开继电器（听到一个响声）；输入“关”，按发送，则关闭继电器（听到一个响声）**

****

* **实验样例10 定时器**

**实验原理：**

**实验通过 STM32 的定时器定时中断控制 LED 闪烁，核心是定时器周期配置与中断联动：**

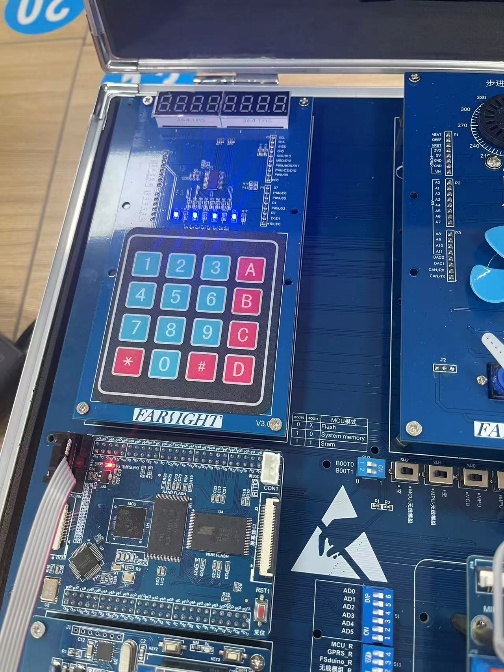
**（1）定时器配置：选择通用定时器（如 TIM3），通过设置 “预分频器” 和 “自动重装载值” 确定定时周期（例如：定时器时钟频率 72MHz，预分频器设为 7199、自动重装载值设为 9999，定时周期 =（7199+1）×（9999+1）/72MHz=1 秒）；**

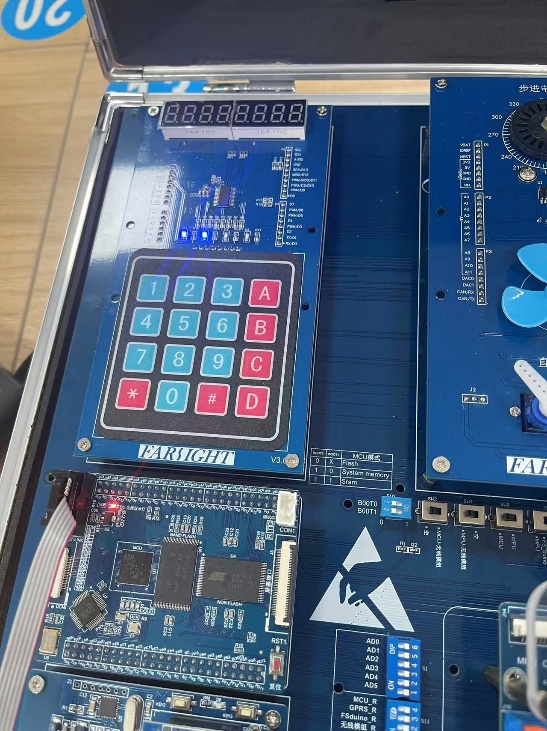
**（2）定时中断使能：配置定时器为 “定时中断模式”，启用中断并设置优先级；当定时器计数达到自动重装载值时，触发定时中断；**

**（3）LED 控制逻辑：在定时中断服务函数中，通过 “GPIO 电平翻转函数”（如HAL\_GPIO\_TogglePin()）切换 LED 对应的 GPIO 电平，每触发一次中断（1 秒），LED 状态翻转一次，实现 “亮 1 秒、灭 1 秒” 的定时闪烁效果。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，实验箱上的LED灯会定时亮/灭（闪烁）**

****

****

* **实验样例11 小键盘/数码管**

**实验原理：**

**实验涉及 “小键盘输入检测” 与 “数码管显示” 的联动，原理如下：**

**（1）小键盘检测：同实验样例 3，通过 GPIO 上拉输入 + 循环查询识别按键，记录按键对应的数值（如 “1”“2” 等）；**

**（2）数码管显示：数码管为共阴极 4 位数码管（或共阳极，需对应调整电平），通过 “段选” 和 “位选” 控制显示：**

* + **段选：8 个 GPIO 引脚控制数码管的 “a~g、dp” 段（高电平点亮对应段），通过段码表（如数字 “1” 对应段码 0x06，即 b、c 段点亮）输出对应电平；**
  + **位选：4 个 GPIO 引脚控制哪一位数码管点亮（高电平选中对应位），通过动态扫描（快速切换位选引脚，利用人眼视觉暂留）实现 4 位同时显示；**

**（3）联动逻辑：按下小键盘按键后，程序将按键数值转换为段码，通过动态扫描显示在数码管上；同时通过 USART 串口发送按键对应的flag值（如按 “1” 键发送flag=1），在调试助手接收框中同步显示。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，按实验箱上的小键盘，观看数码管上的显示 情况，以及串口调试助手接收框中的显示内容**

****

****

* **实验样例12 ADC模数转换**

**实验原理：**

**实验通过 STM32 的ADC（模数转换器）外设将传感器模拟信号转换为数字信号，原理如下：**

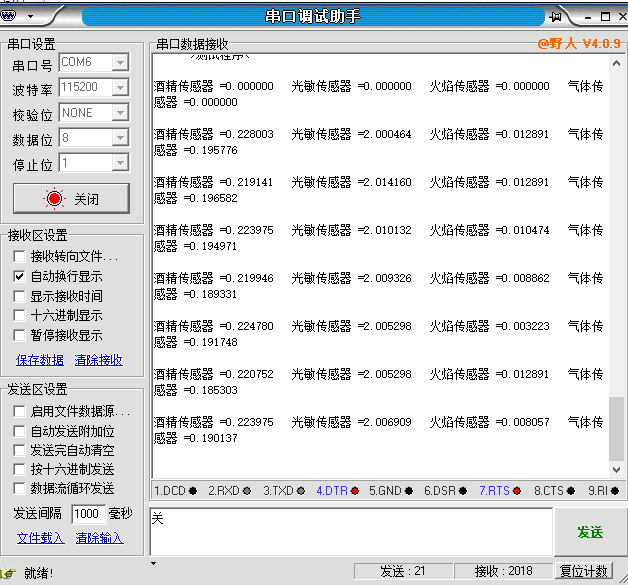
**（1）传感器信号：酒精、光敏、火焰、气体传感器输出的均为模拟电压信号（如光敏传感器光照越强，输出电压越高；酒精浓度越高，输出电压越高），这些信号连接到 STM32 的 ADC 输入引脚；**

**（2）ADC 配置：将 ADC 配置为 “多通道连续采集模式”，采样时间设为足够长（确保采样精度），数据对齐方式为 “右对齐”；ADC 时钟通过预分频器调整（不超过 ADC 最大允许时钟频率）；**

**（3）数据转换与传输：ADC 采集模拟信号后，将其转换为 12 位数字量（范围 0~4095），程序通过公式 “实际电压 =（数字量 / 4095）× 参考电压（如 3.3V）” 将数字量转换为传感器电压值，再通过 USART 串口将 4 个传感器的电压值实时发送到调试助手，故接收框中能观察到持续更新的传感器数据。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，观看串口调试助手接收框中的显示内容**

****

* **实验样例13 红外接收**

**实验原理：**

**实验基于NEC 红外编码协议实现红外信号解析，核心是红外接收与编码识别：**

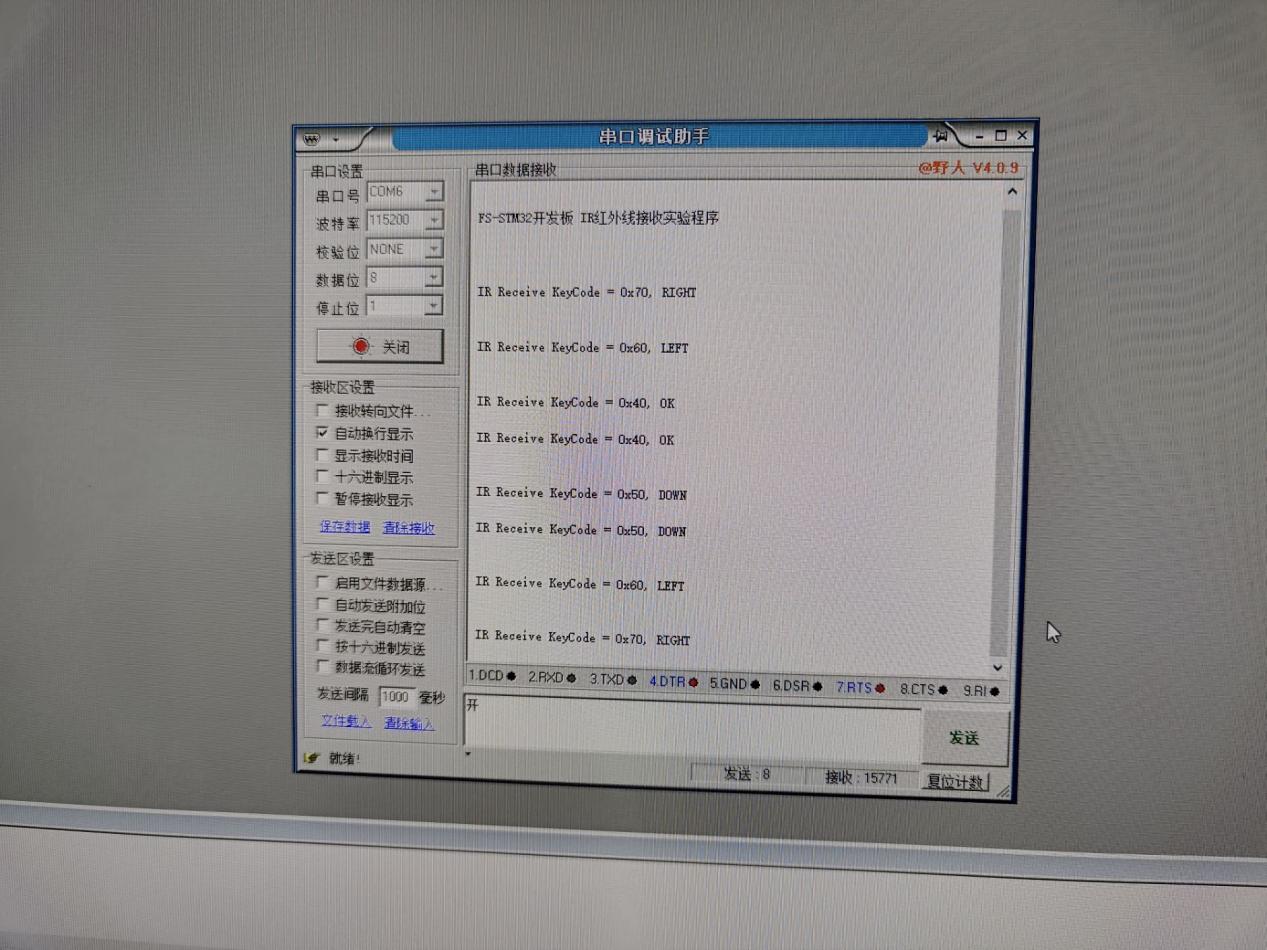
**（1）红外信号传输：红外遥控器按下按键时，会发送调制后的 NEC 编码信号（载波频率 38kHz），红外接收模块（如 HS0038）接收信号后，解调为 TTL 电平信号，输出到 STM32 的 GPIO 引脚；**

**（2）STM32 信号解析：程序将红外接收引脚配置为 “外部中断模式”（下降沿触发），通过中断捕获 NEC 编码的 “引导码”（9ms 高电平 + 4.5ms 低电平）、“地址码”（8 位）、“数据码”（8 位，含反码校验）；（3）按键映射：解析出 “数据码” 后，将其与预设的按键表匹配（如数据码 0x70 对应 “RIGHT” 键，0x60 对应 “LEFT” 键），再通过 USART 串口将 “KeyCode + 按键名称” 发送到调试助手，实现红外按键的可视化识别。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，按红外遥控器上的按键，观看串口调试助手 接收框中的显示内容**

****



* **实验样例14 温度采集**

**实验原理：**

**实验通过数字温度传感器（如 DS18B20）采集温度，STM32 读取传感器数据并转换为实际温度值，原理如下：**

**（1）温度传感器工作：DS18B20 通过单总线协议与 STM32 通信（仅需 1 个 GPIO 引脚），传感器内部通过热敏电阻将温度转换为数字量，精度可达 0.1℃；**

**（2）单总线通信流程：程序初始化单总线后，发送 “复位脉冲”（低电平 480~960μs），DS18B20 回复 “存在脉冲”（低电平 60~240μs）；随后发送 “读温度指令”（0xBE），读取传感器输出的 16 位温度数据（高 8 位为整数部分，低 8 位为小数部分）；**

**（3）温度转换：将 16 位数据按公式转换为实际温度（如数据 0x0550 对应 21.25℃），通过 USART 串口发送到调试助手；当手指按压传感器芯片时，传感器温度升高，转换后的温度值随之增大，故接收框中能观察到温度变化。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，将手指压到温度传感器的芯片上，观看串口 调试助手接收框中的显示内容**

****

* **实验样例15 光电开关**

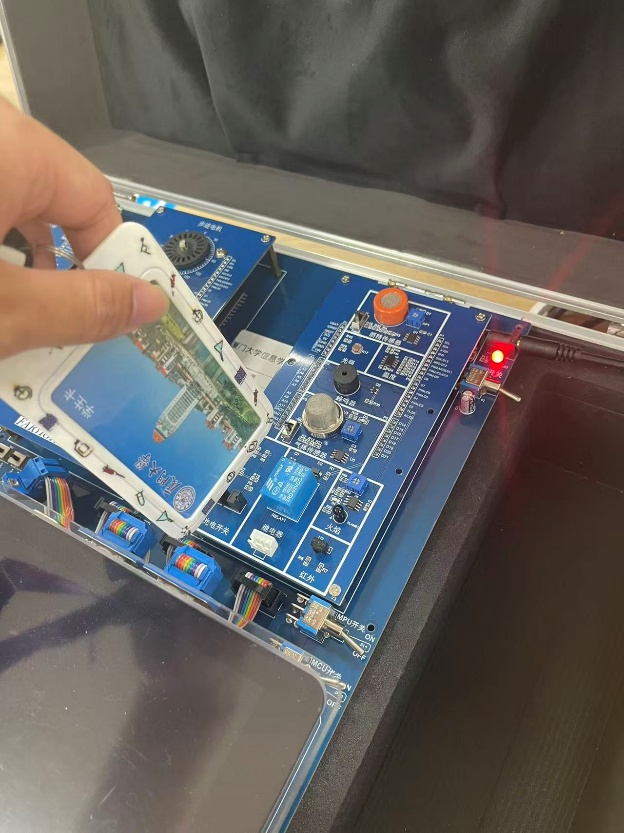
**实验原理：**

**实验基于对射式光电开关的遮挡检测原理，实现计数功能，原理如下：（1）光电开关工作：光电开关包含 “发射管”（发射红外光）和 “接收管”（接收红外光），正常无遮挡时，接收管接收到光线，输出高电平；当有物体（如折叠纸）放入槽口遮挡光线时，接收管无光照，输出低电平；**

**（2）GPIO 检测与计数：光电开关的输出端连接 STM32 的 GPIO 输入引脚，引脚配置为上拉输入模式；程序通过 “外部中断”（下降沿触发，即遮挡时电平从高变低触发中断）或 “循环查询” 检测电平变化；（3）计数逻辑：每检测到一次遮挡（电平变低），程序中的 “计数变量” 加 1，并通过 USART 串口将当前计数值（如 “计数值：1”“计数值：2”）发送到调试助手；移除遮挡后，电平恢复高电平，等待下一次遮挡触发计数，故能观察到 “放纸一次，计数加 1” 的效果。**

**实验结果：**

**按MCU开发板上的Reset键，运行程序；此时，用一张折叠的纸放到光电开关的槽口中，再 拿开，观看串口调试助手上的显示情况（每放1次，计数1次）**

****

****

* 1. **（课后完成）在自己的电脑上安装MDK软件（包括3个DFP包），并对MDK软件进行破解。**