# 嵌入式系统实验报告



|  |  |
| --- | --- |
| 实验名称： | GPIO与系统状态 |
| 姓 名： | 李志毅 |
| 学 号： | 2018211582 |
| 学 院(系)： | 计算机学院 |
| 专 业： | 网络工程 |
| 指导教师： | 刘健培 |

2021年 01月01日

# 实验目的

* 通过 FSM4 实验板了解实验的软硬件环境，熟悉 MDK 开发环境的使用。
* 学习查阅文档和数据手册，获取需要的信息。
* 学会使用 C 语言直接控制 IO 寄存器完成功能。
* 掌握基本的软件编写与调试方式。
* 学会 STM32 GPIO 的基本操作方式。

# 实验环境

* FS-STM32F407开发平台
* ST-Link 仿真器
* RealView MDK5.23集成开发软件
* PC机Window7/8/10 (32/64bit)
* 串口调试工具

# 实验要求

* 基本要求
* 编写程序控制 led 灯的亮灭（或者控制板上蜂鸣器的出声），输出以 字母、数字、空格组成的字符串的摩斯码（以“Hello Cortex-M4”为测试用例）。
* 特殊要求：不能使用 CMSIS 库函数操作 led 灯（蜂鸣器），需用代码 直接操作 GPIO的寄存器。
* 扩展要求
* 使用按键控制系统状态，LED灯显示系统状态：
* 按键 K3 按下：待机，系统进入低功耗模拟
* 按键 K4 长按：系统复位
* 按键 K5 双击：led 灯闪烁
* 按键 K6 长按：随着按动时长，4 个 led 灯依次点亮

# 实验原理

**4.1. STM32 GPIO 的配置**

LED灯的亮灭、蜂鸣器的鸣响、按键电平的读入都需要使用 STM32 芯片的I/O 引脚。

STM32 芯片上，I/O 引脚可以被软件设置成各种不同的功能，如输入或输出，所以被称为 GPIO (General-purpose I/O)。而 GPIO 引脚又被分为 GPIOA、 GPIOB…GPIOG不同的组，每组端口分为 0~15，共 16 个不同的引脚。

具体如何使用，需要参考板子原理图、芯片的datasheet和reference mannual。

**首先需要从原理图找到板上 LED、按键对应的GPIO。**

在实验板使用 D6 查找地底板原理图，可得：

D6——PF7

D7——PF8

D8——PF9

D9——PF10

控制 GPIO的高低电平来控制 LED的状态。0—亮，1—灭。

使用蜂鸣器是使用PF6控制。1-响，0-灭

4个按键，0-按下，1-断开

K3——PI9

K4——PF11

K5——PC13

K6——PA0

**程序如何控制GPIO？**

包括初始化和主逻辑2部分。

STM32 GPIO的初始化需要 2 步：

1. 开启 GPIO时钟

2. 配置好 GPIO的寄存器

如果是复用 GPIO管脚的外设，则一般需要 4 步：

1. 开启复用管脚的 GPIO 时钟

2. 配置好复用管脚的 GPIO寄存器

3. 开启外设时钟

4. 配置好外设的寄存器

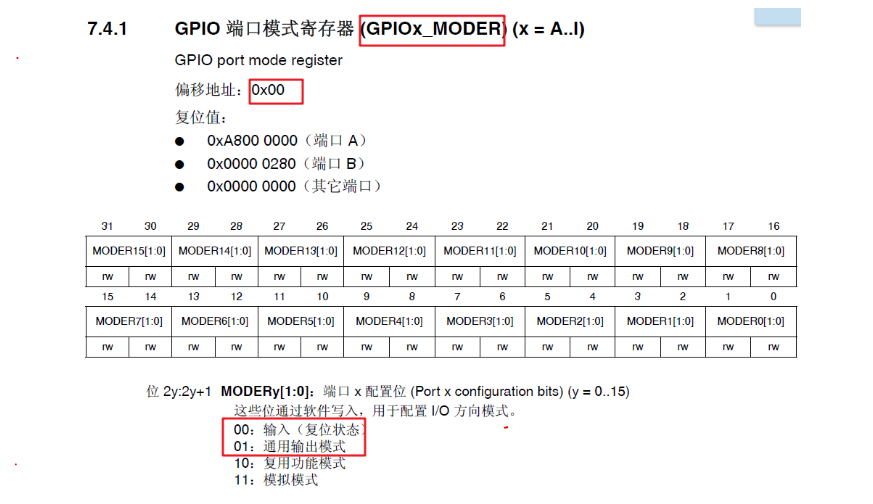
主逻辑部分主要是完成程序功能，如本实验的摩斯码编码输出。

一般主逻辑可分为 2 部分：

1. 与底层外设无关的逻辑部分。如摩斯码的编解码。

2. 需要通过外设与外部交互的驱动部分。如果摩斯码的滴答到 LED/蜂鸣器的映射。

本次实验主要使用到的寄存器为：



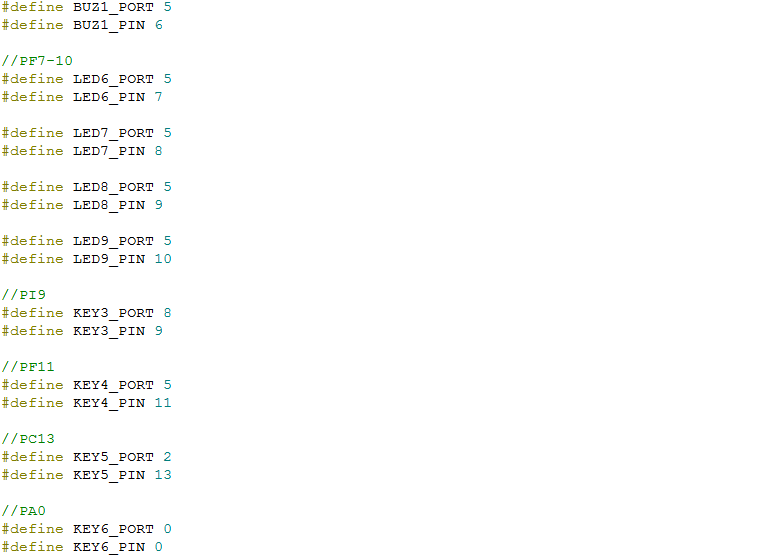
摩尔斯码对应表为

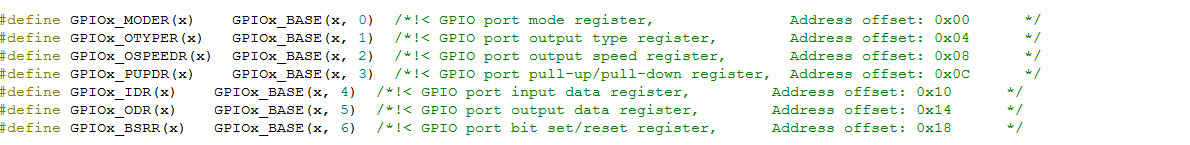


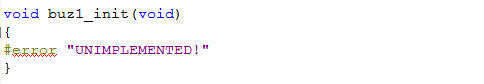
# 实验步骤

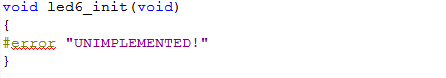
**基础实验：**

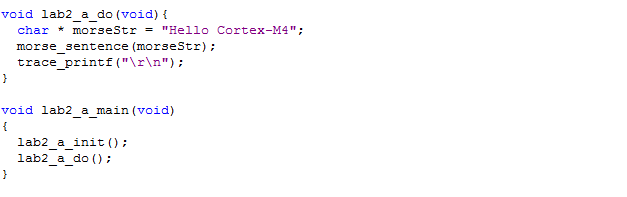
在老师给定的示例代码exp2\_a\_example的基础上进行增加代码，以此控制LED/蜂鸣器进行输出。





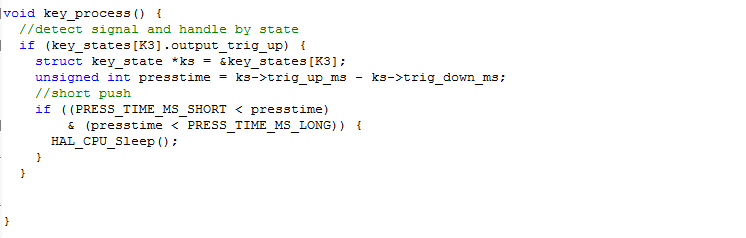


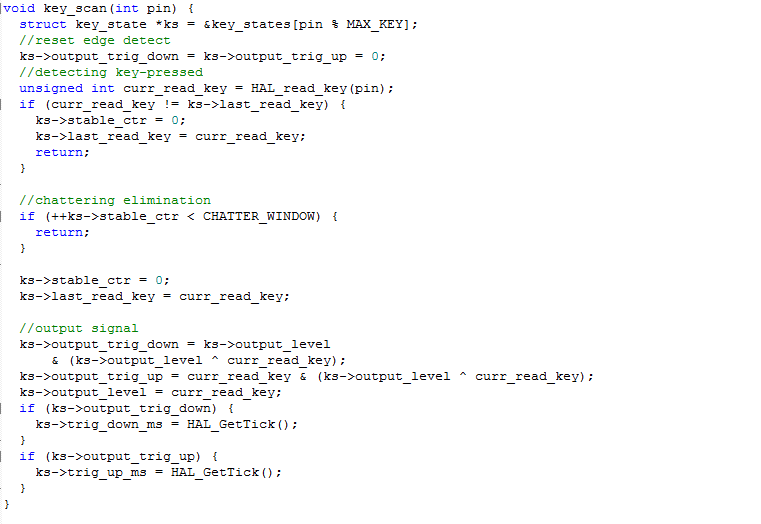




**拓展实验：**

在老师给定的示例代码exp2\_b\_example的基础上编写按键读取和消除抖动的代码，控制LED灯的亮灭。





# 实验方案与实现

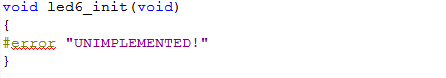
## 软件结构

首先是**基础实验**，按以下步骤进行实验

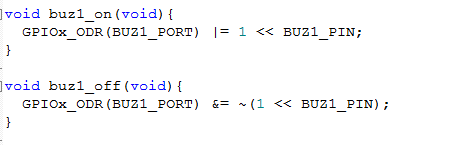


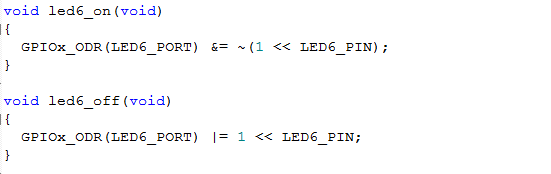
对于老师给定的代码，我们在其基础上进行修改





首先根据老师所给实例代码中对于buz1和led6的开启和关闭代码可以看出

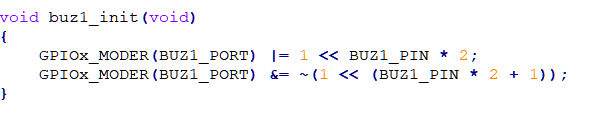


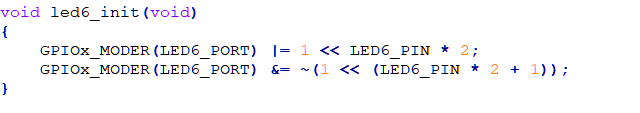


对一个寄存器中某一位或其余各位进行赋值可以通过&=和|=的方式，分析题目要求我们做的，需要对buz1蜂鸣器和led6闪光灯进行初始化，在两个初始化部分我们需要做的事情是，修改GPIO端口模式寄存器GPIOx\_MODER的值，将其设置为通用输出模式。

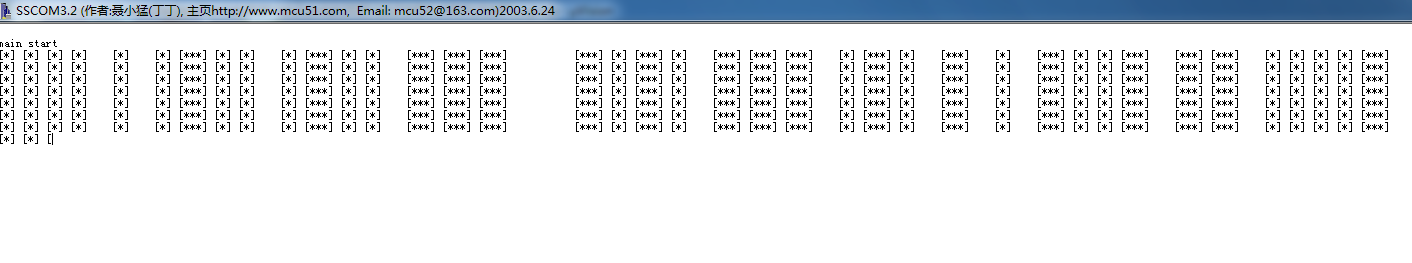
因此根据实验原理，我们需要将第2y位和第2y+1位分别设置成0和1

**如下图**：

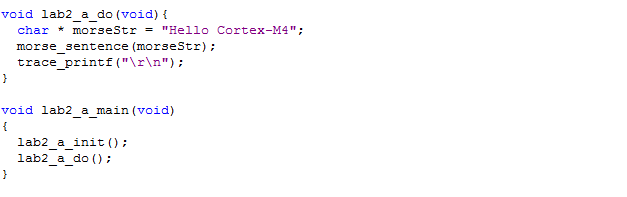




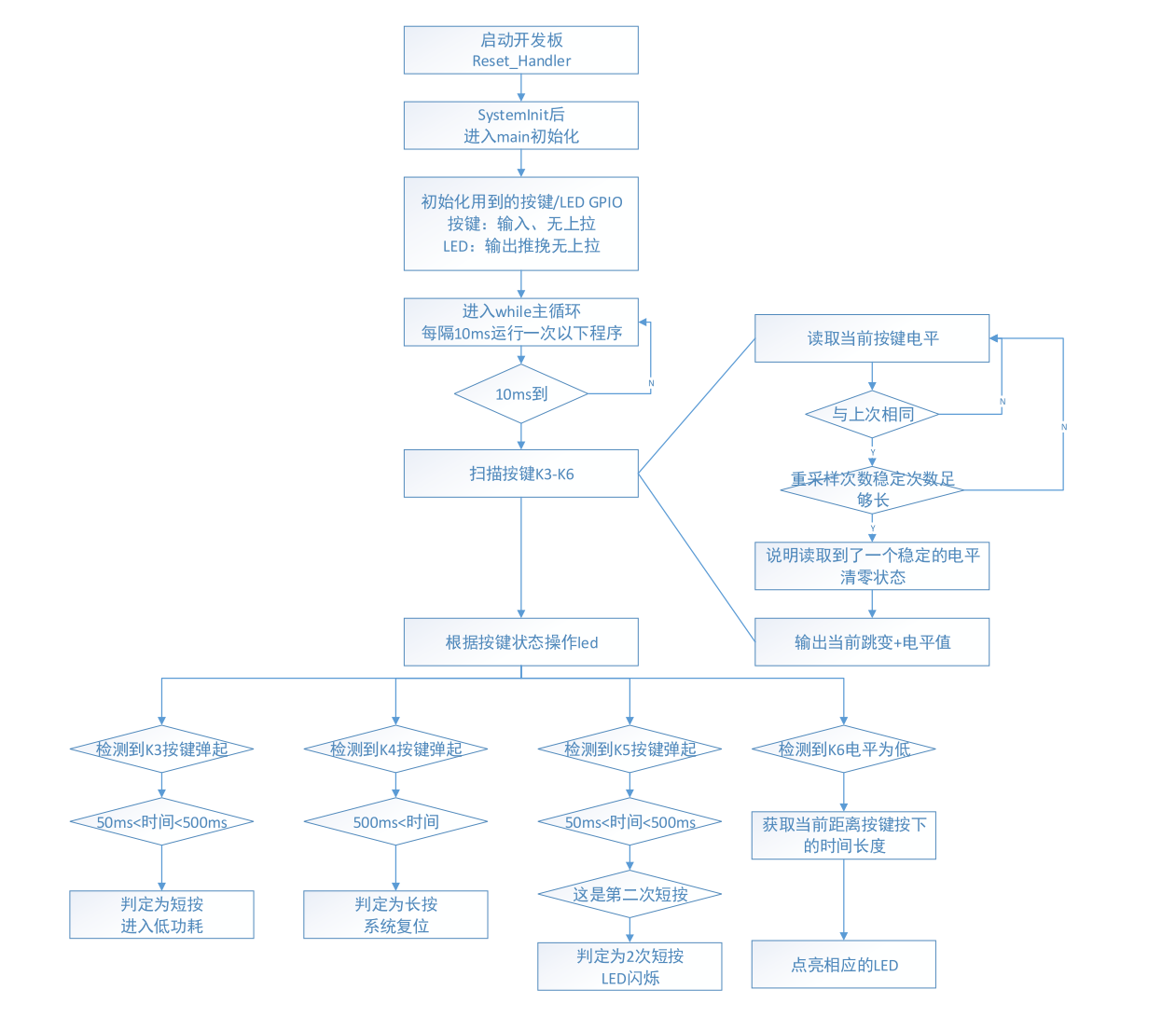
由此我们运行可以得到输出结果展示：



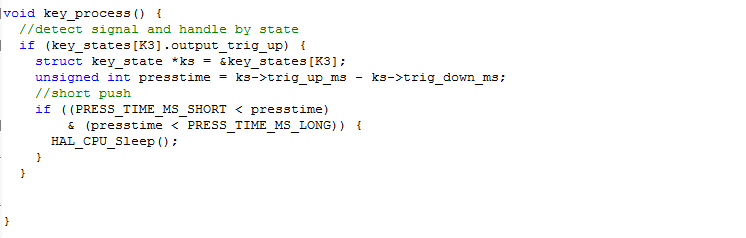
其摩斯码输出的结果翻译后即为我们在预先设计好的“Hello Cortex-M4”



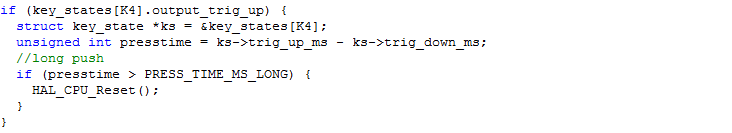
其次是**拓展实验，**拓展实验的实验流程为：



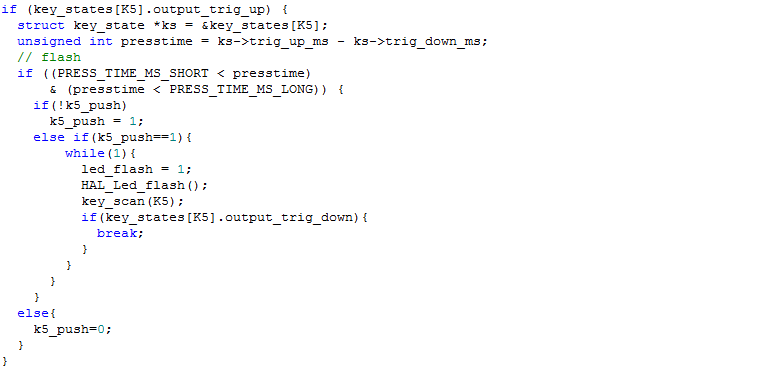
需要根据不同按键的状态对灯的亮灭状态进行操作，示例代码中已经给出对于K3按键弹起时执行的操作：



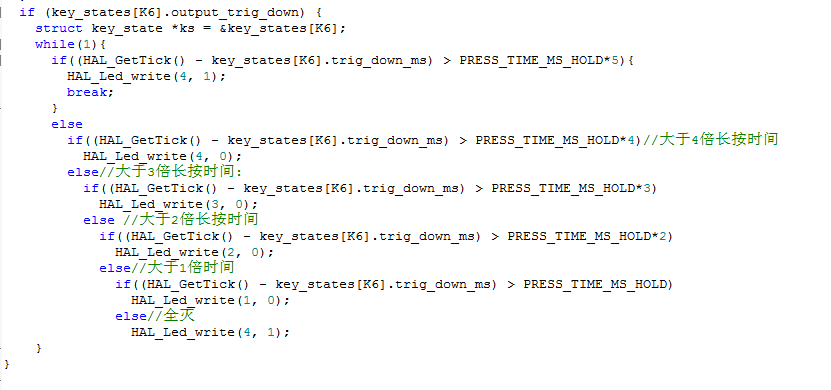
* **K3:**检测到K3按键弹起时，通过ks->trig\_up\_ms减去ks->trig\_down\_ms得到按下到弹起的时间间隔，若时间间隔小于宏定义的PRESS\_TIME\_\_MS\_LONG，即短按K3，则说明K3按键对应的功能开始作用，即调用函数HAL\_CPU\_Sleep()将CPU置为低功耗模式。
* **K4:**检测到K4按键弹起时，通过ks->trig\_up\_ms减去ks->trig\_down\_ms得到按下到弹起的时间间隔，若时间间隔大于宏定义的PRESS\_TIME\_MS\_LONG，即长按K4，则说明要执行系统复位的功能，因此调用函数HAL\_CPU\_Reset()进行系统复位



* **K5:**检测到K5按键弹起时，通过ks->trig\_up\_ms减去ks->trig\_down\_ms得到按下到弹起的时间间隔，若时间间隔小于宏定义的PRESS\_TIME\_MS\_LONG，即短按K5，若检测到两次短按(通过k5\_push的值判定)，则开始执行闪烁操作，不断循环调用HAL\_Led\_flash()函数，直到检测到K5的再次按下，则跳出循环，进行原先的主循环检测。



* **K6:**K6需要根据K6按键按下的时间长度点亮不同的Led灯，当长于一倍时间时，点亮Led1，长于两倍时间时点亮Led2，长于三倍时间时点亮Led3，长于四倍时间时间时，点亮Led4，由此代码为：

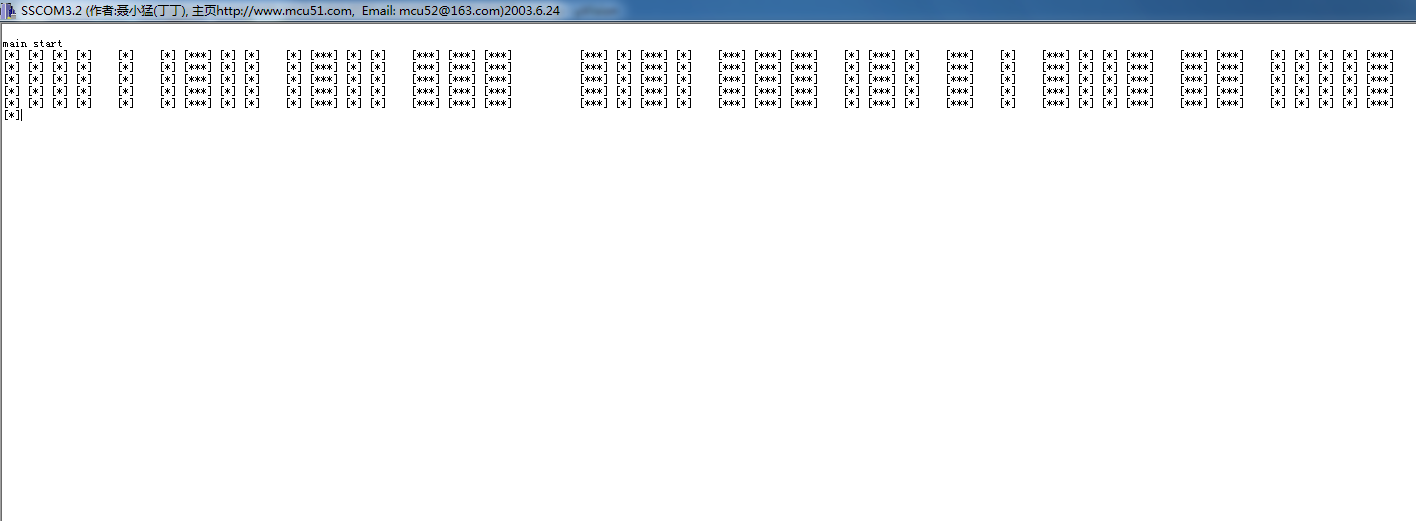


## 源代码

所有源码已在6.1软件结构步骤中给出

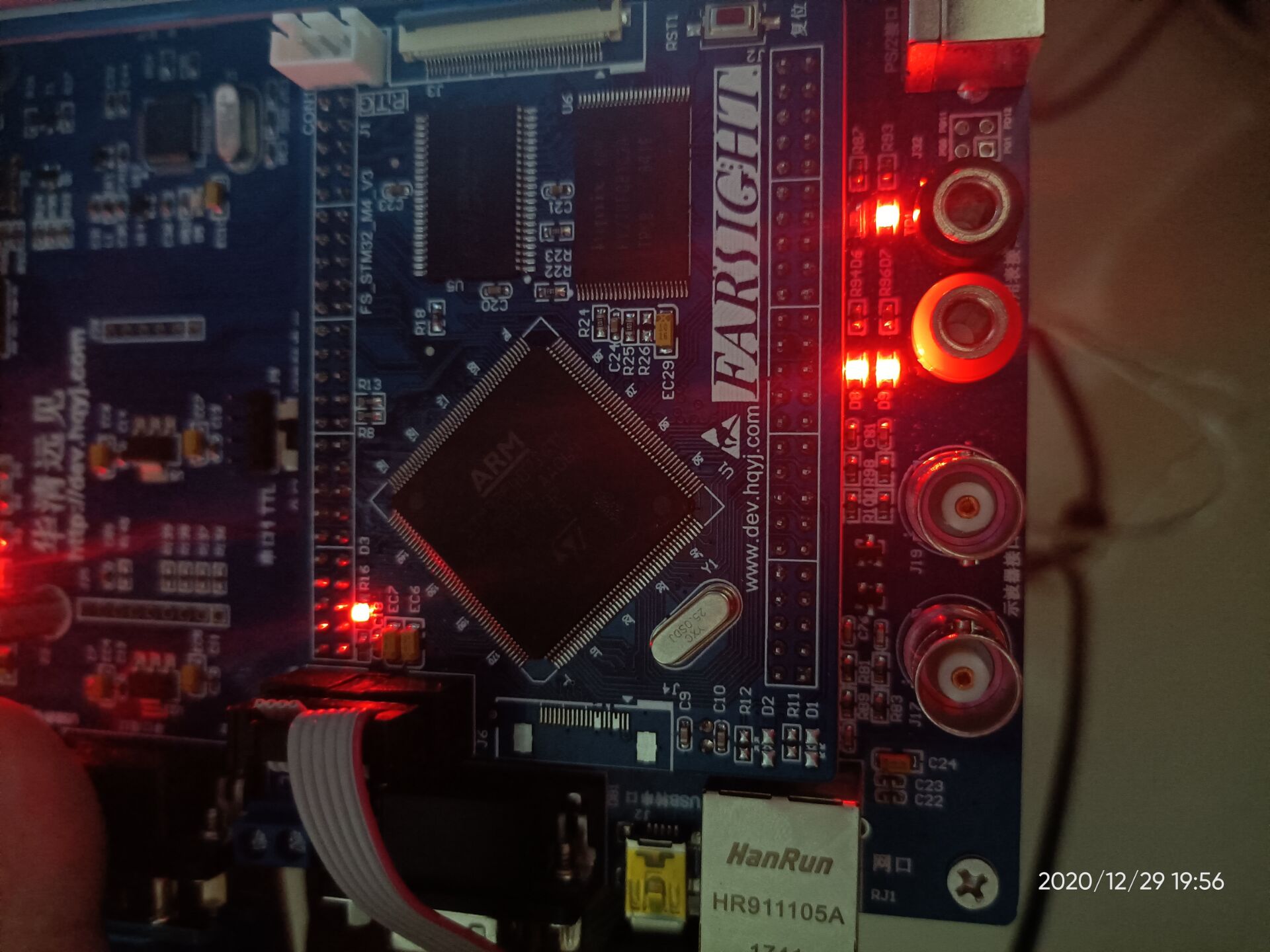
# 实验结果与分析

**基础实验：**

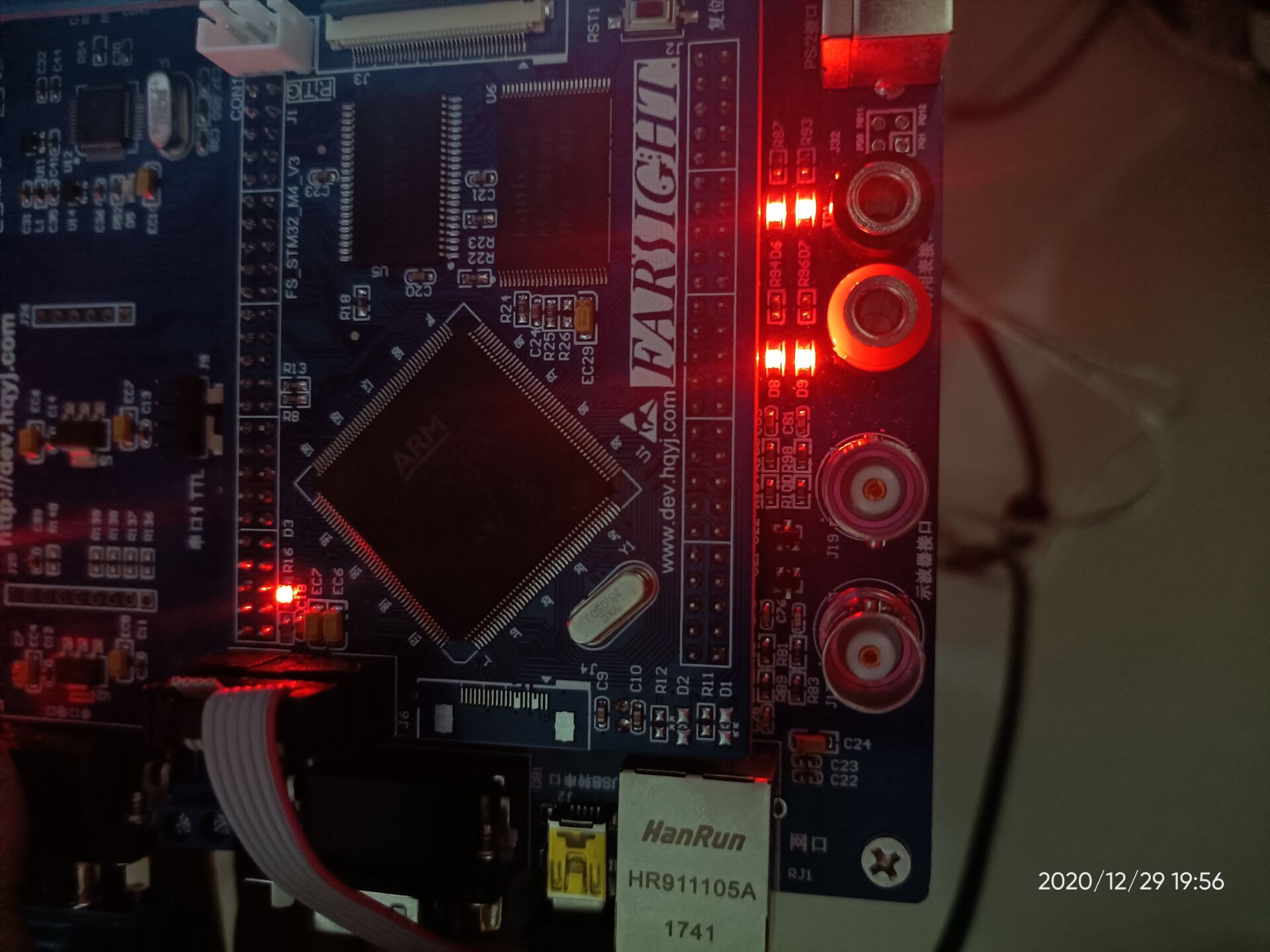


**拓展实验：**

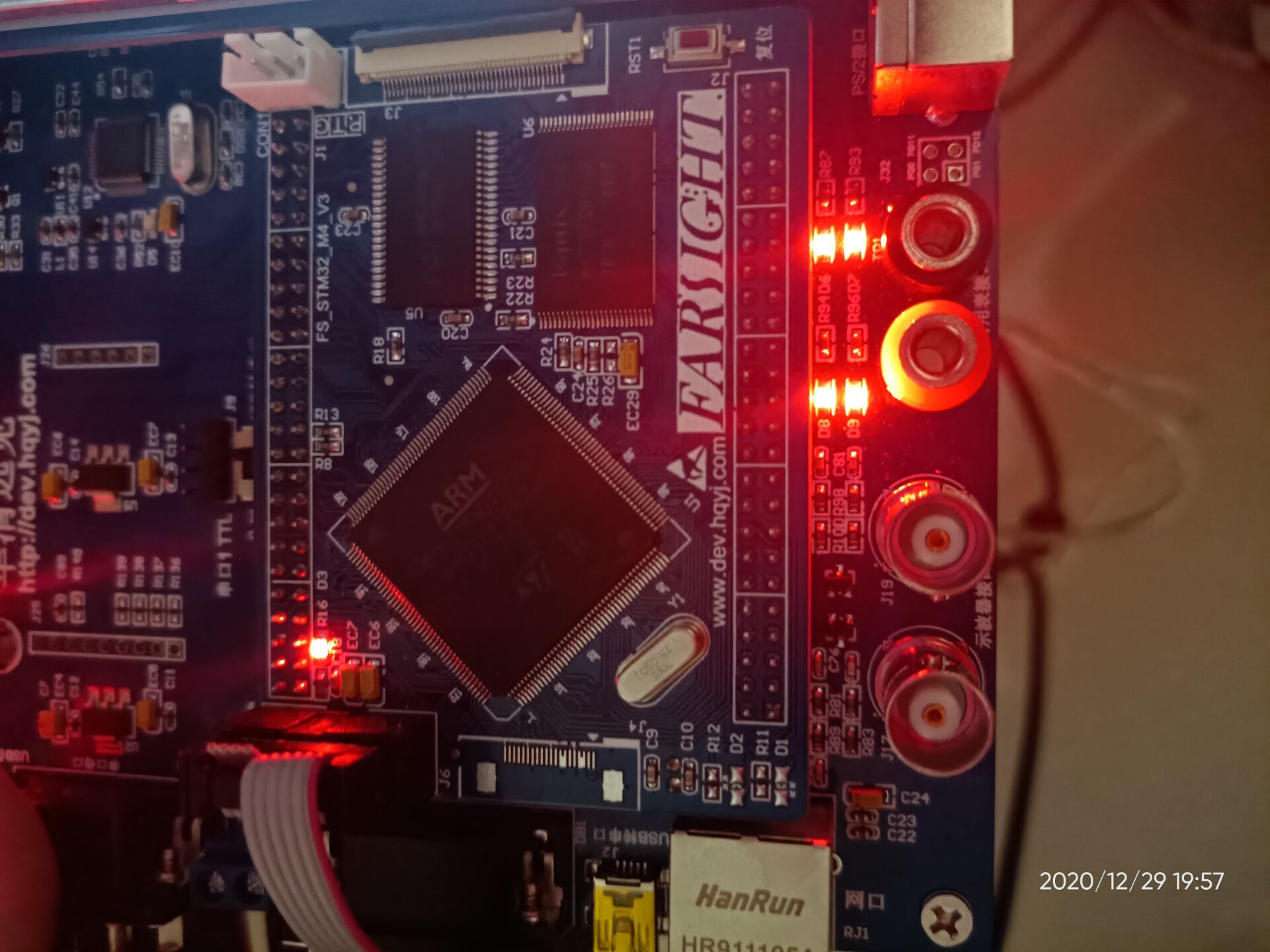
**K3：**

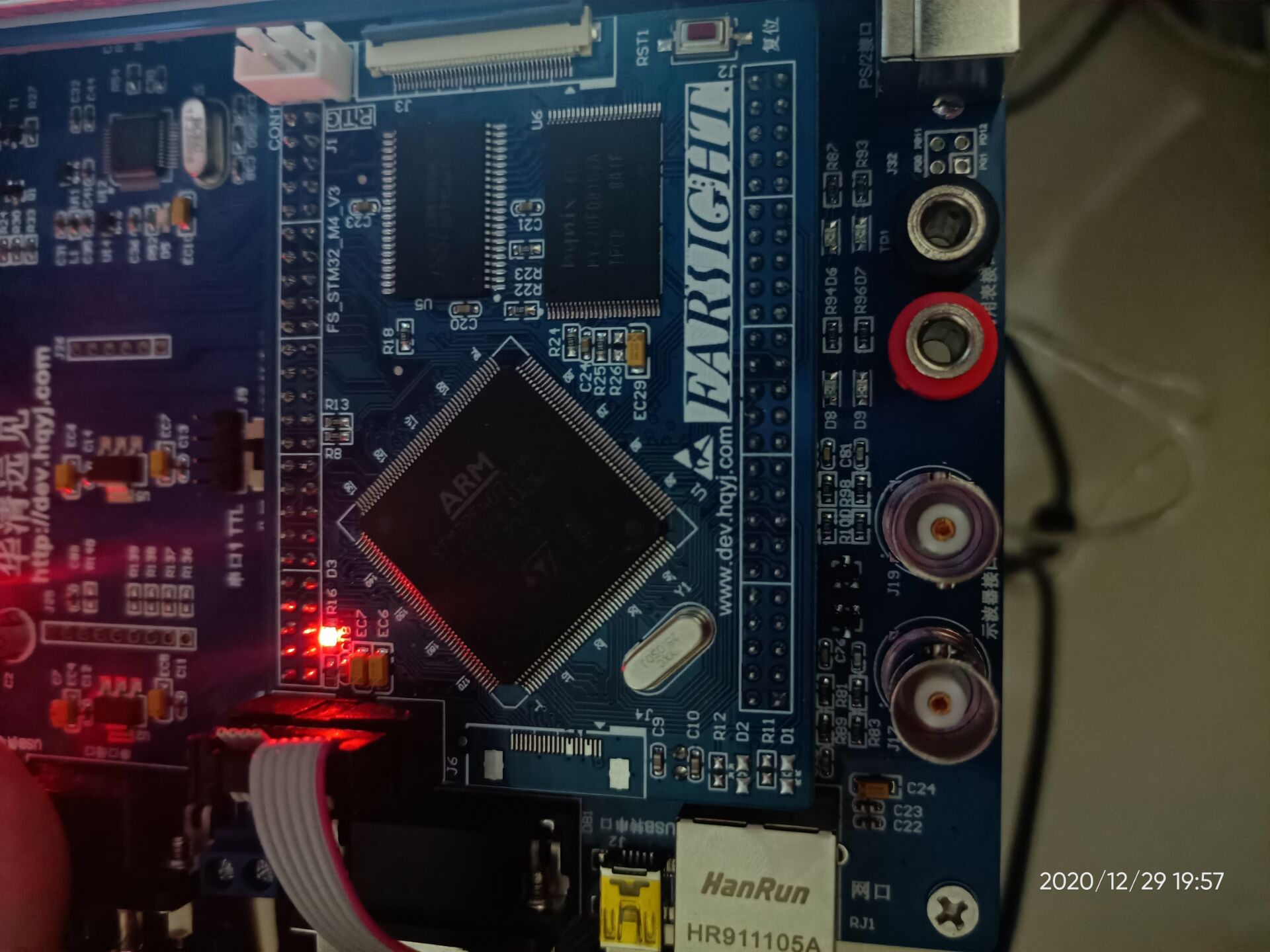


**K4：**

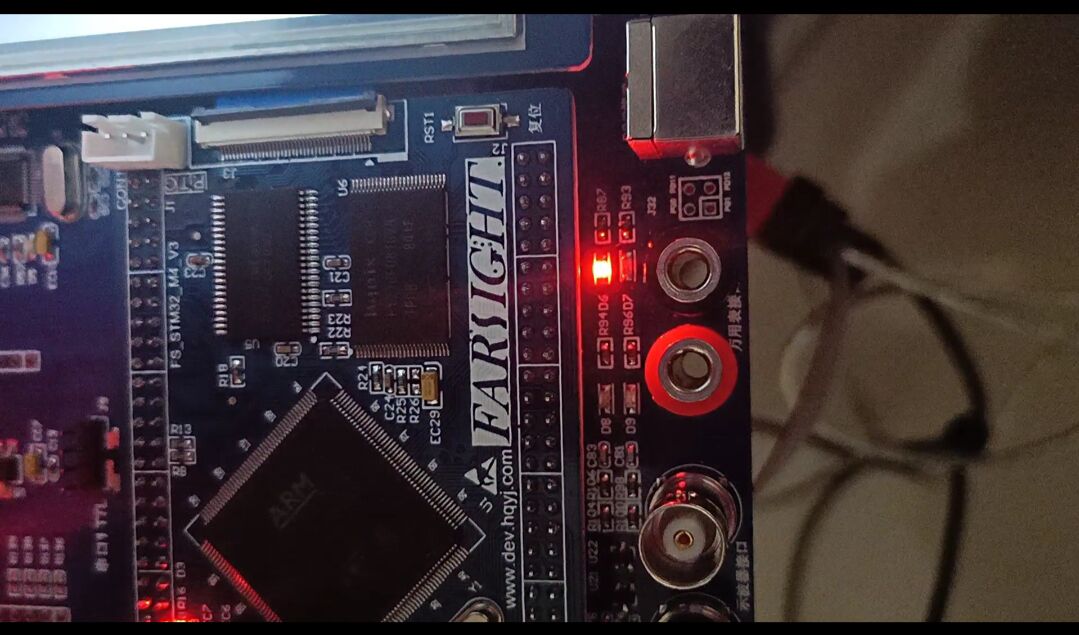


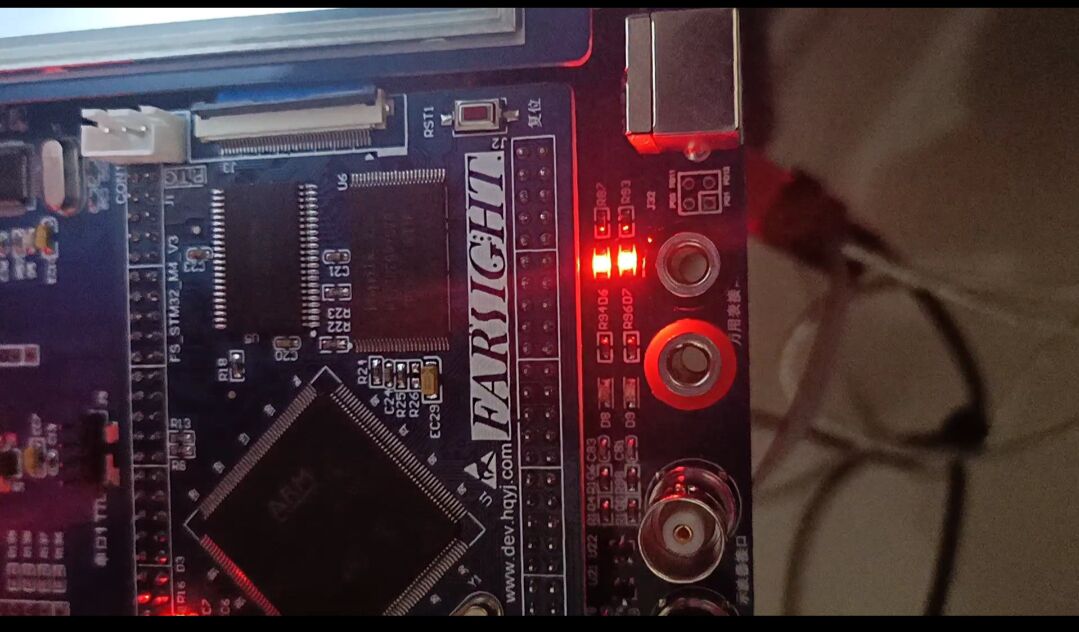
**K5：闪烁**

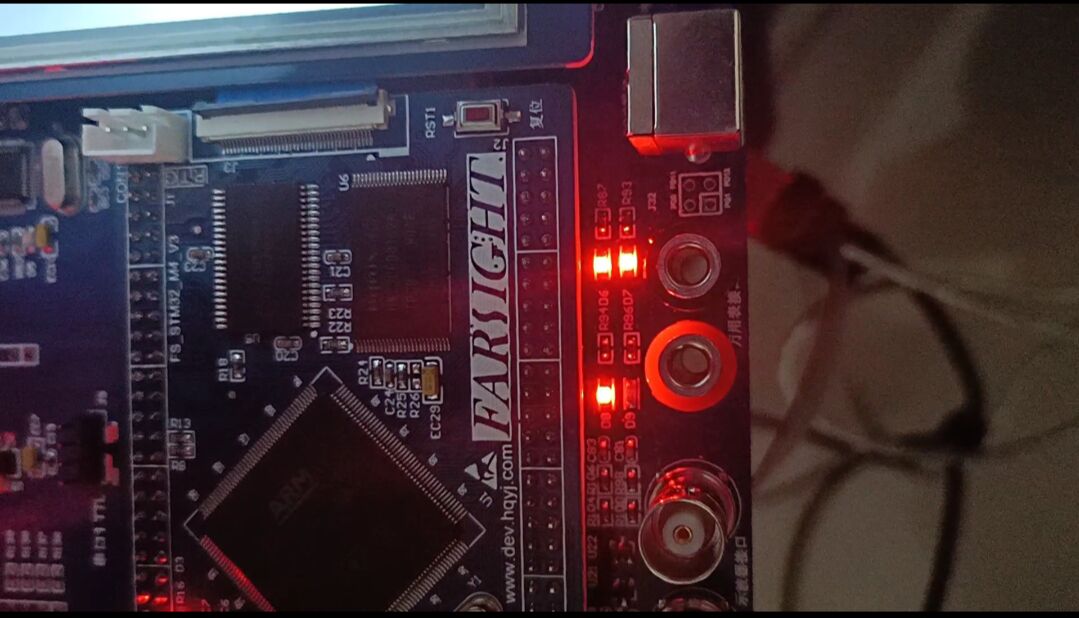


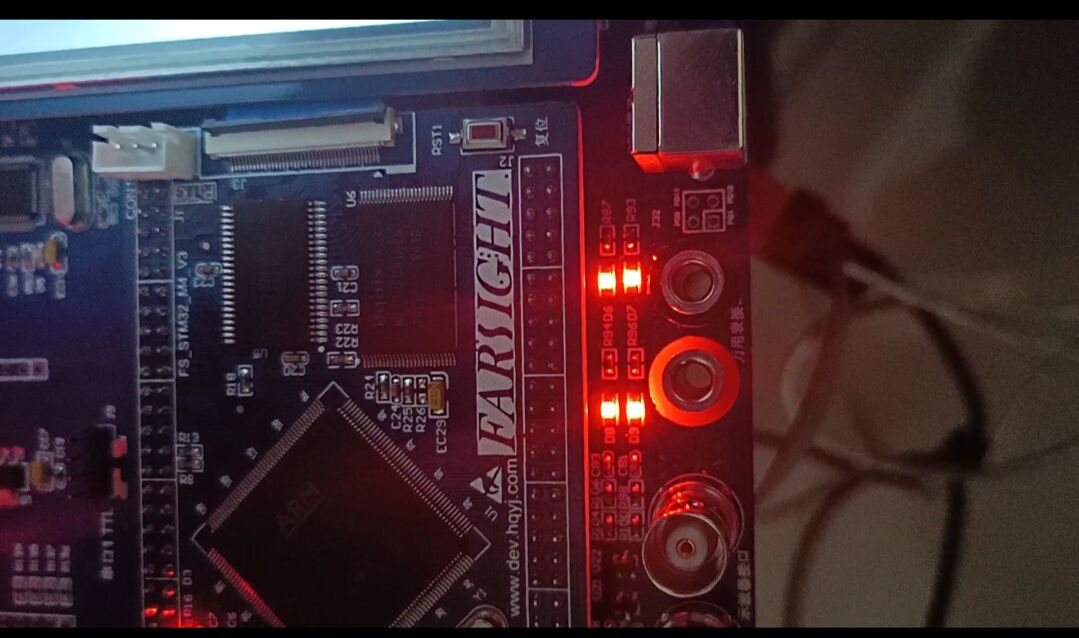


**K6:依次亮起**









# 实验总结

通过本次实验，我对于GPIO的基础操作方式有了更加细致的了解，对于STM32 GPIO的配置和寄存器的作用有了更加清晰的认识，从本次实验中我收获良多。