浙江工业大学

单片机课程设计
实验报告

姓　　名：凌智城

学　　院：信息工程学院

专业班级：通信工程 1803 班

学　　号：201806061211

教　　师：庄婵飞

组　　员：苏豪宇

提交日期：2021 年 7 月 15 日

任务书

一、 熟练掌握 Keil5-MDK 软件的安装及使用，通过学习 ALIENTEK 阿波罗 STM32F4 开发板的各个实验例程，掌握开发板硬件设计技术。

二、 学习和掌握 STemwin 的移植和开发技巧，在开发板的 LCD 触摸屏上，实现按键、动态显示窗口等设计。

三、 熟悉和掌握各类环境监控传感器的开发技巧，包括: DHT11、MQ2、MQ131、PMS5003、SGP30 等，了解数据采集模块的工作原理，实现温湿度、颗粒浓度、可燃气体浓度及有机有毒气体浓度的采集、存储及基于 WIFI 网络的无线传输。

四、 设计基于无线网络的环境参数评估系统，通过 STM32F4 开发板、数据采集板及无线路由器，实现环境监控的人机对话界面设计、环境参数接收、预警及动态显示，预警值可以动态设置。

目录

任务一　Keil5 环境搭建以及模板工程建立

一、　实验任务

1) 熟悉 STM32F429 开发板系统、Keil5、ST-Link 等软件的使用。
2) 通过学习 ALIENTEK 阿波罗 STM32F4 开发板的各个实验例程，掌握开发板硬件设计技术。

二、　实验工具

1) Windows 系统 PC 主机，预装 Keil5 MDK、ST-Link 等程序。
2) STM32F429 阿波罗开发板。

三、　实验过程和步骤

1)　使用已安装完毕的 Keil5 软件，Arm 版本的 Keil 与 C51 版本的 Keil 有所不同，但大部分是一样的。之前安装过 C51 版本的 Keil5，ARM 版本的安装程序也大同小异，没有什么复杂的流程，只是需要额外安装 Arm 的包否则无法运行，可以在主程序打开后的 Pack Installer 中在线安装所需要的包，这次课程设计用的是 STM32F4xx_DFP 2.8.0 版本，所以需要安装相应的包，直接在 Pack Installer 中安装，或者可以到官网上下载对应的包然后安装到 Keil5 的目录下即可。

2)　在个人电脑上安装时，有可能会出现 Error: L6411E: No compatible library exists with a definition of startup symbol __main.，这是因为之前装过 ADS，是 ADS 与 MDK 冲突所致，可以通过在系统变量中添加环境变量解决，变量名：ARMCC5LIB；变量值 E:\Keil_v5\ARM\ARMCC\lib

3)　要将调试好的程序下载到开发板中，可以使用 ISP 串口，优点是占用的引脚少但只能下载程序不能单步调试。也可以使用 S-TLINK，优点是下载的程序可单步调试但占用引脚多。实验室电脑已经安装好驱动，若还想自己了解两种方法的驱动安装方法，请参考下面两个文件。

a)　ISP 串口驱动安装

打开安装资料包“驱动程序”，找到并打开文件夹“CH340 驱动”安装，如果安装过程中，提示：预安装成功且无法正常使用 USB 串口。那么将，请手动将

serenum.sys 和 serial.sys 这两个文件，拷贝到：

C:\Windows\System32\drivers 文件夹下。如果该文件夹下本来就有这两个文件，提示无法替换，那么请先删除这两个原有的文件，再拷贝过去即可。

b) S-TLINK 驱动安装

打开安装资料包“驱动程序”，找到并打开文件夹“STLINK 驱动”进行安装；安装 dpinst_amd64.exe 文件，如果安装之后没有提示报错，那就说明驱动安装成功。如果有报错，大家卸载了之后再安装 dpinst_x86.exe 文件即可（与个人电脑 CPU 是否是 arm 的有关）。

4) 准备 HAL 库开发包：STM32Cube_FW_F4_V1.11.0，可以从 ST 官网下载的 STM32CubeF4 包完整版。资料目录（压缩包）：“\8，STM32 参考资料\1，STM32CubeF4 固件包\en.stm32cubef4.zip”。

5) 创建一个新的文件夹以存放所有工程文件，为了方便存放工程需要的一些其他文件，新建下面 4 个子文件夹：CORE，HALLIB，OBJ 和 USER。在 USER 文件夹目录下新建工程，Project→New μVisionProject，即所有工程文件都保存到了 USER 目录下。

6) 出现选择设备 Device 界面，即选择设备型号，STMicroelectronics 下面的 STM32F429IGT6，选安装了相应的 Pack 才会显示这些型号。生成的.uvprojx 文件为工程文件，通过此文件打开整个工程。

7) 在开发包\STM32Cube_FW_F4_V1.10.0\Drivers\STM32F4xx_HAL_Driver 目录下将相应的文件复制到新建的工程文件下，并不需要全部复制，选取所需要的就可以了，全部复制会造成编译时将过长。

8) Keil5 需要手动将文件添加到工程中，打开 Manage Project Items，Project Targets 为整个工程，Groups 为子文件夹如 USER、CORE 等，Files 为子文件夹下的各个文件，将我们实际文件夹下的所有文件都添加进去如图 1.1。

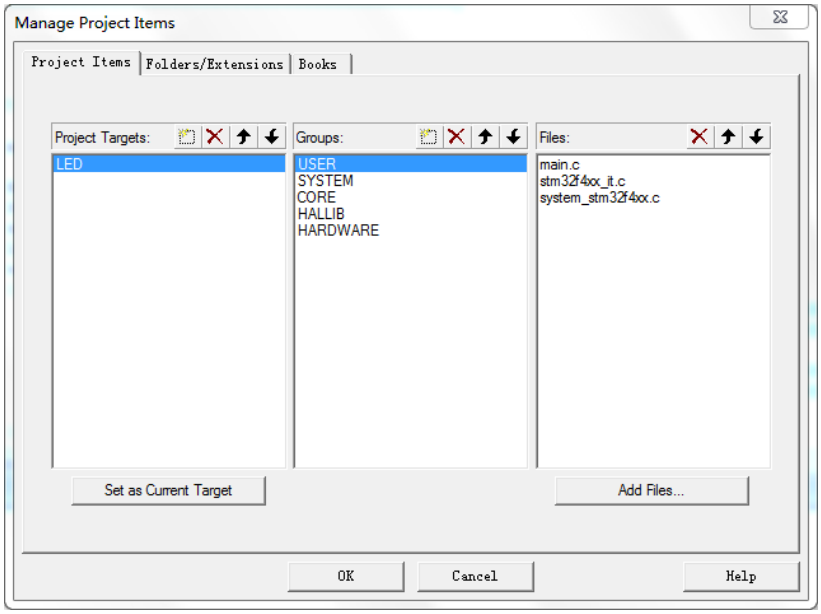


图 1.1 Manage Project Items 添加文件到工程

9) 仅仅添加到工程还不行，需要将 Group 下的子文件夹的路径添加到 MDK 的路径，告诉 MDK 那些目录下包含头文件，否则将无法识别包含头文件的程序，报错头文件路径找不到，如图 1.2，需要注意的是，添加的路径必须添加到头文件所在目录的最后一级。比如在 SYSTEM 文件夹下面有三个子文件夹下面都有.h 头文件，这些头文件在工程中都需要使用到，所以必须将这三个子目录都包含进来。这里需要添加的头文件路径包括：\CORE，\USER\，\SYSTEM\delay，\SYSTEM\usart，SYSTEM\sys 以及\HALLIB\Inc。

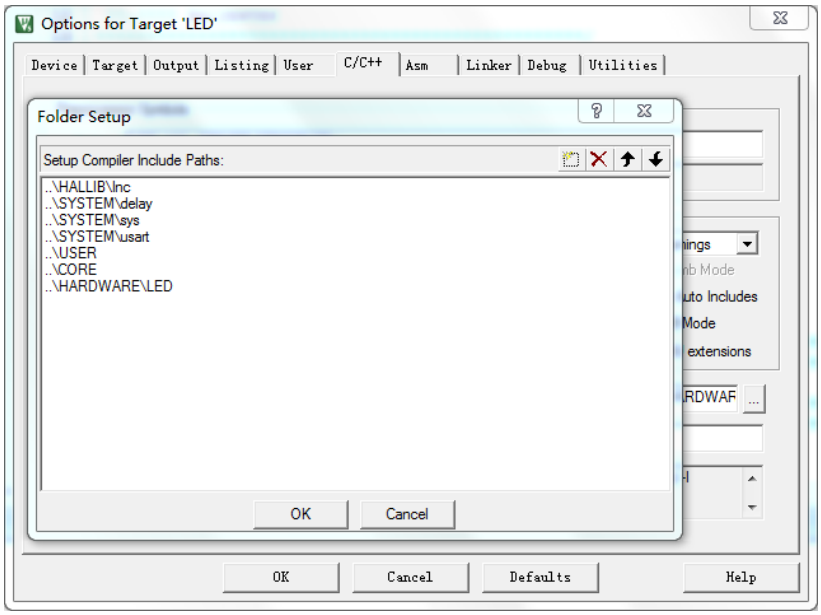


图 1.2 添加文件路径到工程

10) 在编译之前首先要选择编译中间文件编译后存放目录。MDK 默认编译后的中间文

件存放目录为 USER 目录下面的 Listings 和 Objects 子目录，这里为了和 ALIENTEK 工程结构保持一致，重新选择存放到目录 OBJ 目录之下。操作方法是点击魔术棒 ，然后选择“Output”选项下面的“Selectfolderforobjects…”,然后选择目录为上面新建的 OBJ 目录，然后依次点击 OK 即可。

11）选择 OBJ 目录为中间文件存放目录选择完 OBJ 目录为编译中间文件存放目录之后，点击 OK 回到 Output 选项卡。这里我们还要勾上“CreateHEXFile”选项和 BrowseInformation 选项。CreateHEXFile 选项选上是要求编译之后生成 HEX 文件。而 BrowseInformation 选项选上是方便查看工程中的一些函数变量定义等。

12）main.c

```
#include "sys.h"
#include "delay.h"
#include "usart.h"
#include "led.h"
int main(void)
{
   HAL_Init();
   Stm32_Clock_Init(360,25,2,8);
   delay_init(180);
   LED_Init();
   while(1)
   {
      HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET);
      HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_SET);
      delay_ms(500);
      HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_SET);
      HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_RESET);
      delay_ms(500);
   }
}
```

13）点击编译工程按钮编译整个工程即可。

四、 实验总结

1) 这一部分实验主要在介绍如何使用 Arm 版本的 Keil5 软件，几个最重要并且容易出问题的就是，驱动适配，可能会出现意想不到的问题，但给的参考资料都有详细的介绍，基本都可以自行解决，若果用实验室的机器则不会出现问题，大部分都已经配置好。
2) Keil 和 ADS 会产生冲突，添加系统环境变量即可，之前有使用 C51 版本的 Keil5 经验，所以使用起来也不算困难，库函数比较多，理解起来需要花费很多时间，认为只需要慢慢理解当前需要用到的就可以，出问题了再随着问题一步步找到源头。

任务二　跑马灯实验

一、　实验任务

1) 了解 STM32F429 的 IO 口作为输出使用的方法。
2) 通过代码控制 ALIENTEK 阿波罗 STM32 开发板上的两个 LED 灯 DS0 和 DS1 交替闪烁，实现类似跑马灯的效果。

二、　实验工具

1) Windows 系统 PC 主机，预装 Keil5 MDK、ST-Link 等程序。
2) STM32F429 阿波罗开发板。

三、　实验过程和步骤

1) 使用任务一的工程文件，将编译好的程序下载到硬件上。
   a) ①组 HALLIB 下面存放的是 ST 官方提供的 HAL 库文件，每一个源文件 stm32f4xx_hal_ppp.c 都对应一个头文件 stm32f4xx_hal_ppp.h。分组内的源文件可以根据工程需要添加和删除。这里对于跑马灯实验，需要添加 11 个源文件。
   b) ②组 CORE 下面存放的是固件库必须的核心头文件和启动文件。这里面的文件用户不需要修改。
   c) ③组 SYSTEM 是 ALIENTEK 提供的共用代码。
   d) ④组 HARDWARE 下面存放的是每个实验的外设驱动代码，通过调用 HALLIB 下面的 HAL 库文件函数实现的，比如 led.c 中函数调用 stm32f4xx_hal_gpio.c 内定义的函数对 led 进行初始化。
   e) ⑤组 USER 下面存放的主要是用户代码。但是 system_stm32f4xx.c 文件用户不需要修改，同时 stm32f4xx_it.c 里面存放的是中断服务函数，这两个文件的作用在 3.3 节有讲解。main.c 函数主要存放的是主函数。

2) STM32F4 每组通用 I/O 端口包括 4 个 32 位配置寄存器（MODER、OTYPER、OSPEEDR 和 PUPDR）、2 个 32 位数据寄存器（IDR 和 ODR）、1 个 32 位置位/复位寄存器(BSRR)、1 个 32 位锁定寄存器(LCKR)和 2 个 32 位复用功能选择寄存器（AFRH 和 AFRL）等。这样，STM32F4 每组 IO 有 10 个 32 位寄存器控制，其中常用的有 4 个配置寄存器+2 个数据寄存器+2 个复用功能选择寄存器，共 8 个，如果在使用的时候，每次都直接操作寄存器配置 IO，代码会比较多，也不容易记住。

3) HAL 库操作该寄存器读取 IO 输入数据相关函数：
   GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx,uint16_t GPIP_Pin);
4) 该函数用来读取一组 IO 下一个或者多个 IO 口电平状态。比如读取 GPIOF.5 的输入电平，方法为：
   HAL_GPIO_ReadPin(GPIOF,GPIO_PIN_5),该函数返回值就是 IO 口电平状态。
5) IO 操作步骤:
   a) 使能 IO 口时钟，调用函数为 HAL_RCC_GPIOX_CLK_ENABLE(其中 X=A~K)。

b) 初始化 IO 参数。调用函数 HAL_GPIO_Init();

c) 操作 IO 输入输出。操作 IO 的方法就是上面我们讲解的方法。

6) 相关代码注释，LED 低电平亮高，电平灭，通过 for 循环和延时来实现循环点亮。

```
{
    HAL_Init();    // 通过调用函数 HAL_GPIO_Init 实现配置 PB0 和 PB1 为推挽输出
    Stm32_Clock_Init(360,25,2,8);
    delay_init(180);
    LED_Init();    // 完成对 PB0 和 PB1 的初始化配置
    while(1)
    {
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET); // PB1=0 即 LED0=0，亮
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_SET);   // PB0=1 即 LED1=1，灭
        delay_ms(500);                                      // 延时 500ms
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_RESET);
        delay_ms(500);
    }
```

7) 下载验证

a) 使用 Flumcu 下载

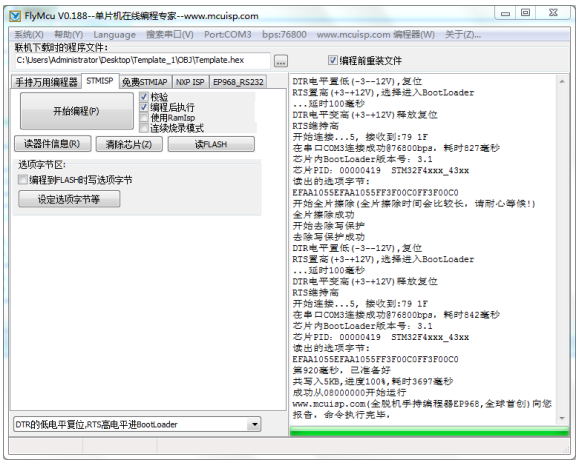


图 2.1 使用 Flumcu 下载

b) 使用 ST-LINK 下载

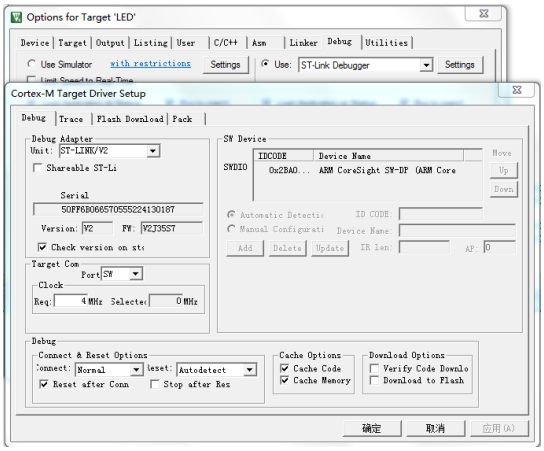


图 2.2 使用 ST-LINK 下载

四、 实验总结

跑马灯实验在第一次实验的基础上下载到 STM32 板子，两种下载方式在正确安装驱动后均可用；区别在于 Flymcu 下载使用端口少，在程序小的情况下可以依靠 USB 连接直接供电，但是烧写时间长；ST-LINK 下载的设置较为复杂，并且部分组出现了无法使用的状况，猜测可能是驱动版本或者电脑上其他软件的冲突，因为我们组两种下载方式用的比较顺利，所以没有深入研究，但应该查阅相关资料能够找到解决办法，ST-LINK 的烧写时间大大缩短，特别是程序较大时，在五百 Kb 左右用 Flymcu 需要数分钟，ST-LINK 在一分钟内完成；另外，有一点需要注意，使用 Flymcu 下载时千万不能选择“编程到 FLASH 时写选项字节”，选择了会出现连接芯片超市，一定要仔细阅读芯片参考手册，一旦错误设置即使取消选择也不可逆，导致芯片被锁，无法连接上，需要用另外的方式来解除锁定。

任务三　修改跑马灯实验

一、　实验任务

1) 熟悉 STM32F429 开发板系统、Keil5、ST-Link 等软件的使用。

2) 修改跑马灯实验，资料：参考标准例程-HAL 库版本实验 1

修改 1：为四种状态:01,10,00,11　0-亮，1-灭

修改 2：5 种状态：01，10，00，11，{0，1-0-1-0 即 LED0 闪烁}

二、　实验工具

1) Windows 系统 PC 主机，预装 Keil5 MDK、ST-Link 等程序。

2) STM32F429 阿波罗开发板。

三、　实验过程和步骤

1) 修改代码和注释

```
int main(void)
{
   HAL_Init();
   Stm32_Clock_Init(360,25,2,8);
   delay_init(180);
   LED_Init();
   while(1)
   {
      HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET);
      //LED0 对应引脚 PB1 拉低，亮，等同于 LED0(0)
      HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_SET);
      //LED1 对应引脚 PB0 拉高，灭，等同于 LED1(1)
      delay_ms(500);
      //延时 500ms，LED 显示亮灭，即 01
      HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_SET);
```

```
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_RESET);
        delay_ms(500);
        //延时 500ms，LED 显示灭亮，即 10
       HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_RESET);
        delay_ms(500);
        //延时 500ms，LED 显示亮亮，即 00
       HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_SET);
        delay_ms(500);
        //延时 500ms，LED 显示灭灭，即 11
        //下面是第五种状态，分成四部分，每部分延时 500/4=125ms
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_SET);
        delay_ms(125);
       HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_RESET);
        delay_ms(125);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_SET);
        delay_ms(125);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,GPIO_PIN_RESET);
        delay_ms(125);
    }
}
```

四、 实验总结

修改比较简单，就是增加几个不同的状态和使用不同的延时，但实际上涉及到的 I/O 电平以及初始化设置等内置函数比较复杂。

任务四　按键实验

一、　实验任务

1) 熟悉 STM32F429 开发板系统、Keil5、ST-Link 等软件的使用。
2) 参考标准例程，实现按键 key_up 同时启动蜂鸣器。

二、　实验工具

1) Windows 系统 PC 主机，预装 Keil5 MDK、ST-Link 等程序。
2) STM32F429 阿波罗开发板。

三、　实验过程和步骤

1) 修改代码和注释

```
int main(void)
{
    u8 key;
    u8 beepsta=1;                    // 蜂鸣器初始状态高电平，不叫
    HAL_Init();                      // 初始化 HAL 库
    Stm32_Clock_Init(360,25,2,8);    // 设置时钟,180Mhz
    delay_init(180);                 // 初始化延时函数
    uart_init(115200);               // 初始化 USART
    LED_Init();                      // 初始化 LED
    KEY_Init();                      // 初始化按键
    SDRAM_Init();                    // 初始化 SDRAM
    PCF8574_Init();                  // 一定要初始化 PCF8574 否则蜂鸣器不会叫

    while(1)
    {
        key=KEY_Scan(0);             //按键扫描
                    switch(key)
```

```
                {
                    case  WKUP_PRES:   //控制 LED0,LED1 互斥点亮
                                beepsta=!beepsta;                //蜂鸣器状态取反
                                PCF8574_WriteBit(BEEP_IO,beepsta);//控制蜂鸣器
                                LED1=!LED1;
                                LED0=!LED1;
                                break;
                    case  KEY2_PRES:   //控制 LED0 翻转.left
                                LED0=!LED0;
                                break;
                    case  KEY1_PRES:   //控制 LED1 翻转  mid
                                LED1=!LED1;
                                break;
                    case  KEY0_PRES:   //同时控制 LED0,LED1 翻转 right
                                LED0=!LED0;
                                LED1=!LED1;
                                break;
            }
       delay_ms(10);
   }
}
```

四、 实验总结

这个实验相对来说也比较简单，需要注意的是一定要使用 PCF8574_Init();函数向蜂鸣器写入初始化的值，否则蜂鸣器一直不会叫；同样的，在反转蜂鸣器操作 beepsta=!beepsta;后一定要使用 PCF8574_WriteBit(BEEP_IO,beepsta);函数重写，否则蜂鸣器的反转不会生效。

任务五　GUIbuilder 实验

一、　实验任务

1) 熟悉 STM32F429 开发板系统、Keil5、ST-Link 等软件的使用。

2) 学习和掌握 STemwin 的移植和开发技巧，在开发板的 LCD 触摸屏上，实现按键、动态显示窗口等设计。

二、　实验工具

1) Windows 系统 PC 主机，预装 Keil5 MDK、ST-Link 等程序。

2) STM32F429 阿波罗开发板。

三、　实验过程和步骤

1) 软件路径：阿波罗 STM32F429 资料盘(A 盘)→8，STM32 参考资料→1，STM32CubeF4 固件包→en.stm32cubef4.zip，解压后变成 STM32Cube_FW_F4_V1.11.0，在目录下找到 Middlewares→ST→STemWin→Software，如图 5.1 所示。

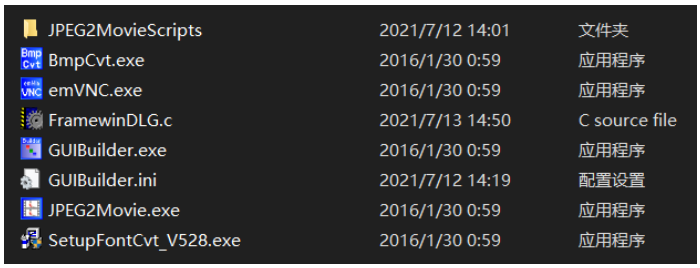


图 5.1 STemwin 文件路径

2) 依照参考视频所说，将上述文件拷贝到其他路径后也能打开，确实如此，但是操作后会发现，在 GUIBuilder 生成的.c 文件仍然会存到上图的文件路径中，所以其实将这个文件夹拷贝到其他路径并没有用。

3) GUIBuilder 界面如下图 5.2 所示，实验所用 LCD 显示屏分辨率 480*272。

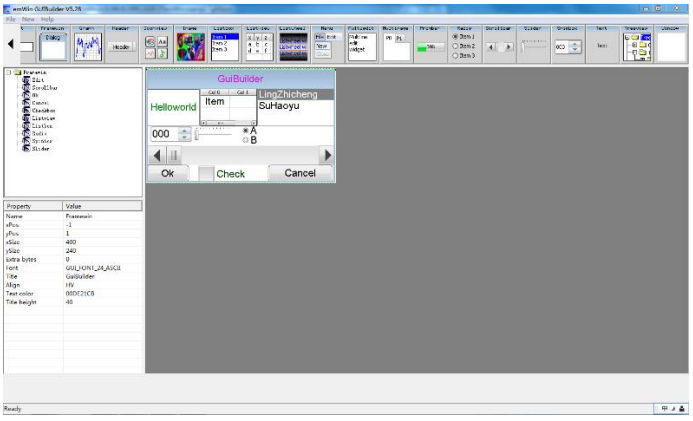


图 5.2 GUIBuilder emWIN 界面

4) 附上部分代码，初始化以及修改参数代码部分，添加过多组件代码会比较冗长。

```
/*********************************************************************
* 初始化每个组件，ID_组件名称_组件序号(GUI_ID_USER + 0x**)
**********************************************************************
*/
#define ID_FRAMEWIN_0 (GUI_ID_USER + 0x10)
#define ID_EDIT_0 (GUI_ID_USER + 0x11)
#define ID_SCROLLBAR_0 (GUI_ID_USER + 0x12)
#define ID_BUTTON_0 (GUI_ID_USER + 0x13)
#define ID_BUTTON_1 (GUI_ID_USER + 0x14)
#define ID_CHECKBOX_0 (GUI_ID_USER + 0x15)
#define ID_LISTVIEW_0 (GUI_ID_USER + 0x18)
#define ID_LISTBOX_0 (GUI_ID_USER + 0x19)
#define ID_RADIO_0 (GUI_ID_USER + 0x1C)
#define ID_SPINBOX_0 (GUI_ID_USER + 0x1E)
#define ID_SLIDER_0 (GUI_ID_USER + 0x1F)

/*********************************************************************
*
*    这里是设置一些创建组件时的初始化参数，如以组件左上角定位的 x，y 值，组件大小长宽值等
*/
static const GUI_WIDGET_CREATE_INFO _aDialogCreate[] = {
  { FRAMEWIN_CreateIndirect, "Framewin", ID_FRAMEWIN_0, -1, 1, 400, 240, 0, 0x64,
0 },
  { EDIT_CreateIndirect, "Edit", ID_EDIT_0, -1, 0, 109, 77, 0, 0x64, 0 },
  { SCROLLBAR_CreateIndirect, "Scrollbar", ID_SCROLLBAR_0, -1, 119, 399, 40, 0, 0x0,
100 },
  { BUTTON_CreateIndirect, "Ok", ID_BUTTON_0, -3, 159, 89, 36, 0, 0x0, 0 },
  { BUTTON_CreateIndirect, "Cancel", ID_BUTTON_1, 252, 159, 136, 35, 0, 0x0, 0 },
  { CHECKBOX_CreateIndirect, "Checkbox", ID_CHECKBOX_0, 105, 161, 114, 35, 0, 0x0,
0 },
  { LISTVIEW_CreateIndirect, "Listview", ID_LISTVIEW_0, 110, -2, 119, 80, 0, 0x0,
0 },
  { LISTBOX_CreateIndirect, "Listbox", ID_LISTBOX_0, 230, -2, 162, 79, 0, 0x0, 0 },
  { RADIO_CreateIndirect, "Radio", ID_RADIO_0, 197, 79, 97, 37, 0, 0x1402, 0 },
  { SPINBOX_CreateIndirect, "Spinbox", ID_SPINBOX_0, 5, 79, 88, 31, 0, 0x0, 555 },
  { SLIDER_CreateIndirect, "Slider", ID_SLIDER_0, 97, 80, 80, 28, 0, 0x0, 0 },
  // USER START (Optionally insert additional widgets)
  // USER END
};

static void _cbDialog(WM_MESSAGE * pMsg) {
  WM_HWIN hItem;
  int     NCode;
```

```
    int    Id;
    // USER START (Optionally insert additional variables)
    // USER END
    // 这里是修改组件信息的函数，也是最重要的函数之一，有两个方面，一个是用 switch 选择修改那个组件，第二个是通过 WM_HWIN  hItem;获取句柄，从而得到当前所控制组件的“地址”，即选中某个组件，在使用相关函数对组件的参数进行修改。
    switch (pMsg->MsgId) {
    case WM_INIT_DIALOG:
      //
      // Initialization of 'Framewin'
      //
      // 以下面两个为例，其他组件修改都差不多，不再赘述
      //
      hItem = pMsg->hWin;
      // 获取句柄，这是获取 FRAMEWIN 的句柄,与获取其他组件的写法稍有不同
      FRAMEWIN_SetFont(hItem, GUI_FONT_24_ASCII); // 设置字体为 GUI_FONT_24_ASCII
      FRAMEWIN_SetText(hItem, "GuiBuilder");     // 设置文本内容为"GuiBuilder"
      FRAMEWIN_SetTextAlign(hItem, GUI_TA_HCENTER | GUI_TA_VCENTER);// 设置对齐
      FRAMEWIN_SetTextColor(hItem, 0x00DE21CB);   // 设置字体颜色为 0x00DE21CB
      FRAMEWIN_SetTitleHeight(hItem, 40);        // 设置标题高度为 40
      //
      // 修改 ID_EDIT_0 组件
      //
      hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);
      EDIT_SetText(hItem, "Helloworld");
      EDIT_SetFont(hItem, GUI_FONT_24_ASCII);
      EDIT_SetTextAlign(hItem, GUI_TA_HCENTER | GUI_TA_VCENTER);
      EDIT_SetTextColor(hItem, EDIT_CI_ENABLED, 0x00008000);
      }
    }
```

5) 效果图

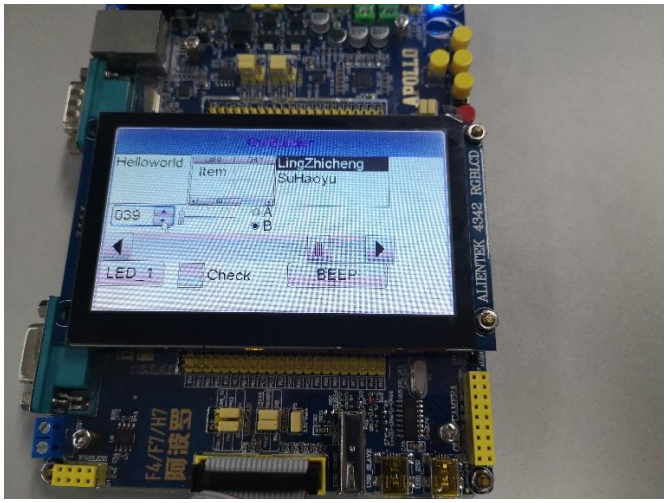


图 5.3 自设组件的 GUI 效果图

四、 实验总结

1) 第一次使用 GUIBuilder 时感觉并不陌生，非常类似于 VB 和 VC 的操作，选择组件，添加到窗体，设置各个组件的长宽大小字体颜色等属性，保存后在文件路径下产生一个对应文件名的.c 文件。
2) 实验所用 LCD 屏幕分辨率是 480*270，屏幕左上角默认为坐标 x,y=0,0，使用绝对定位的方式，较为直接，应该也能用相对坐标来表示，还没深究，如果在 GUIBuilder 中将组件的大小设置过大，下载到板子上后只会显示左上角在分辨率范围内的一部分，并不会进行适配。
3) 生成.c 文件后需要修改相应代码，即最后段的 WM_HWIN Framewin(void)函数名修改成自己创建整个 Frame 组件时设置的名字，并且在.c 文件中删除此函数的声明，否则将组建移动到 STemWin 文件中后会出现函数重复定义的情况，移动到工程文件后修改其他相关的函数名否则会出现函数名使用错误，因为工程使用默认的函数名为 Framewin 而自设的可能是其他各种名字。

任务六　Button 实验

一、　实验任务

1) 熟悉 STM32F429 开发板系统、Keil5、ST-Link 等软件的使用。
2) 学习和掌握 STemwin 的移植和开发技巧，在开发板的 LCD 触摸屏上，实现按键、动态显示窗口等设计。
3) 参考扩展例程-emwin 扩展例程，整合 Button 按钮和 Stemwin。

二、　实验工具

1) Windows 系统 PC 主机，预装 Keil5 MDK、ST-Link 等程序。
2) STM32F429 阿波罗开发板。

三、　实验过程和步骤

1) 在实验五的基础上进行了适当修改，即将 Button 的按下操作绑定到其他组件，以实现组件之间的相关调用。
2) 在 GUIBuilder 中只能创建简单的组件，并不能实现组件之间的绑定，或者是可以实现但是我们还没有发现，整个操作手册由于时间原因并没有看完。
3) 组件之间绑定操作的代码：
   a) Button0 按下实现 LED 的反转：

```
case ID_BUTTON_0: // Notifications sent by 'Ok'
     switch(NCode) {
     case WM_NOTIFICATION_CLICKED:
      break;
     case WM_NOTIFICATION_RELEASED:
      LED1=~LED1;      //LED1 反转
      break;
     }
     break;
```

   b) Button1 按下实现蜂鸣器开关

```
case ID_BUTTON_1: // Notifications sent by 'Cancel'
      switch(NCode) {
      case WM_NOTIFICATION_CLICKED:
        break;
      case WM_NOTIFICATION_RELEASED:
         beepsta=!beepsta; //最开始定义了 static u8 beepsta=1;，在这里用！实现值的反转
         PCF8574_WriteBit(BEEP_IO,beepsta); // 用 PCF8574_WriteBit 函数写入值到 I/O 口
        break;
```

```
    }
    break;
```

c) 使用 Radio 选中 A 或 B 来使得 LISTBOX 组件来分别选中两个字段

```
case ID_RADIO_0: // Notifications sent by 'Radio'
    switch(NCode) {
    case WM_NOTIFICATION_CLICKED:
      break;
    case WM_NOTIFICATION_RELEASED:
      break;
    case WM_NOTIFICATION_VALUE_CHANGED:
                hItem = WM_GetDialogItem(pMsg->hWin, ID_LISTBOX_0);
    // 用 WM_GetDialogItem 函数获取 RADIO 的句柄，拿到值使得 hItem 指向组件 RADIO
                pos=LISTBOX_GetSel(hItem);
    // 看 LISTBOX 初始化部分代码可以发现，两个字段分别对应有他们的 Sel，于是对照参考手册中的 API，用 LISTBOX_GetSel 函数拿到当前选中的 Sel 值
                pos=1-pos;
    // 此处我只设置了两个字段，所以哟弄个了最简单的 pos=1-pos，多个字段可用其他算法
                LISTBOX_SetSel(hItem,pos);
    // 用 LISTBOX_SetSel 函数写回 Sel 值，完成选中操作
      break;
    }
    break;
```

d) 通过 SPINBOX 组件修改 Text 文本框的对其模式（top，middle，bottom）

```
case ID_SPINBOX_0: // Notifications sent by 'Spinbox'
    switch(NCode) {
    case WM_NOTIFICATION_CLICKED:
      break;
    case WM_NOTIFICATION_RELEASED:
      break;
    case WM_NOTIFICATION_MOVED_OUT:
      break;
    case WM_NOTIFICATION_VALUE_CHANGED:
                hItem = WM_GetDialogItem(pMsg->hWin, ID_SPINBOX_0);
// 用 WM_GetDialogItem 函数获取 SPINBOX_0 的句柄，拿到值使得 hItem 指向组件 SPINBOX_0
                pos=SPINBOX_GetValue(hItem);
// 用 SPINBOX_GetValue 获得当前 SPINBOX 的值，为下面做铺垫
                hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);
// 用 WM_GetDialogItem 函数获取 EDIT_0 的句柄，拿到值使得 hItem 指向组件 EDIT_0
                pos=pos%3;
// 三种位置，top，middle，bottom，因为 SPINBOX 的值会超过三，用取余来反复循环三种状态
                if (pos==0){
                        EDIT_SetTextAlign(hItem,GUI_TA_TOP);
// 用 EDIT_SetTextAlign 函数来 Set 文本对齐方式，GUI_TA_TOP 等值可以查 API，下同
                        }
```

```
                        if (pos==1){
                        EDIT_SetTextAlign(hItem,GUI_TA_VCENTER);
                        }
                         if (pos==2){
                        EDIT_SetTextAlign(hItem,GUI_TA_BOTTOM);
                        }
      break;
    }
    break;
```

4) 效果显示如图 6.1 所示

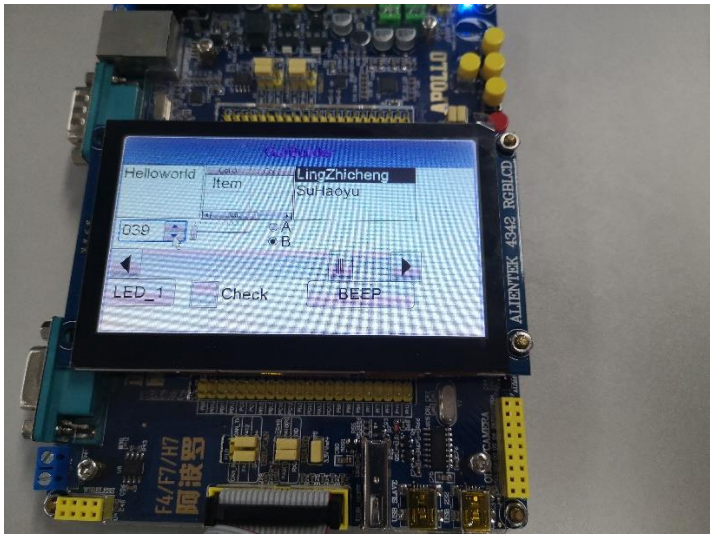


图 6.1 效果图

四、 实验总结

1) 在这次整合 Button 按钮和 Stemwin 实验中，除了完成两个按钮分别切换 LED 灯亮灭和蜂鸣器开与关，还增加 Radio 选项切换 Listbox 的的选中项目，用 Spinbox 的值来切换 Text 的文本对齐方式。

2) 主要的思路就是在 GUIBuilder 生成的.c 文件中修改相应代码，生成的.c 文件并不复杂，初始化部分和修改部分，变量命名也比较有规则，一个特点就是每次对组件操纵之前都要用 hItem = WM_GetDialogItem(pMsg->hWin, ID_SPINBOX_0);函数来获取组件句柄，只要修改 ID_****_num 即可，详细可以参考函数 API，获取句柄后然后通过修改函数来修改其他组件的属性，大量设置属性的函数使用 Set、Get，提供的参考手册不全，有时候需要猜测是否有没有这种 Set 和 Get 函数。

任务七　基于多传感器的环境评价实验

一、　实验任务

1) 熟悉 STM32F429 开发板系统、Keil5、ST-Link 等软件的使用。
2) 通过 STM32F4 开发板、数据采集板及无线路由器，实现环境监控的人机对话界面设计、环境参数接收、预警及动态显示，预警值可以动态设置。
3) 了解数据采集模块的工作原理，实现温湿度、颗粒浓度、可燃气体浓度及有机有毒气体浓度的采集、存储及基于 WIFI 网络的无线传输。
4) 设计进入界面，密码进入；进入界面后，选择 3 个环境参数显示，下面对应着三个按键；点击显示窗口，出现对应的阈值设计界面，用滚动条设计阈值的大小；点击按钮进入动态显示界面等。

二、　实验工具

1) Windows 系统 PC 主机，预装 Keil5 MDK、ST-Link 等程序。
2) STM32F429 阿波罗开发板、无线模块、路由器，装有传感器模块的开发板。

三、　实验过程和步骤

1) Display.c 部分代码

```
/*********************************************************************

*

* aDialogCreate 窗体初始化部分，为了能有能令人接收的视觉效果，重新计算了各个组件坐标点和大小长宽值

*/

static const GUI_WIDGET_CREATE_INFO _aDialogCreate[] = {

  { FRAMEWIN_CreateIndirect, "Framewin", ID_FRAMEWIN_0, 0, 0, 480, 272, 0, 0x64, 0 },

  { IMAGE_CreateIndirect, "Image", ID_IMAGE_0, 1, 0, 480, 272, 0, 0, 0 },

  { BUTTON_CreateIndirect, "Temp",  ID_BUTTON_0, 50, 70, 140, 40, 0, 0x0, 0 },

  { BUTTON_CreateIndirect, "PM2.5", ID_BUTTON_2, 50, 180, 140, 40, 0, 0x0, 0 },
```

```
  { BUTTON_CreateIndirect, "CO2",   ID_BUTTON_4, 280, 70, 140, 40, 0, 0x0, 0 },
  { BUTTON_CreateIndirect, "PM10",  ID_BUTTON_8, 280, 180, 140, 40, 0x0, 0 },
  { EDIT_CreateIndirect, "Temp",  ID_EDIT_0, 50, 20, 140, 40, 0, 0x64, 0 },
  { EDIT_CreateIndirect, "pm2.5", ID_EDIT_2, 50, 130, 140, 40, 0, 0x64, 0 },
  { EDIT_CreateIndirect, "Co2",   ID_EDIT_4, 280, 20, 140, 40, 0, 0x64, 0 },
  { EDIT_CreateIndirect, "pm10",  ID_EDIT_8, 280, 130, 140, 40, 0, 0x64, 0 },
  { TEXT_CreateIndirect, "WiFi", ID_TEXT_0, 0, 0, 80, 13, 0, 0x0, 0 },
  { TEXT_CreateIndirect, "data", ID_TEXT_1, 81, 0, 390, 13, 0, 0x0, 0 },
  // USER START (Optionally insert additional widgets)
  // USER END
};
-----------------------------------分割线-----------------------------------------
// Initialization of 'Temp'
// 部分组件属性定义的操作，其他组件类似，使用该组件的函数即可，可参考 API 文档
hItem = WM_GetDialogItem(pMsg->hWin, ID_EDIT_0);     // 获取需要设置的组件的句柄
EDIT_SetText(hItem, " ");                            // 设置文本框内容，此处为空
EDIT_SetTextAlign(hItem, GUI_TA_HCENTER | GUI_TA_VCENTER); // 设置对齐方式
EDIT_SetFont(hItem, GUI_FONT_32_ASCII);  //设置文本框的字体大小
hItem = WM_GetDialogItem(pMsg->hWin, ID_BUTTON_0);
// 获得 BUTTON_0 组件句柄，下面开始操作 BUTTON_0
BUTTON_SetFont(hItem, GUI_FONT_32_ASCII);
```

2) Image.c 部分说明

   a) const U8 _acImage_0[391734]是背景图片，在其他函数内都有调用到，因此不能使用 static 静态否则会报错，_acImage_0 是一个 391734 长度的数组，是从各种图片格式转成 16 进制的数组，经过多次尝试发现，如果是使用 BMP 格式的图片，需要的位深度是 24 位，其他位深度下不能正确设置背景，24 位深度下数组长度均为 391734，与图片分辨率无关；根据代码和 API 可知也可使用 JPG、PNG 等格式的图片，在其他函数内部将 BMP 相关的函数替换成所

使用的格式即可。

b) static const U8 _acImage_1[41198]是在背景图片之上的文字图片，只在登陆界面使用。

3) Input.c 部分说明

Input 界面就是设定阈值的界面，最开始是使用+和-按钮来进行阈值的设定，从初始化的阈值开始手动+1 或者-1，然后再调用 value.c 对数值进行保存、显示、调用等其他操作。

```
while(1)
{
    int i=0;
    if(Tp!=pl)//字符长度有变化 刷新数据
        {//如果检测到输入的字符长度发生变化，则将 Text 输出到 PBUF，即设置阈值
          hItem = WM_GetDialogItem(hWin, ID_EDIT_0);
          EDIT_SetText(hItem,(void*)PBUF);
        }
        if(closeP==3)//error 清除显示
        {
          hItem = WM_GetDialogItem(hWin, ID_TEXT_0);
          TEXT_SetText(hItem,(void*)"");
          closeP=0;
        }
   else if(closeP==2)
        {
          hItem = WM_GetDialogItem(hWin, ID_TEXT_0);
          TEXT_SetText(hItem,(void*)"Error");
        }
   else if(closeP==1)
        { // 实现将 Pbuf 的值存到 DataInfo，即实现 0~9 组合数字输入阈值，最关键的一部分
            DataInfo.Atemp=0;
```

```
                for( i=0;i<pl;i++){
                    DataInfo.Atemp=DataInfo.Atemp*10+PBUF[i]-'0';
                }
            DataInfo.Avalue[DataInfo.DtypeSel]=DataInfo.Atemp;
            AT24CXX_Write(adrr+DataInfo.DtypeSel*2,(u8*)&DataInfo.Avalue[DataInfo.Dty
            peSel],2) ;
           hItem = WM_GetDialogItem(hWin, ID_TEXT_0);
           WM_DeleteWindow(hWin);//删除窗体返回主界面

         return 0;
            }
      GUI_Delay(100);
    }
```

4) main.c 部分代码

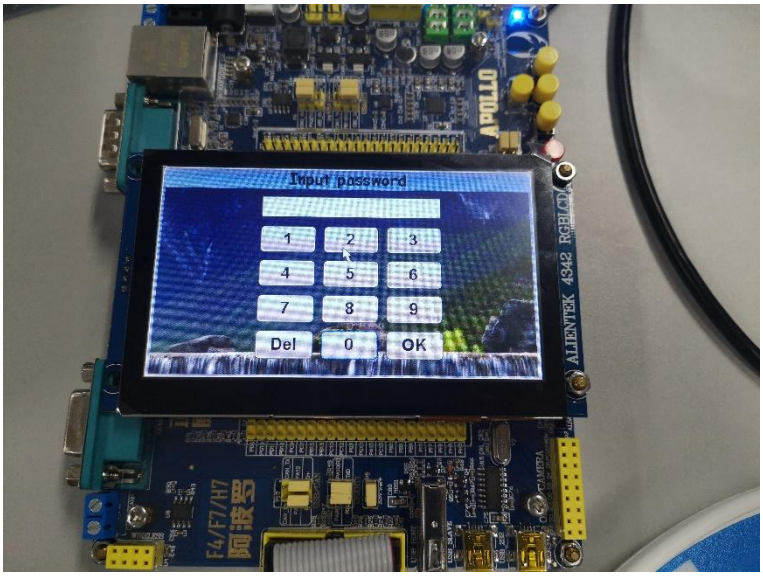
```
/*********************************************

*  main.c 中主要是对 wifi 模块进行设置，此处有一个判断是否连上 wifi 的操作，可以设置成没连上 wifi 就
在蜂鸣器警报或者是连上就有警报，都可以自定义设置。

*********************************************/

if(DataInfo.wifiF==0)//未连接 初始化连接

{

    atk_8266_send_cmd("AT","OK",50);

    atk_8266_send_cmd("AT+CWMODE=1","OK",50);      //设置 WIFI STA 模式

    atk_8266_send_cmd("AT+RST","OK",20);       //重启模块

    delay_ms(4000);         //延时 2S 等待重启成功

    PCF8574_WriteBit(BEEP_IO,1);

     printf("pre-2222222222222222222\r\n");

    sprintf((char*)p,"AT+CWJAP=\"%s\",\"%s\"",wifista_ssid,wifista_password);

    //设置无线参数:ssid,密码

    while(atk_8266_send_cmd(p,"WIFI GOT IP",300));
```

```
    //连接目标路由器,并且获得 IP
    PCF8574_WriteBit(BEEP_IO,1);
    printf("pre-333333333333333333333333\r\n");
    delay_ms(1000);
    sprintf((char*)p,"AT+CIPSTART=\"UDP\",\"255.255.255.255\",8080,8086");
    //配置目标 UDP 服务器 8080 远程端口 8086 本地端口   ipbuf,,(u8*)portnum
    delay_ms(200);
    atk_8266_send_cmd("AT+CIPMUX=0","OK",20);  //单链接模式
    delay_ms(200);
    while(atk_8266_send_cmd(p,"OK",500));
    printf("pre-44444444444444444444444444444\r\n");
    memset(USART3_RX_BUF,0,sizeof(USART3_RX_BUF));//将缓存区清 0
    USART3_RX_STA=0;              //启动下一次接收
}
```

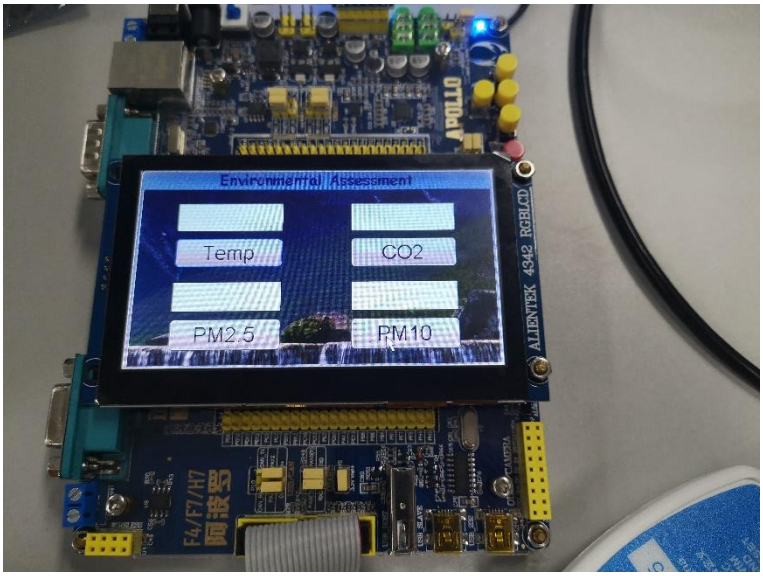
5) 演示结果

a) 登陆首页

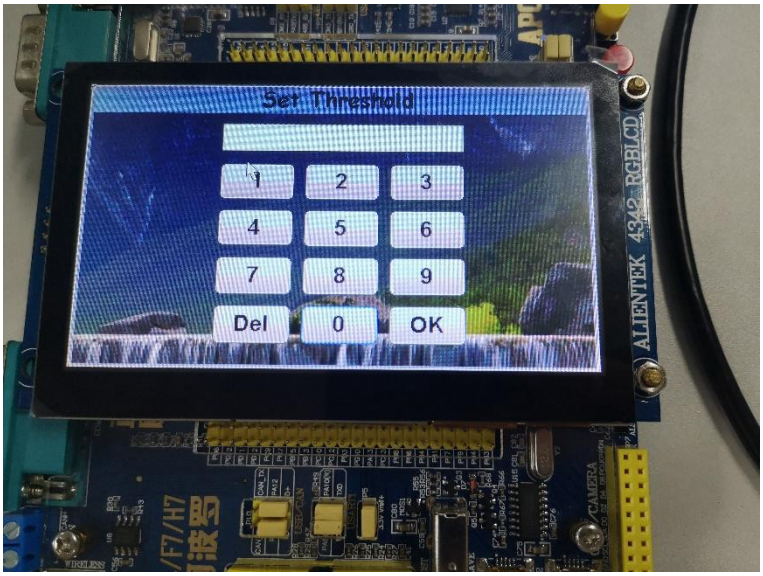
b) 密码登录界面



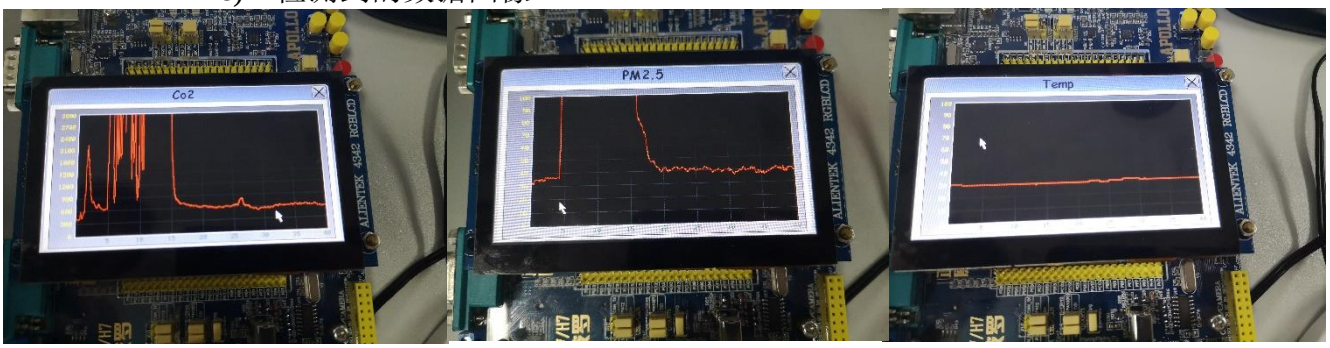
c) 为连上 WIFI 前的检测初始界面



d) 输入设置阈值画面



e) 检测到的数据图像



四、 实验总结

1) WM_DeleteWindow(hWin);//删除窗体，整个 GUI 是采用窗体叠加的形式，而不是用窗体切换，底层是背景，之后开启的窗体在其之上层层叠加，操作完毕 OK 或者返回键删除窗体，即恢复到原来的界面。
2) 整体思路就是借用密码登录界面的布局，整体移植到阈值设定模块，最难操作的就是在如何将输入的阈值绑定到单片机内部实际设定的阈值，为此进行多次修改和尝试最后成功，并且让其显示在其中一个文本框以证明我们的想法和实践是正确的。
3) 这次课程设计的时间比较紧迫，也遇到很多问题，虽然老师有给我们关于课设的资料，通过网上查询，请教老师去解决我以发现的问题，总的来说，这次课程设计对我很有帮助，让我获益匪浅，使我在这方面的知识有了不小的提高。