湖南大學

HUNAN UNIVERSITY

创新设计报告

课　程　名　称：电子与计算机系统工程实训

设计项目名称：可视化交互集成图鉴终端

专　业　班　级：计科 2301

姓　　名：常一帆

学　　号：202308010126

指　导　教　师：张洪杰

完　成　时　间：2025 年 9 月 18 日

信息科学与工程学院

摘要

在日常学习和社交场景中，常需要快速识别特定对象并获取其相关信息， 传统纸质图鉴或纯软件图鉴缺乏硬件交互的便捷性和趣味性。而小学期通过 STC 开发板系统学习了单片机编程、串口通信等基础硬件开发技术，同时我接触到嘉立创 K230 开发板的 AI 部署能力，具备将硬件交互与图像识别结合的技术基础融合 STC 开发板硬件交互与 K230 开发板 AI 算力，实现 “硬件 + 软件 + AI” 一体化的可视化图鉴终端，突破传统图鉴的纯软件或纯硬件局限； 打造兼具实用性和扩展性的硬件终端，为类似智能识别交互类项目提供开发参考。

STC 板通过串口通信结合使用嘉立创 K230 开发板，通过 yolo 训练数据集，可以简单识别跟数据集内相似度高的成员并显示其相关信息。另外附有手动查找功能以及 STC 板可实现的多个基础项目移植(利用 BSP 库函数) 采用 LCD 显示模块，接收 K230 传输的识别结果并可视化展示； 集成按键模块，实现 “手动查找” 功能（通过按键结合编写的虚拟键盘查找）；移植振动传感、霍尔传感模块（BSP 库：传感器采集函数），触发传感时自动刷新图鉴内容（如振动时切换识别模式） 移植 LED 指示灯、蜂鸣器等基础项目：识别成功时 LED 闪烁、蜂鸣器提示等等

最终结果符合预期要求，并且在开发过程中获得了额外的灵感并进行了实装（如全双工 RTC、磁力系生物探测、摇一摇随机切换等功能）。STC 的数码管和 LED 随着 K230 的不同子菜单选择通过串口通信进行变化，实现了双板合一。

具体项目详见 https://github.com/Chlaki7/Nautilus-Prime

目录

1. 绪论

1.1 选题背景

在日常学习和社交场景中，常需要快速识别特定对象并获取其相关信息， 传统纸质图鉴或纯软件图鉴缺乏硬件交互的便捷性和趣味性。而小学期通 过 STC 开发板系统学习了单片机编程、串口通信等基础硬件开发技术，同 时我接触到嘉立创 K230 开发板的 AI 部署能力，具备将硬件交互与图像 识别结合的技术基础 融合 STC 开发板硬件交互与 K230 开发板 AI 算力，实现 “硬件 + 软件 + AI” 一体化的可视化图鉴终端，突破传统图鉴的纯软件或纯硬件局限； 打造兼具实用性和扩展性的硬件终端，为类似智能识别交互类项目提供开发参考。

1.2 任务与功能

STC 板通过串口通信结合使用嘉立创 K230 开发板，通过 yolo 训练数据集，可以简单识别跟数据集内相似度高的成员并显示其相关信息。另外附有手动查找功能以及 STC 板可实现的多个基础项目移植(利用 BSP 库函数) 采用 LCD 显示模块，接收 K230 传输的识别结果并可视化展示； 集成按键模块，实现 “手动查找” 功能（通过按键结合编写的虚拟键盘查找)；移植振动传感、霍尔传感模块（BSP 库：传感器采集函数)，触发传感时自动刷新图鉴内容（如振动时切换识别模式） 移植 LED 指示灯、蜂鸣器等基础项目：识别成功时 LED 闪烁、蜂鸣器提示等等

1.3 本文结构

接下来会分为系统总体设计、硬件、软件以及系统测试几个方面来全面展开本项目。

系统总体设计主要讲述系统以何种方式进行各功能间的切换、各板所

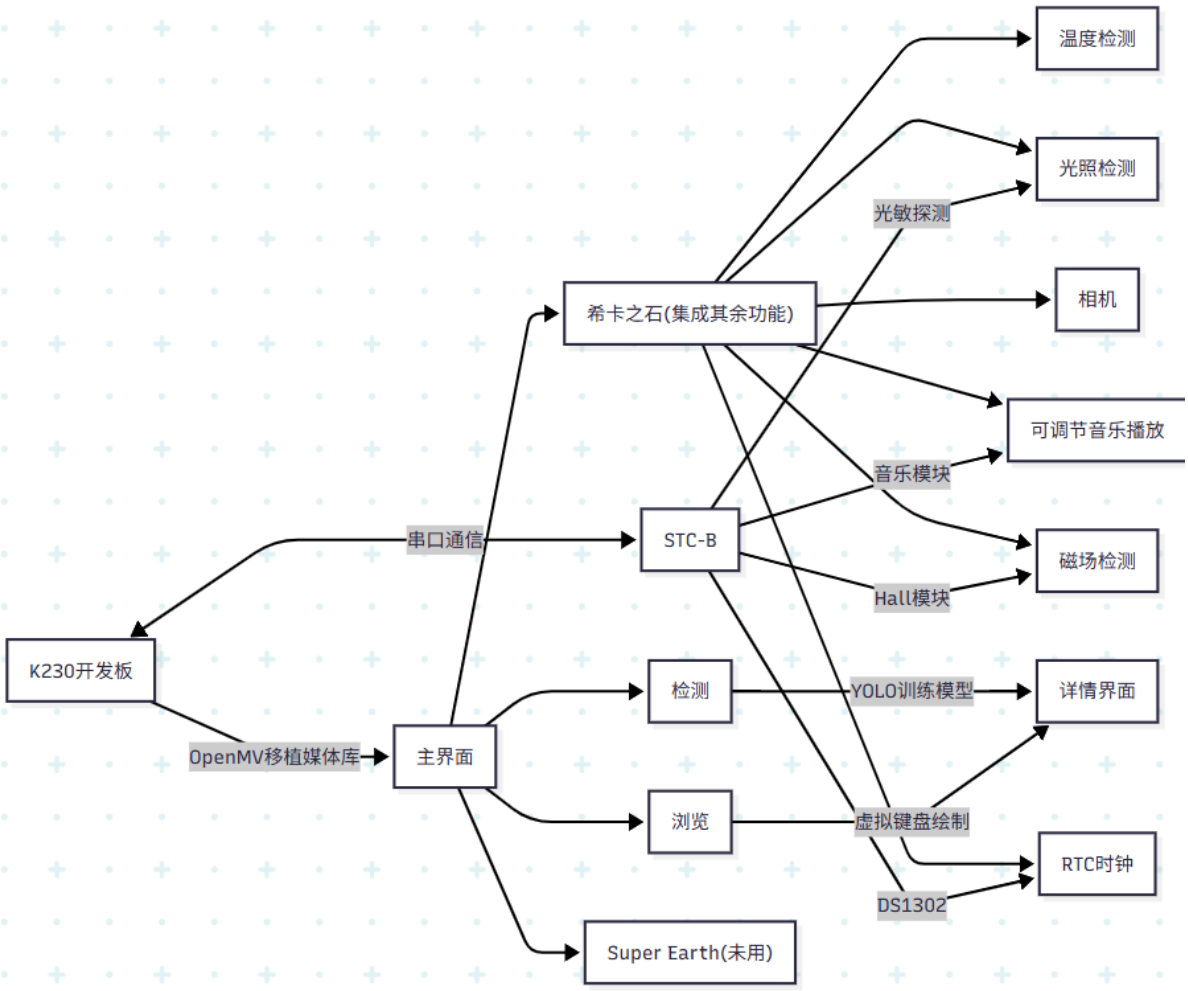
用到的模块、两板的通信方式以及设计的难点；

硬件主要讲述所用到主要元器件的原理、外接模块的连接方式和工作原理；

软件主要讲述各功能是如何编写的、代码的总体架构实现；

系统测试主要讲述具体的操作演示过程及现象。

2. 系统总体设计与分工



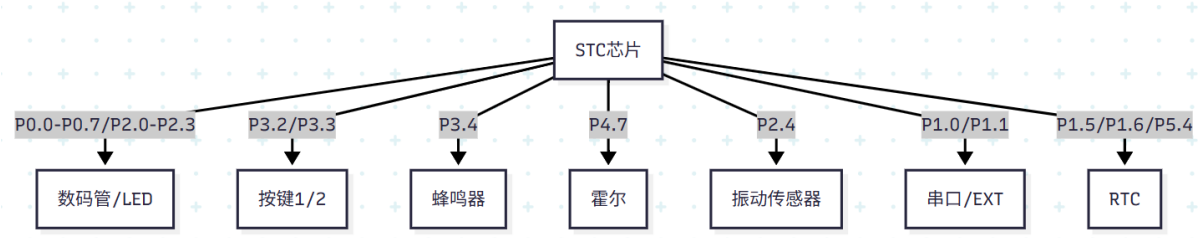
该项目分别在两块板上进行开发，通过串口进行通信。总体架构为 K230 上通过移植 openmv 库实现多级菜单，再一步步实现具体功能。拿主页面来举例：给板上电后，先进行所有全局变量的初始化，然后就进入 while true 循环。根据初始化变量，第一个进入初始界面，首先清空画布，随后是打印字体，绘制矩形拟制子菜单，再通过配置外接的五向按键进行总控选择。

通过按键，即可在不 RST 的情况下在整个系统内自由切换功能。并且在大部分的功能里，双板会通过串口进行互动操作，比如：刚上电的初始状态，STC 数码管会显示项目名“Nautilus”；不同按键进行选择时，数码管会实时显示对应模块名，如“Detect”“Browse”“Sheikah”等等。

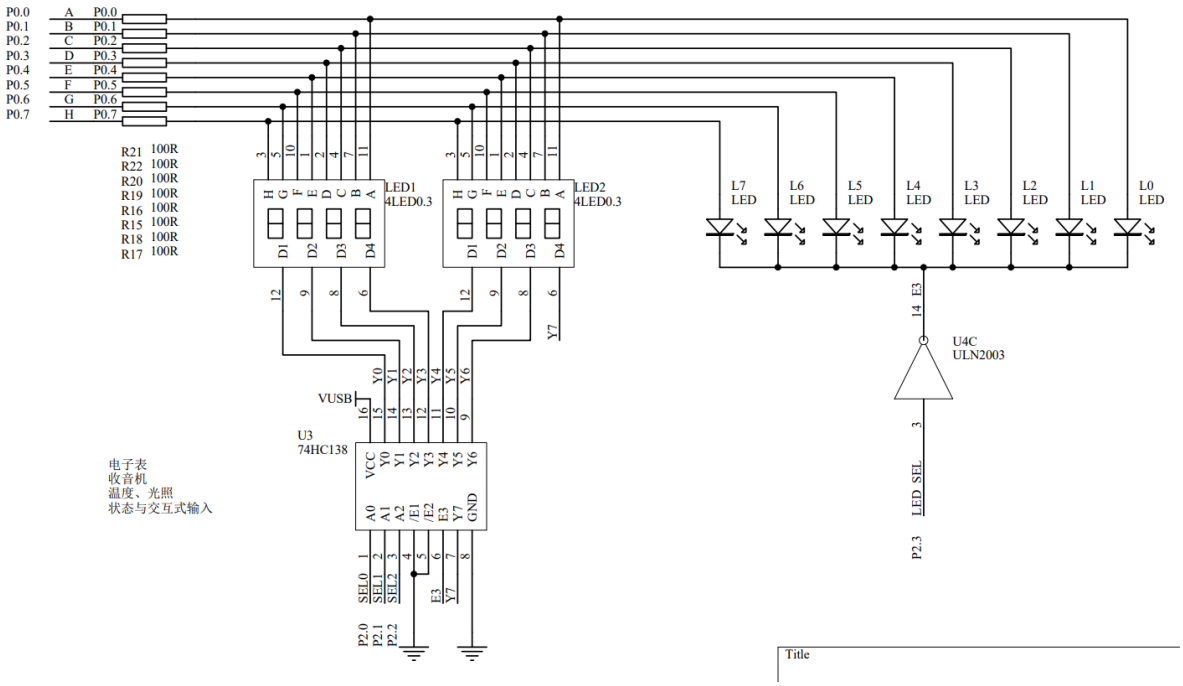
设计的难点之一是如何高效的在各功能之间进行切换，如何进行高精度模型的训练以及如何整理数据集使得一个对象的所有元素可以共用，等等。全部难点具体会在之后的软硬件章节进行展示。

3. 硬件电路原理

STC



数码管和 LED



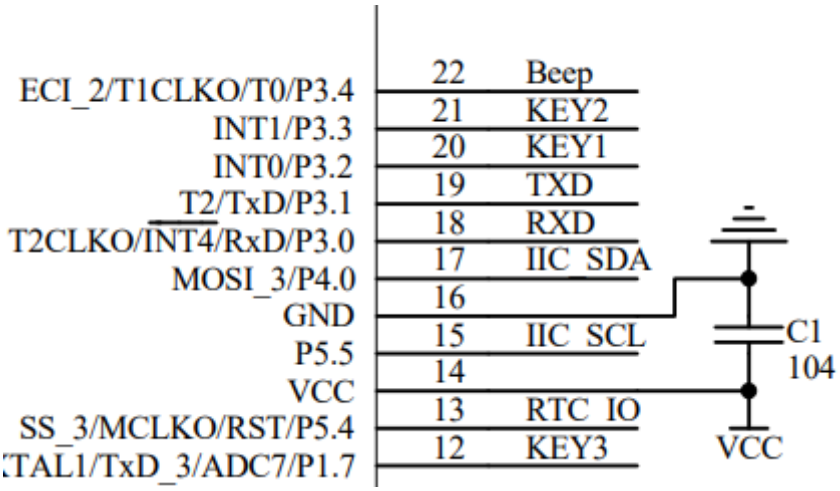
displayer 用于控制“STC-B 学习板”上 8 个 7 段数码管（Seg7）和 8

个指示灯（Led）工作

Seg7Print(char d0,char d1,char d2,char d3,char d4,char d5,char d6,char d7)：将 8 个参数值分别译码显示到对应的数码管上。显示译码表(code char decode_table[])在 main.c 中,用户可以修改和增减.

LedPrint(char led_val)：控制 8 个指示灯开关。参数 light_val 的 8 个 bit 位对应 8 个指示灯的开关,“1”——指示灯“亮”

按键

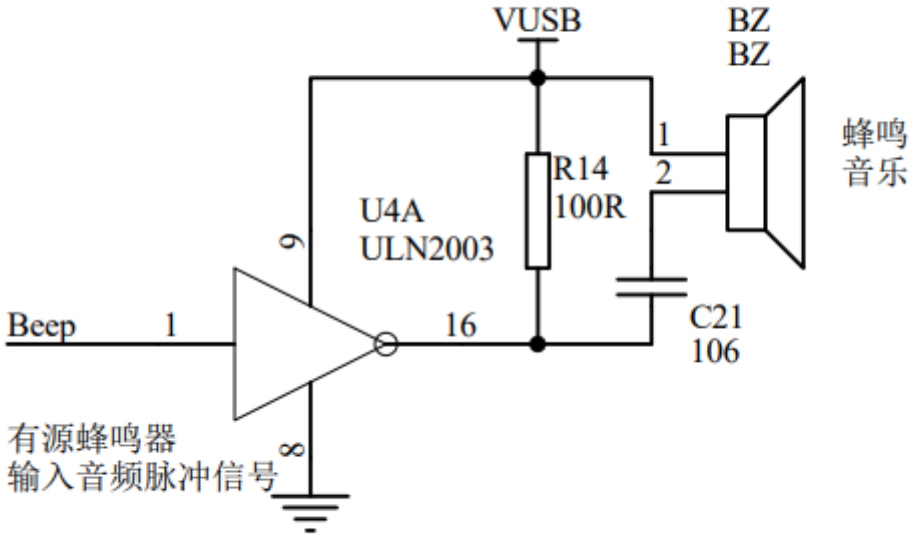


Key 模块用于获取“STC-B 学习板”上三个按键的状态

GetKeyAct(char Key)：获取按键状态

当三个按键（enumKey1,enumKey2,enumKey3）中任意一个按键有”按下“或”抬起“动作时，将产生一个”按键事件“，响应按键事件的用户处理函数由用户编写,并有 sys 中提供的 SetEventCallBack 函数设置.

蜂鸣器



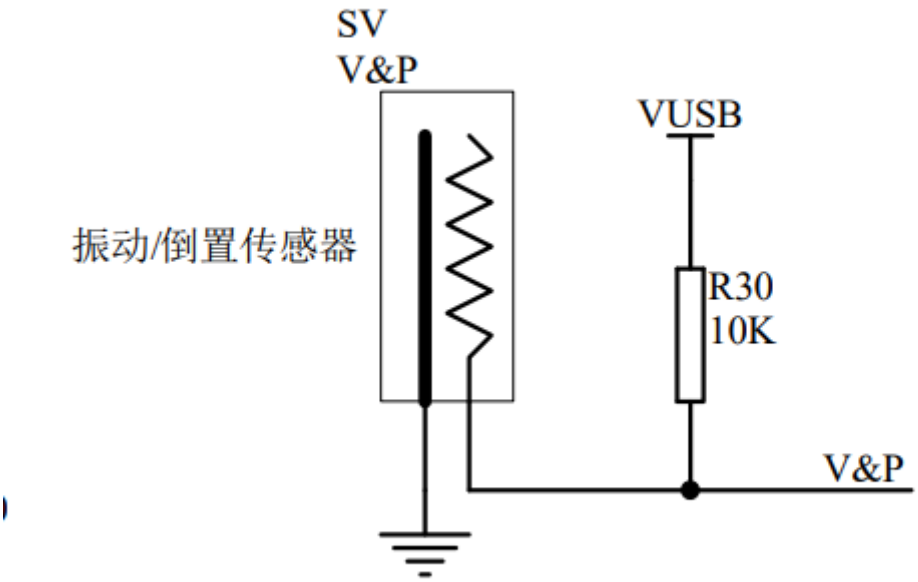
Beep 用于控制“STC-B 学习板”上无源蜂鸣器的发声

Set_Beep(unsigned int Beep_freq, unsigned char Beep_time)：控制蜂鸣器发声，非阻塞型；

Beep_freq：指定发声频率，单位 Hz。小于<10 Hz，不发音

Beep_time：指定发声时长。发声时长＝10*Beep_time (mS) ，最长 655350mS

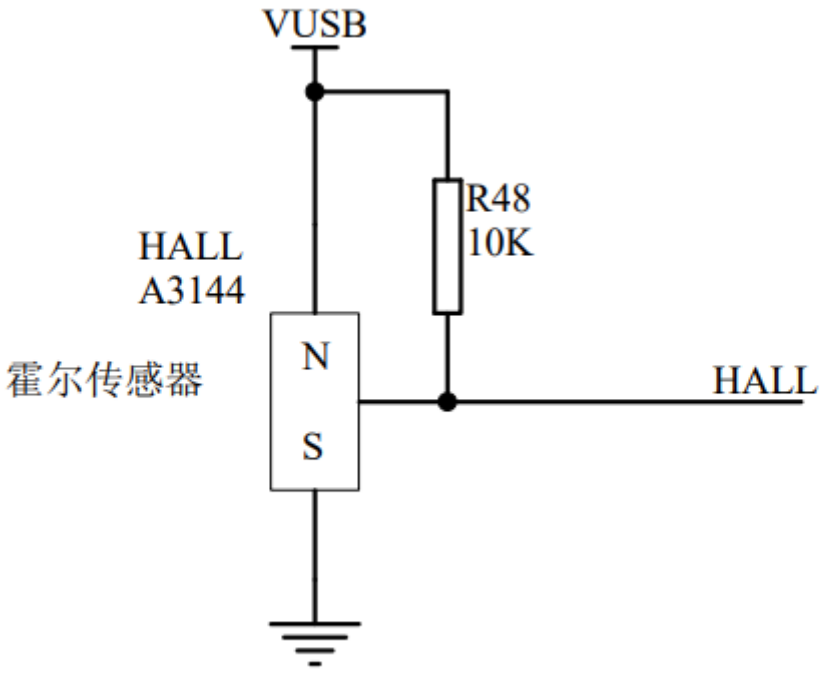
振动传感器



SV 模块用于获取"STC-B 学习板"上 Vib 传感器状态.

当 Vib 检测到有”振动“事件时，将产生一个”振动事件“，响应事件的用户处理函数由用户编写,并有 sys 中提供的SetEventCallBack()函数设置振动事件用户处理函数

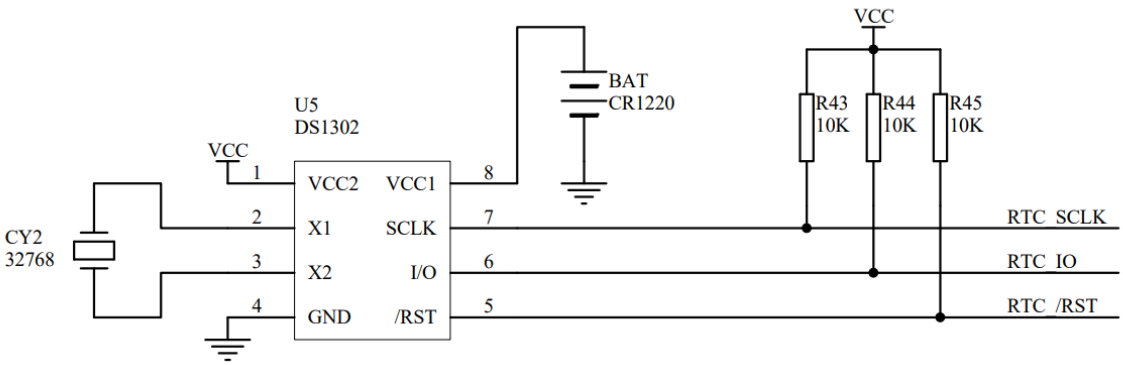
霍尔传感器



Hall 模块用于获取“STC-B 学习板”上 hall 传感器状态。

当 Hall 检测到有"磁场接近"或"磁场离开"事件时，将产生一个 Hall 传感器事件(enumEventHall).响应事件的用户处理函数由用户编写，并 有 sys 中提供的 SetEventCallBack()函数设置事件响应函数.

DS1302



DS1302 模块用于控制“STC-B 学习板”上 DS1302 芯片操作。

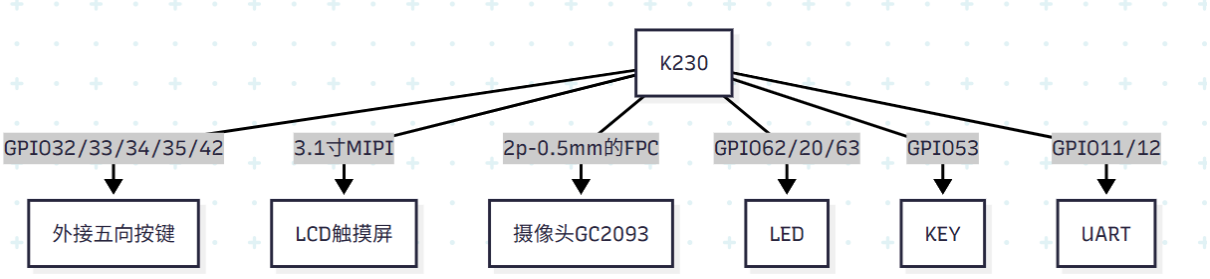
DS1302 提供 RTC（实时时钟）和 NVM（非易失存储器）功能（断电后，RTC 和 NVM 是依靠纽扣电池 BAT 维持工作的)。其中：

RTC 提供：年、月、日、星期、时、分、秒功能

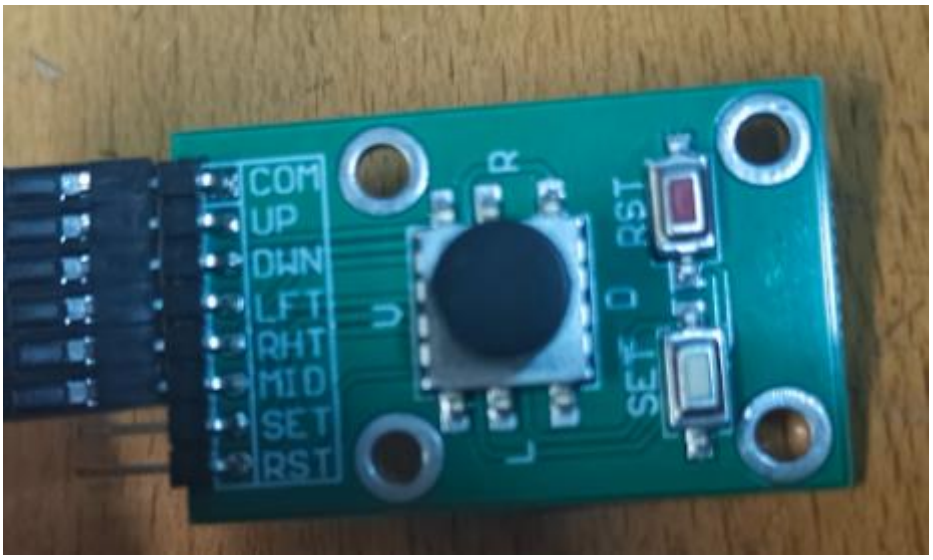
NVM 提供：31 Bytes 非易失存储器功能(地址为：0～30)。地址 30 单元被 DS1302Init 函数用于检测 DS1302 是否掉电，用户不能使用。

通过 RTC_Read 和 RTC_Write 进行读写

K230



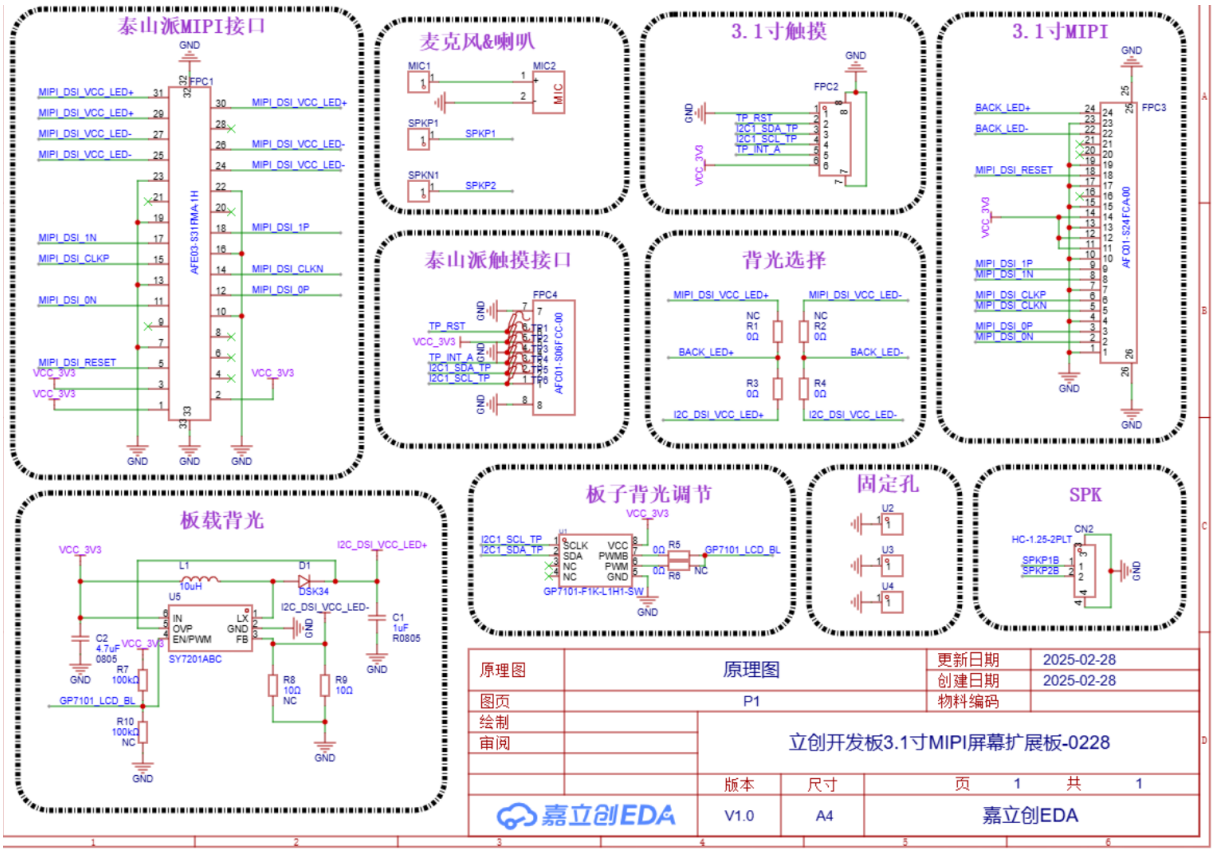
外接五向按键



使用 FPIOA 自配引脚

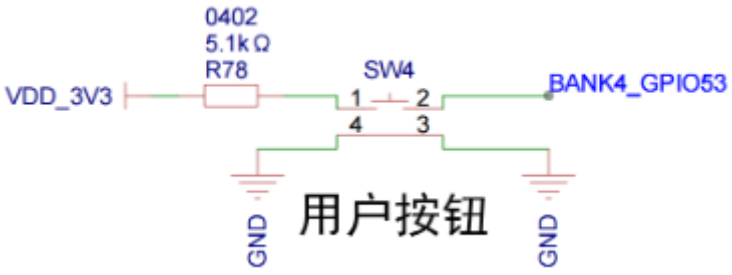
```
# 上下左右中
fpioa.set_function(32, FPIOA.GPIO32)
fpioa.set_function(42, FPIOA.GPIO42)
fpioa.set_function(35, FPIOA.GPIO35)
fpioa.set_function(34, FPIOA.GPIO34)
fpioa.set_function(33, FPIOA.GPIO33)
```

LCD 触摸屏



LED

KEY



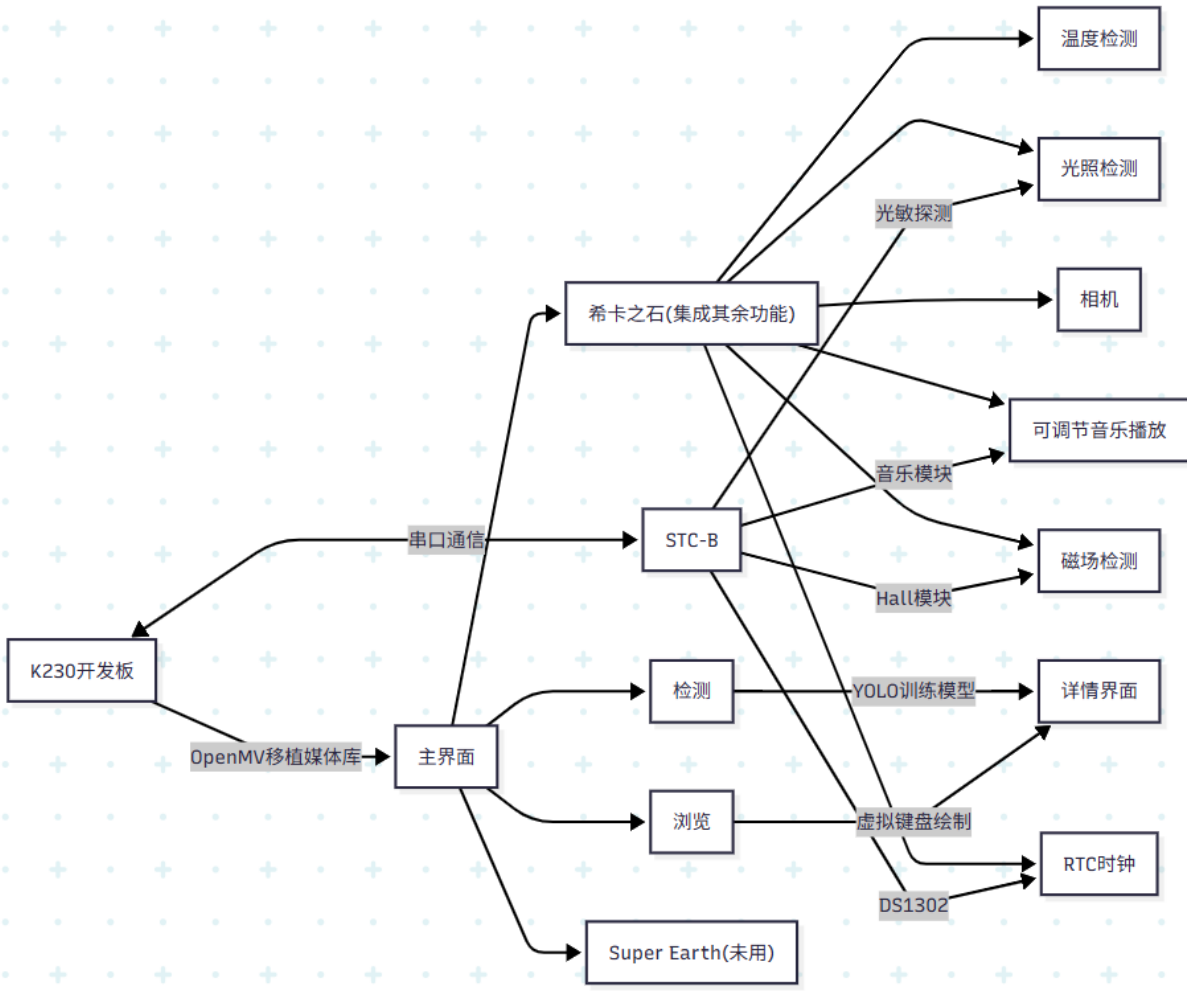
下拉电阻，按下为高电平

UART

使用 FPIOA 自配引脚

```
# 串口收发
fpioa.set_function(11, FPIOA.UART2_TXD)
fpioa.set_function(12, FPIOA.UART2_RXD)
```

4. 软件设计与实现



K230 五向按键总控

首先根据硬件原理图分配引脚

```
fpioa = FPIOA()

# 上下左右中
fpioa.set_function(32, FPIOA.GPIO32)
fpioa.set_function(42, FPIOA.GPIO42)
fpioa.set_function(35, FPIOA.GPIO35)
fpioa.set_function(34, FPIOA.GPIO34)
fpioa.set_function(33, FPIOA.GPIO33)
```

主函数里，采用类似状态机的原理，手动获取下降沿(按键是上拉电阻)

```
# 按键实例化
key1 = Pin(32, Pin.IN, pull=Pin.PULL_UP, drive=7) # 上
key2 = Pin(42, Pin.IN, pull=Pin.PULL_UP, drive=7) # 下
key3 = Pin(35, Pin.IN, pull=Pin.PULL_UP, drive=7) # 左
key4 = Pin(34, Pin.IN, pull=Pin.PULL_UP, drive=7) # 右
key5 = Pin(33, Pin.IN, pull=Pin.PULL_UP, drive=7) # 中
```

初始化

```
last_key_state1=1
last_key_state2=1
last_key_state3=1
last_key_state4=1
last_key_state5=1
```

循环末尾

```
last_key_state1 = current_key_state1
last_key_state2 = current_key_state2
last_key_state3 = current_key_state3
last_key_state4 = current_key_state4
last_key_state5 = current_key_state5
```

通过 if 获取按键状态

```
911             # 确认键
912 >           if current_key_state5 == 0 and last_key_state5 == 1: ···
974
975             # 上
976 >           if current_key_state1 == 0 and last_key_state1 == 1: ···
1027            # 下
1028 >          if current_key_state2 == 0 and last_key_state2 == 1: ···
1081            # 左
1082 >          if current_key_state3 == 0 and last_key_state3 == 1: ···
1132            # 右
1133 >          if current_key_state4 == 0 and last_key_state4 == 1: ···
1183
1184            # 返回
1185 >          if button.value() == 1: ···
```

界面设计

使用 OpenMv 库移植模块

手动根据绘制效果一点点移动坐标到合适位置，通过全局变量 menu_collect 控制哪个子菜单高亮，高亮用反色实现，也就是(255,255,255)->(0,0,0)

```
#主界面
if flag == -1:
    img.clear()
    img.draw_string_advanced(95, 25, 80, "NAUTILUS-PRIME", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
    img.draw_string_advanced(265, 385, 30, "Press Any Key to Start", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
    img.draw_string_advanced(225, 425, 40, "Presented by Zaphkiel", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
    img.draw_image(pokedex_img, 310, 170, 1.5, 1.5,alpha=256)
    #暂时无效，原因未知，只能用Display
    #改变了img通道后完全解决
    #if flag == -1:
        #Display.show_image(pokedex_img,x=350,y=200,layer = Display.LAYER_OSD1)

    # 触摸控制
    p = tp.read()
    if p != () and time_flag == 1: ···

    chosen_color=255

    # 选中视觉效果
    if menu_collect == 1: ···
    elif menu_collect==2: ···
    elif menu_collect == 3: ···
    elif menu_collect == 4: ···
    # 主界面
    else: ···
```

同时，为了使 STC 板的数码管同步显示，每当 menu 不为 0 时，向串口发送数据

```
# 选中视觉效果
if menu_collect == 1:
    uart_data=bytes([0xaa, 0x55, 0x00, 0x01])
    uart.write(uart_data)
```

STC 通过检测传奇包头 0xaa 0x55 来接收数据，通过显示回调函数来显示不同的单词

```
void myDisplay_callback()
{
    char flag = data_recieved[2];
    char menu_select = data_recieved[3];
    if (flag == 0x00)
    {
        if (menu_select == 0x00)
            Seg7Print(23, 10, 30, 29, 18, 21, 30, 28);
        else if (menu_select == 0x01)
            Seg7Print(42, 13, 40, 29, 40, 38, 29, 42);
        else if (menu_select == 0x02)
            Seg7Print(42, 11, 27, 24, 32, 28, 40, 42);
        else if (menu_select == 0x03)
            Seg7Print(28, 17, 40, 18, 20, 36, 17, 42);
        else if (menu_select == 0x04)
            Seg7Print(42, 14, 36, 27, 29, 17, 42, 42);
    }
    else if (flag == 0x01)
        Seg7Print(23, 10, 30, 29, 18, 21, 30, 28);
    else if (flag == 0x03)
        Seg7Print(16, 14, 29, 42, 13, 10, 35, 14);
```

查找模式

Flag==2 时进入查找模式，此时发送数据包使得 STC 的 LED 变为流水灯

```
# 进入查找模式
if(flag == 2):
    uart_data=bytes([0xaa, 0x55, 0x01, 0x04])
    uart.write(uart_data)
```

虚拟键盘采用遍历的方式绘制，首先要定义键盘数组，然后

```
x = col * (key_width + margin) + 10 + row*10
y = row * (key_height + margin) + 270
if key_chosen==count:
    img.draw_rectangle(x, y, key_width, key_height, color=(255, 255, 255),fill=True)
    img.draw_string_advanced(x + 15, y + 5, 30, the_key, color=(0, 0, 0), font="/sdcard/res/font/ChillBitmap7x.ttf")
else:
    img.draw_rectangle(x, y, key_width, key_height, color=(255, 255, 255))
    img.draw_string_advanced(x + 15, y + 5, 30, the_key, color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
```

根据坐标制作功能键随机和退格

```
# 制作功能键 一二三行末尾分别是：(710,270)(650, 340)(520, 410)
if key_chosen==20:
    img.draw_rectangle(650, 340, 120, key_height, color=(255, 255, 255),fill=True)
    #img.draw_string_advanced(650, 340, "Random", color=(0, 0, 0), font="/sdcard/res/font/ChillBitmap7x.ttf")
    img.draw_string_advanced(660, 340, 30,"Random", color=(0, 0, 0), font="/sdcard/res/font/ChillBitmap7x.ttf")
else:···

if key_chosen == 28:
    img.draw_rectangle(520, 410, 190, key_height, color=(255, 255, 255), fill=True)
    img.draw_string_advanced(530, 410, 30, "<- Backspace", color=(0, 0, 0), font="/sdcard/res/font/ChillBitmap7x.ttf")
else:···
```

通过读取 TF 卡内文件数据进行搜索，分为两种匹配方式，当显示界面宝可梦少于 10 个时补充模糊匹配

```
with open("/data/pinyin.txt", "r",encoding='utf-8') as f:
    pinyin_list = eval(f.read())

pinyin_results=[]
for i,pinyin in enumerate(pinyin_list):
    if pinyin.startswith(input_text):
        pinyin_results.append(i)

result_num=len(pinyin_results)

# 补充模糊匹配
if result_num<10:
    for i,pinyin in enumerate(pinyin_list):
        if not pinyin.startswith(input_text) and input_text in pinyin:
            pinyin_results.append(i)

del pinyin_list
gc.collect()
```

拍摄模式

拍摄是同样使用串口启用 LED 流水灯，随后绘制摄像头采集的图像，这时可以读取串口数据，若 STC 上的振动传感器发送事件，即可以随机选择一个图鉴内的宝可梦进行展示

随机选择采用 utime 库

```
# 随机选择赋值
if random_flag==1:
    random_flag=0
    current_frame = 0
    read_init_flag = 1
    form_num=0
    current_time = utime.ticks_ms()
    k = (current_time % 386) + 1

    x=str(k)
    x='0'*(4-len(x))+x
    linkname=pokemon_linkname[k-1]
```

K230

```
# 拍摄模式
if flag == 1:
    uart_data=bytes([0xaa, 0x55, 0x01, 0x04])
    uart.write(uart_data)
    sensor.run()
    img.clear()
    # 单独提取拍摄图像
    captured_img = sensor.snapshot(chn=CAM_CHN_ID_0)
    img.draw_image(captured_img,0,0,0.4167,0.4167 )

    data = uart.read(3)
    if(data!=None and data[2]==0x01):
        random_flag = 1
```

STC

```
void myVib_callback()
{
    char k = GetVibAct();
    if (k == enumVibQuake)
    {
        uart_data[0] = 0xaa;
        uart_data[1] = 0x55;
        uart_data[2] = 0x01;
        Uart2Print(uart_data, 3);
    }
}
```

按下按键进入识图模式，为了防止消耗资源添加了一个标志位yolo_flag，记录最后返回的结果作为查找对应数据的依据

```
# 识图模式
if flag == 0 and yolo_flag == 1 and read_init_flag == 0:
    yolo_flag = 0
    print(gc.mem_free())
    img.clear()


    test_img , test_img_ori= read_image("/data/test/0001.jpg")
    rgb888p_size=[test_img.shape[2],test_img.shape[1]]
    yolo=YOLOv5(task_type="classify",mode="image",kmodel_path=kmodel_path,labels=pokemon_lables,rgb88
    yolo.config_preprocess()

    img.draw_image(test_img_ori, 100, 5, 0.5, 0.5, alpha=256)
    # 可知返回结果res为一个元组，分别代表序号和可信度(int)
    res=yolo.run(test_img)
    yolo.draw_result(res,test_img_ori)

    img.draw_image(captured_img, 5, 5, 0.5, 0.5, alpha=256)
    img.draw_string_advanced(5,5,60 ,"识别结果:"+pokemon_linkname[res[0]]+'\n'+"可信度: "+str(res[1]),

    # 这里可向串口发送数据，用于喇叭、LED等功能

    # 由于不含0，k为实际编号
    k=res[0]+1
    x=str(k)
    x='0'*(4-len(x))+x
    linkname=pokemon_linkname[k-1]

    yolo.deinit()
    gc.collect()
```

补充模型训练过程

```
  95/100    0.182G      0.966       1.34       0.94      0.978: 100%|          | 5301/5301 [03:19<00:00, 26.53it/
  96/100    0.182G      0.963       1.34      0.941      0.979: 100%|          | 5301/5301 [03:21<00:00, 26.31it/
  97/100    0.182G      0.961       1.34       0.94       0.98: 100%|          | 5301/5301 [03:19<00:00, 26.54it/
  98/100    0.182G      0.959       1.34       0.94       0.98: 100%|          | 5301/5301 [03:22<00:00, 26.19it/
  99/100    0.182G      0.956       1.34      0.939      0.979: 100%|          | 5301/5301 [03:29<00:00, 25.29it/
 100/100    0.182G      0.954       1.34      0.938      0.979: 100%|          | 5301/5301 [03:28<00:00, 25.45it/

Training complete (4.978 hours)
Results saved to runs\train-cls\exp
Predict:         python classify/predict.py --weights runs\train-cls\exp\weights\best.pt --source im.jpg
Validate:        python classify/val.py --weights runs\train-cls\exp\weights\best.pt --data 1-3data_sorted
Export:          python export.py --weights runs\train-cls\exp\weights\best.pt --include onnx
PyTorch Hub:     model = torch.hub.load('ultralytics/yolov5', 'custom', 'runs\train-cls\exp\weights\best.pt')
Visualize:       https://netron.app
```

另外须配环境转为 kmodel，这里不做演示

再次按下按键后进入识别到宝可梦的详情界面，开始绘制共同的 UI

```
if read_init_flag == 1:
    img.clear()
    img.draw_image(captured_img, 5, 5, 0.2, 0.2, alpha=256) # 缩放到左上角

    if k<=151:···
    elif k<=251:···
    elif k<=386:···
    elif k<=493:···
    elif k<=649:···
    elif k<=721:···
    elif k<=809:···
    elif k<=905:···
    else:···

    file_path="/data/gen"+str(gen)+"/"+x+linkname+"/inform.txt"

    with open(file_path, "r",encoding='utf-8') as f:···

    del data
    gc.collect()

    # 共有UI
```

根据 id 对应的文件作为源数据，过程还是比较繁琐，仅作个别列举

```
img.draw_string_advanced(450,190,40,b"特性", color=(200, 200, 200), font="/sdcard/res/font/ChillBitmap7x.ttf")

img.draw_string_advanced(300,230,30,b"种族值:", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(300,270,25,b"HP", color=(138,198,84), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(300,300,25,b"攻击", color=(248,203,60), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(300,330,25,b"防御", color=(217,136,55), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(300,360,25,b"特攻", color=(89,195,208), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(300,390,25,b"特防", color=(88,144,205), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(300,420,25,b"速度", color=(164,86,208), font="/sdcard/res/font/ChillBitmap7x.ttf")

if data1[0]!='?':···
else:···

img.draw_string_advanced(420,230,30,b"%s"%(str(data1_sum)), color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(560,190,30,b"%s"%(height), color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(640,190,30,b"%s"%(weight), color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")

img.draw_rectangle(295,270,480,180,(255,255,255), thickness=2)
img.draw_string_advanced(360,270,25,"%s"%(data1[0].split(": ")[-1]), color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(360,300,25,"%s"%(data1[1].split(": ")[-1]), color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(360,330,25,"%s"%(data1[2].split(": ")[-1]), color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(360,360,25,"%s"%(data1[3].split(": ")[-1]), color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(360,390,25,"%s"%(data1[4].split(": ")[-1]), color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
img.draw_string_advanced(360,420,25,"%s"%(data1[5].split(": ")[-1]), color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
```

宝可梦动图的显示，调试过程很容易出 bug，原理是按帧播放 gif 图片

```
# 易出bug
if time_flag == 1:
    time_flag = 0
    frame = image.Image("/data/gen"+str(gen)+"/"+x+linkname+"/gif-jpg/"+str(current_frame+1)+".bmp") # 调试
    gif_w=frame.width()
    gif_h=frame.height()
    img.draw_image(frame, 10, 250, 3, 3, alpha=256)
    current_frame = (current_frame + 1) % gif_count
    del frame
    gc.collect()
```

左右切换 id 临近的宝可梦

```
# 左右切换赋值
if change_flag1==1:
    change_flag1=0
    form_num=0
    k=(pokemon_linkname.index(linkname)-1)%386+1
    x=str(k)
    x='0'*(4-len(x))+x
    linkname=pokemon_linkname[k-1]

if change_flag2==1:
    change_flag2=0
    form_num=0
    k=(pokemon_linkname.index(linkname)+1)%386+1
    x=str(k)
    x='0'*(4-len(x))+x
    linkname=pokemon_linkname[k-1]
```

接下来是希卡之石的界面，分为 6 个子模块，分别对应 STC 的 RTC、光敏、温度、音乐、霍尔模块

```
if flag == 10:
    img.clear()
    img.draw_string_advanced(200, 15, 60, "SHEIKAH-STONE", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
    img.draw_string_advanced(265, 385, 30, "Press Any Key to Start", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
    img.draw_string_advanced(180, 425, 40, "Please Choose A Function", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
    img.draw_image(pokedex_img, 310, 170, 1.5, 1.5,alpha=256)

    # 触摸控制
    p = tp.read()
    if p != () and time_flag == 1:...


    chosen_color=255

    # 选中视觉效果
    if menu_collect == 1:...

    elif menu_collect == 2:...

    elif menu_collect == 3:...

    elif menu_collect == 4:...

    elif menu_collect == 5:...

    elif menu_collect == 6:...
    else:...
```

RTC 模式

逻辑还是很基础的，切换到此界面发送数据通知 STC，STC 调用 DS1302

模块进行时间读，随后在数码管上显示时分秒数据并把包括年月日的数据全部发回给 K230，绘制在 LCD 屏上

```
# RTC模式
if flag == 11:
    if RTC_time == 0:
        img.clear()
    RTC_time = 1
    uart_data=bytes([0xaa, 0x55, 0x11, 0x01])
    uart.write(uart_data)
    data = uart.read(15)
    if(data!=None and data[6]==0xaa):
        img.clear()
        img.draw_string_advanced(20, 120, 200, str(data[0]) + str(data[1])+ ':' + str(data[2]) + str(data[3])+ ':' + str(data[4]) + str(da
        img.draw_string_advanced(240, 80, 100, str(data[7]) + str(data[8])+ '-' + str(data[11]) + str(data[12])+ '-' + str(data[13]) + st
        img.draw_string_advanced(340, 340, 60, "星期" + str(data[10]), color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
    img.draw_string_advanced(160, 15, 60, "--现在的时间是--", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
```

STC

```
else if (flag == 0x11)
{
    unsigned char dispData[8]; // 数码管显示数据数组

    // 读取当前时间
    currentTime = RTC_Read();

    // 解析时分秒为数码管显示格式（时时-分分-秒秒）
    dispData[0] = (currentTime.hour >> 4) & 0x0F;   // 时高位
    dispData[1] = currentTime.hour & 0x0F;          // 时低位
    dispData[2] = 42;                               // '-'分隔符
    dispData[3] = (currentTime.minute >> 4) & 0x0F; // 分高位
    dispData[4] = currentTime.minute & 0x0F;        // 分低位
    dispData[5] = 42;                               // '-'分隔符
    dispData[6] = (currentTime.second >> 4) & 0x0F; // 秒高位
    dispData[7] = currentTime.second & 0x0F;        // 秒低位

    uart_data[0] = dispData[0];
    uart_data[1] = dispData[1];
    uart_data[2] = dispData[3];
    uart_data[3] = dispData[4];
    uart_data[4] = dispData[6];
    uart_data[5] = dispData[7];
    uart_data[6] = 0xaa;
    uart_data[7] = (currentTime.year >> 4) & 0x0F;
    uart_data[8] = currentTime.year & 0x0F;
    uart_data[9] = (currentTime.week >> 4) & 0x0F;
    uart_data[10] = currentTime.week & 0x0F;
    uart_data[11] = (currentTime.month >> 4) & 0x0F;
    uart_data[12] = currentTime.month & 0x0F;
    uart_data[13] = (currentTime.day >> 4) & 0x0F;
    uart_data[14] = currentTime.day & 0x0F;
    Uart2Print(uart_data, 15);
    // 输出到数码管（8位显示）
    Seg7Print(dispData[0], dispData[1], dispData[2], dispData[3],
              dispData[4], dispData[5], dispData[6], dispData[7]);
    LedPrint(0xFF);
}
```

光温模式

串口获取 STC 板上的光照值和温度 AD 值

```
# 光温模式
if flag == 13:
    if temp_time == 0:
        img.clear()
    temp_time = 1
    uart_data=bytes([0xaa, 0x55, 0x13, 0x01])
    uart.write(uart_data)
    data = uart.read(7)
    if(data!=None and data[-1]==0xaa):
        img.clear()
        img.draw_string_advanced(20, 80, 140, "Temp:" + str(data[0]) + str(data[1]) + str(data[2]) + "\nLux:" + str(data[3]) + str(data[4]) + str(data[5]),
    img.draw_string_advanced(20, 15, 60, "--现在的光照和温度(AD值)是--", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
```

模拟探测磁力系宝可梦

磁体靠近/远离霍尔元件时发送不同数据

```
# 磁力探测模式
if flag == 14:
    if mag_time == 0:
        img.clear()
        img.draw_string_advanced(100, 15, 60, "--正在探测磁力宝可梦--", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
    data = uart.read(3)
    if(data!=None and data[2]==0x02):
        mag_time = 1
        img.clear()
        img.draw_string_advanced(180, 15, 60, "!检测到磁力宝可梦!", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
        img.draw_string_advanced(200, 100, 60, "!请保持安全距离!", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
    elif(data!=None and data[2]==0x03):
        img.clear()
        img.draw_string_advanced(140, 100, 60, "~磁力宝可梦已经离开~", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
        img.draw_string_advanced(70, 280, 50, "~请按返回键以使用其他功能~", color=(255, 255, 255), font="/sdcard/res/font/ChillBitmap7x.ttf")
```

STC

```
void myHall_callback()
{
    char k = GetHallAct();
    if (k == enumHallGetClose)
    {
        uart_data[0] = 0xaa;
        uart_data[1] = 0x55;
        uart_data[2] = 0x02;
        Uart2Print(uart_data, 3);
        // Beep
        interp = 0;
    }
    else if (k == enumHallGetAway)
    {
        uart_data[0] = 0xaa;
        uart_data[1] = 0x55;
        uart_data[2] = 0x03;
        Uart2Print(uart_data, 3);
    }
}
```

互动小功能：识别成功蜂鸣器发声且 LED 高亮闪烁，数码管显示

```
# 识图进入详情界面
elif flag==0:
    read_init_flag=1
    # 识别成功，蜂鸣器发声
    uart_data=bytes([0xaa, 0x55, 0x03, 0x01])
    uart.write(uart_data)
```

```c
void beep_check()
{
    if (GetBeepStatus() == enumBeepFree && interp == 0)
    {
        a += 1;
    }

    if (a > 4)
    {
        a = 0;
    }

    if (interp == 0 && a == 1)
    {
        SetBeep(523, 17);
    }
    else if (interp == 0 && a == 2)
    {
        SetBeep(587, 17);
    }
    else if (interp == 0 && a == 3)
    {
        SetBeep(659, 17);
    }
    else if (interp == 0 && a == 4)
    {
        SetBeep(784, 80);
        interp = 1;
    }
}
```

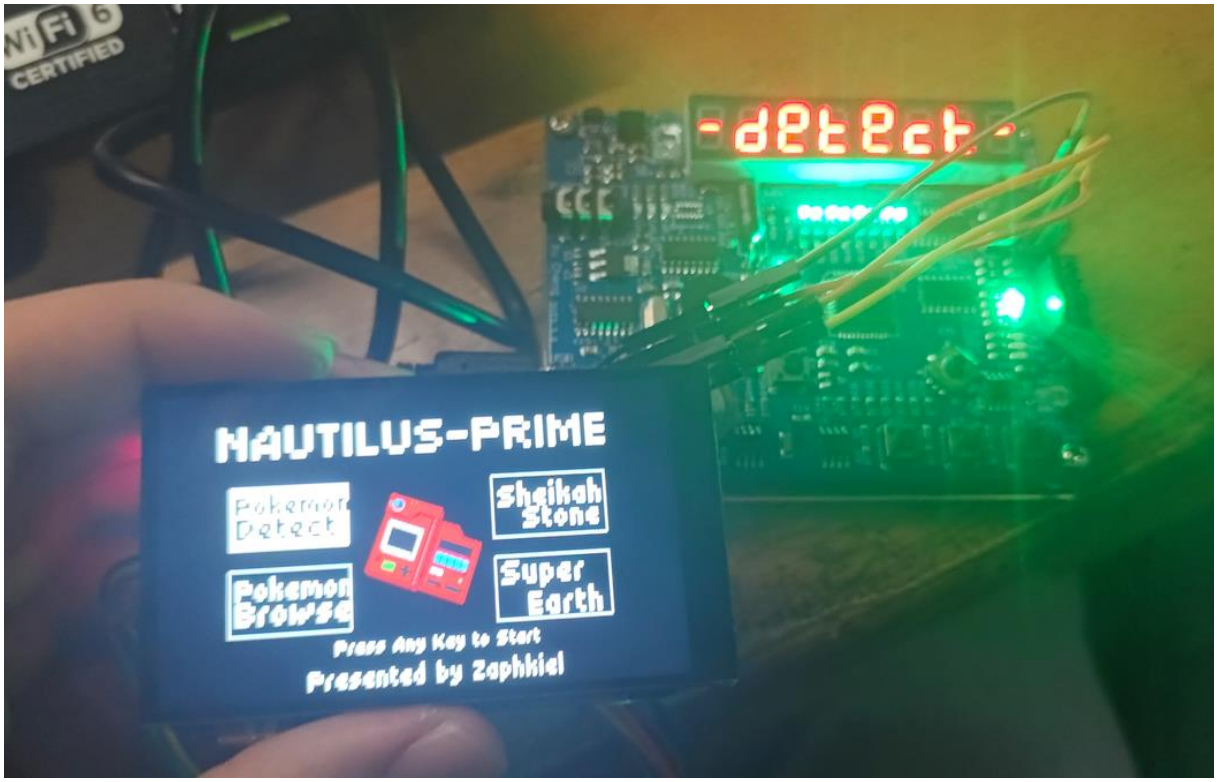
```c
else if (LEDmode == 3 && LEDchange == 1)
{
    LedPrint(0xFF);
}
else if (LEDmode == 3 && LEDchange == -1)
{
    LedPrint(0x00);
}
LEDchange *= -1;
```

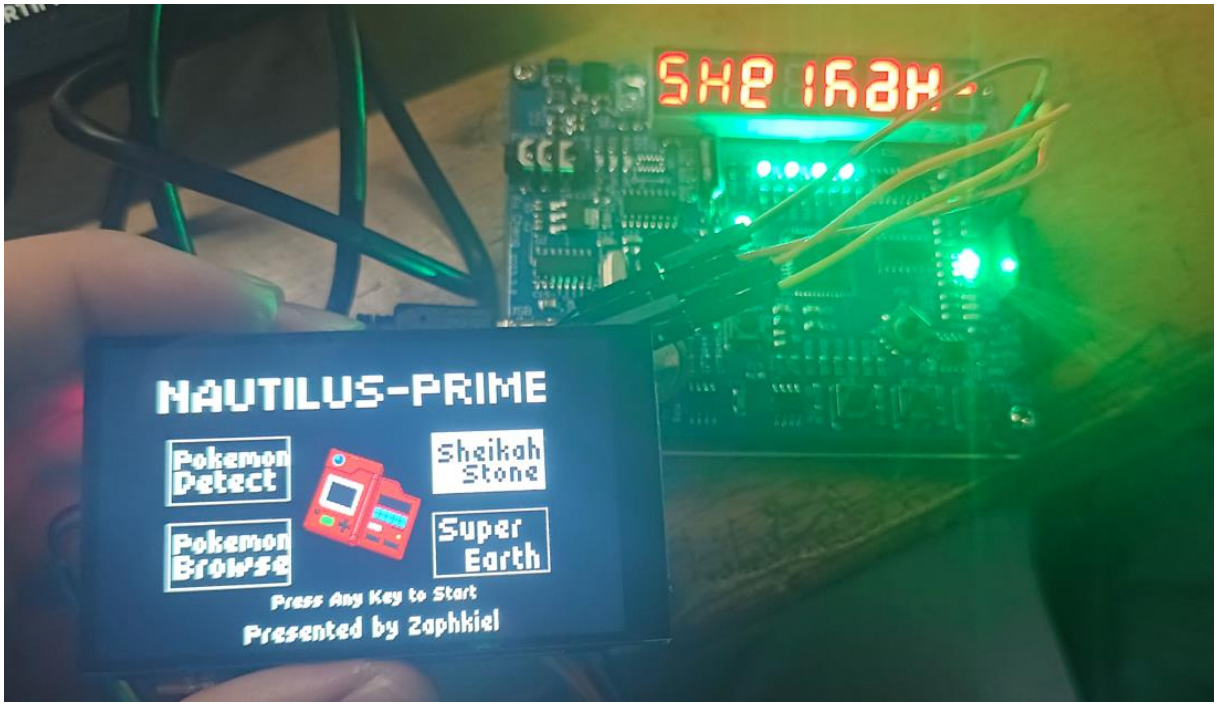
K1/K2 可控制音乐播放

```c
void myKey_callback()
{
    if (GetKeyAct(enumKey1) == enumKeyPress)
    {
        SetPlayerMode(enumModePlay);
    }
    if (GetKeyAct(enumKey2) == enumKeyPress)
    {
        SetPlayerMode(enumModePause);
    }
}
```

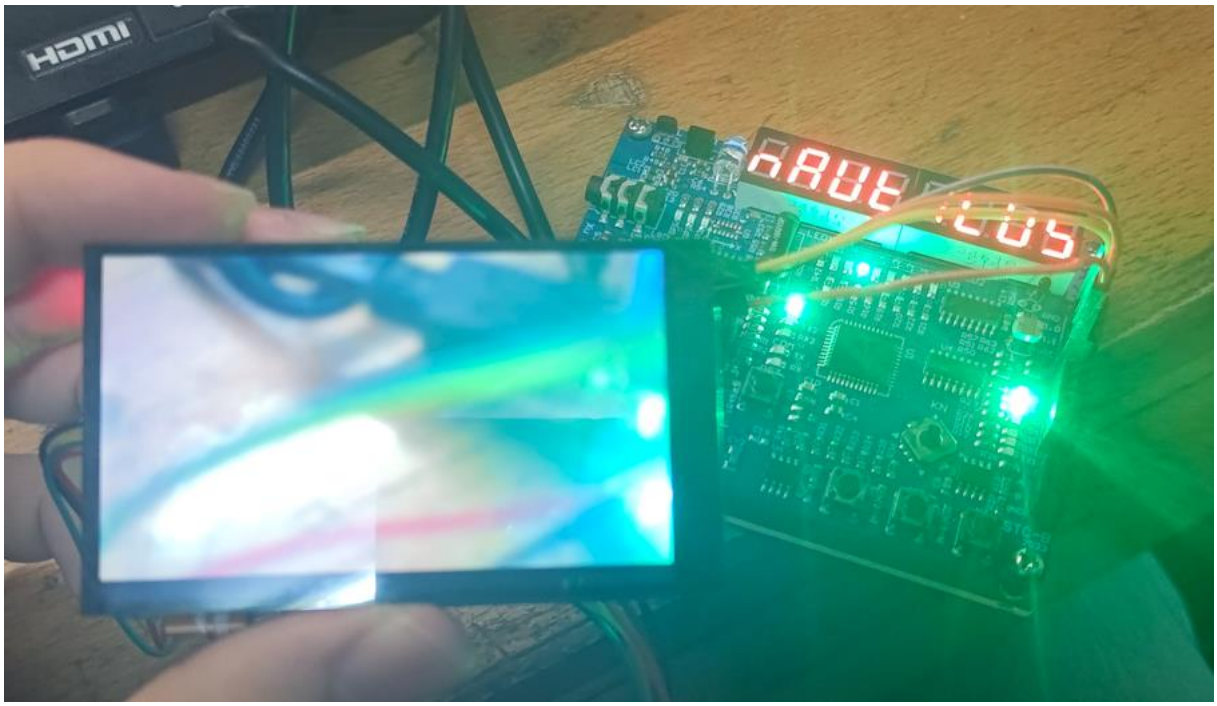
5. 系统测试与分析

主界面，数码管会实时显示对应选项

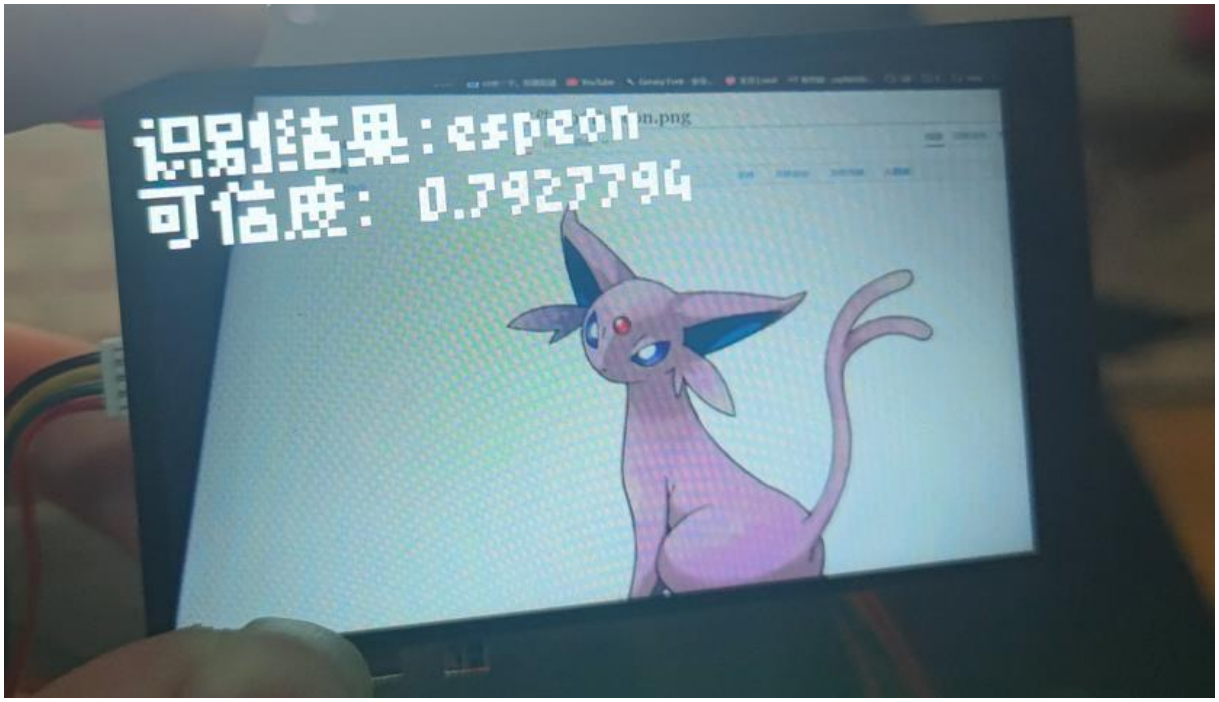

NAUTILUS-PRIME
Pokemon Detect
Sheikah Stone
Pokemon Browse
Super Earth
Press Any Key to Start
Presented by Zaphkiel


NAUTILUS-PRIME
Pokemon Detect
Sheikah Stone
Pokemon Browse
Super Earth
Press Any Key to Start
Presented by Zaphkiel

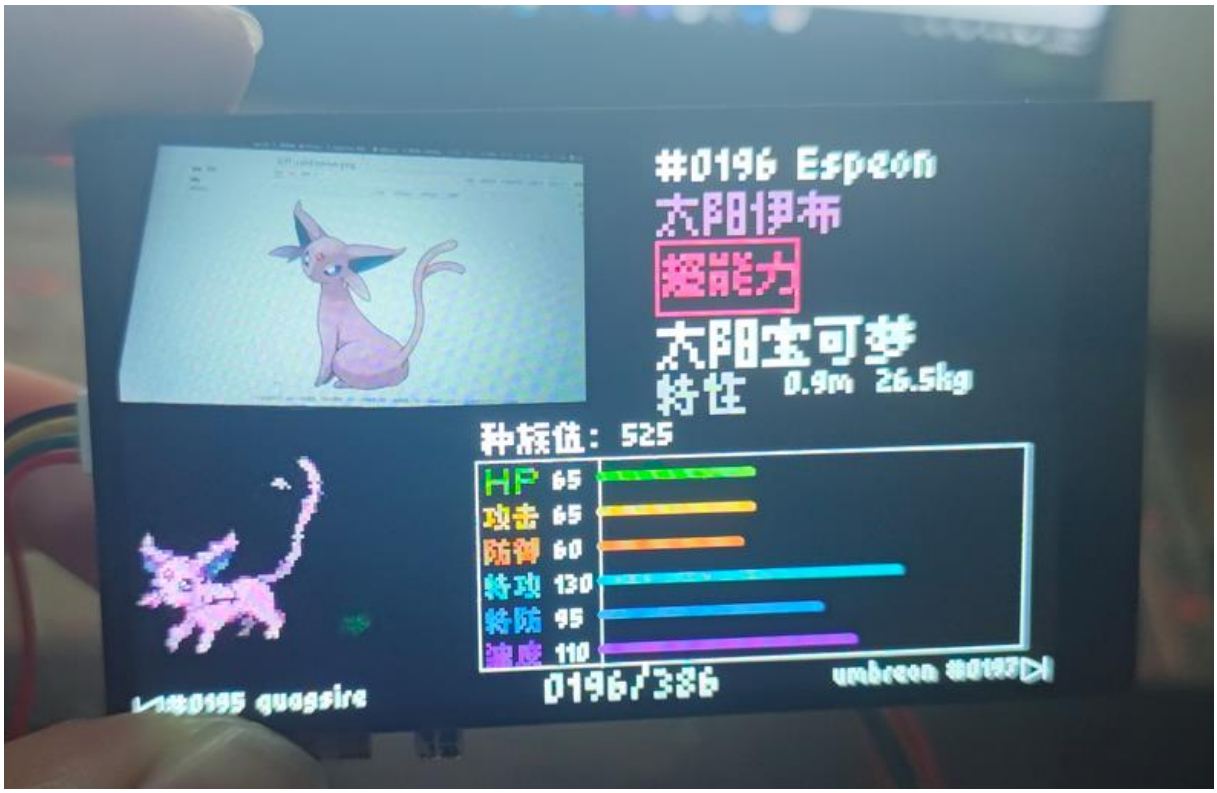
拍摄界面正常工作



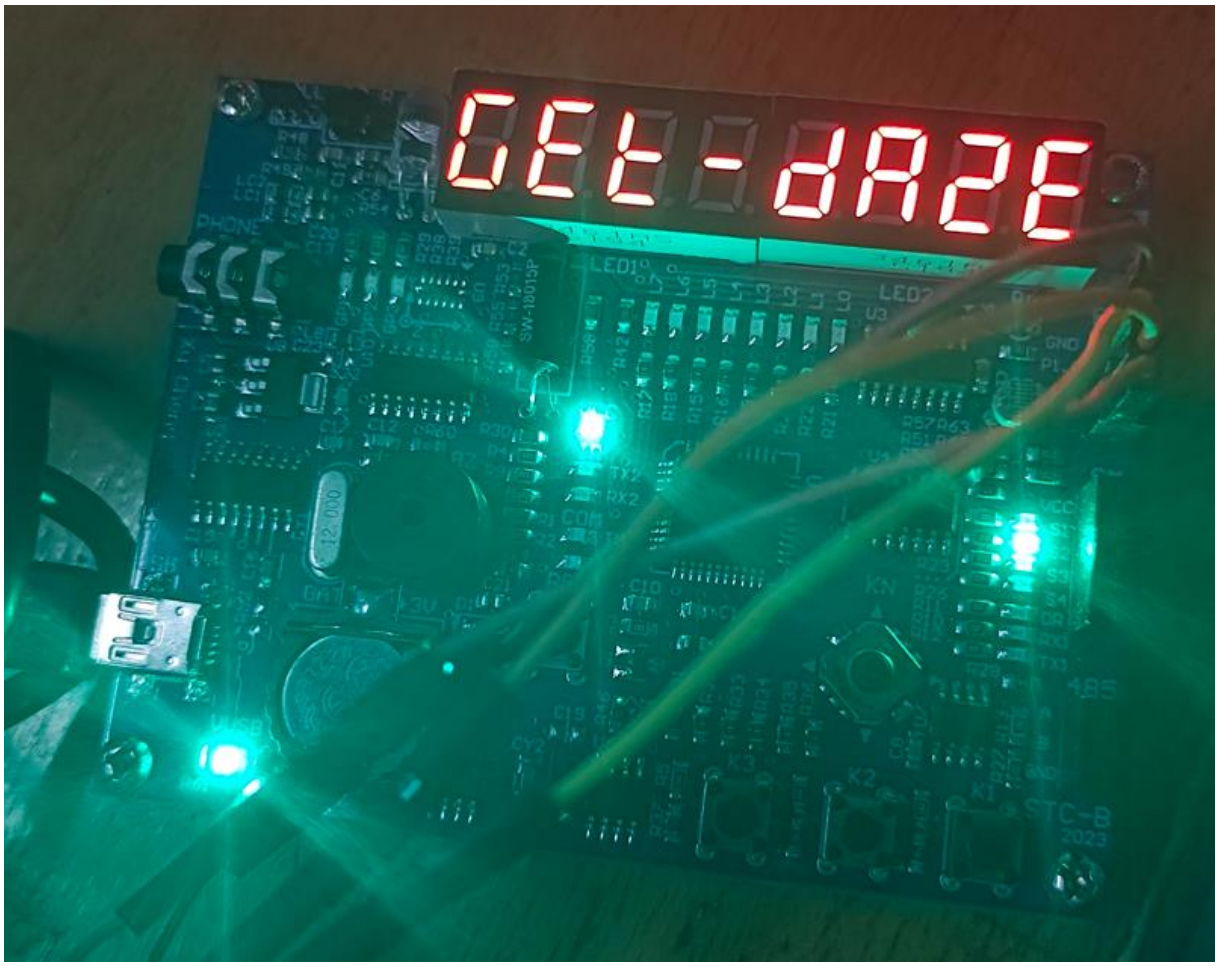
按键拍摄并识别



再次按键进入详细界面

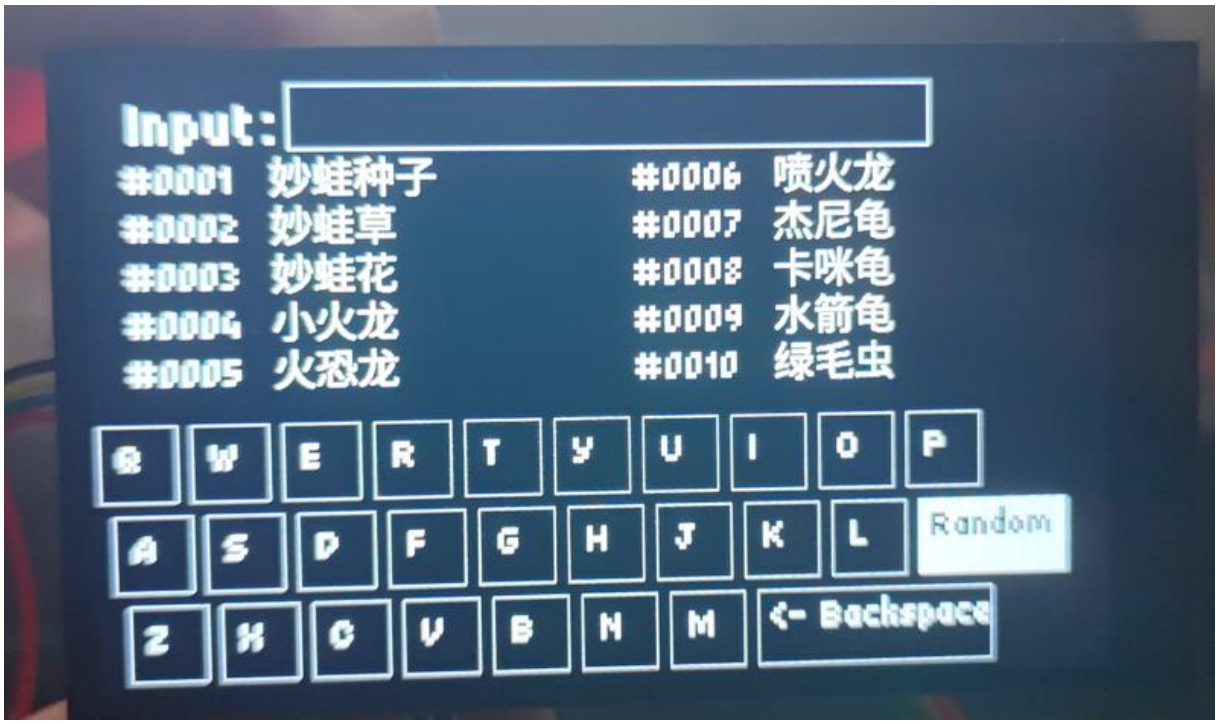


识别成功数码管 LED 都显示，同时蜂鸣器

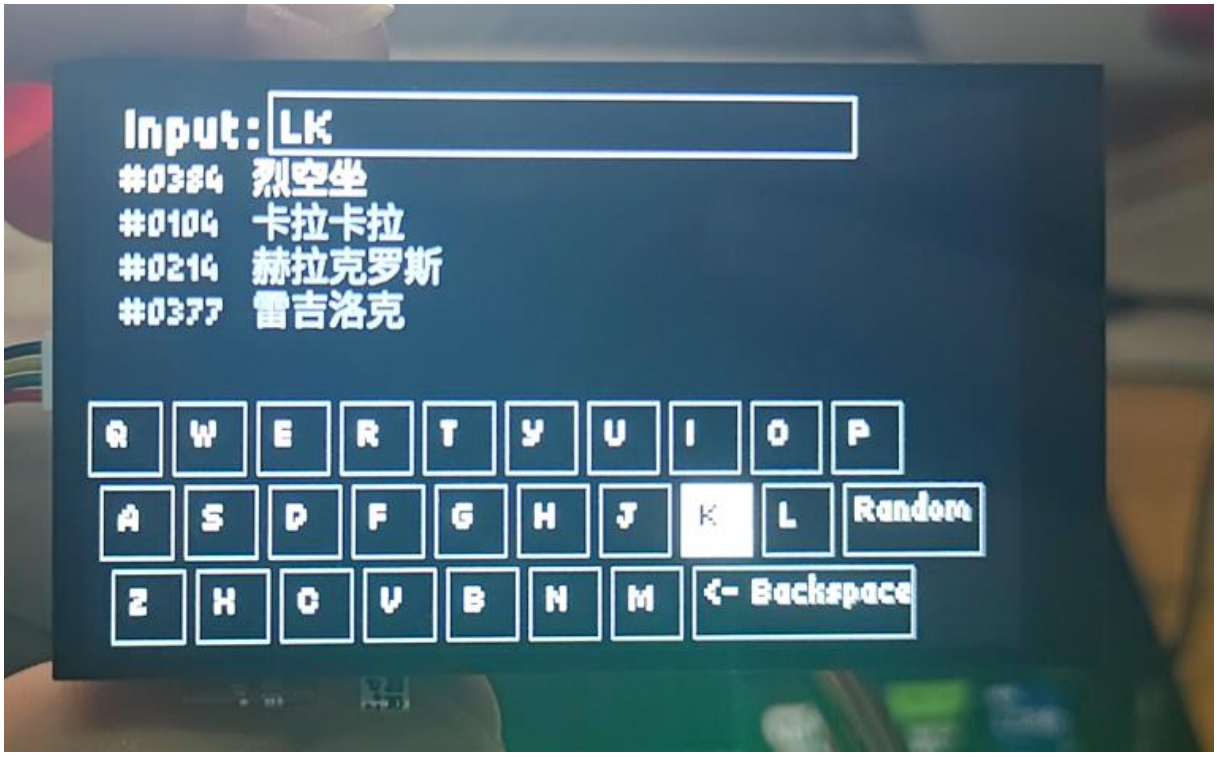


左右切换正常工作

搜索界面

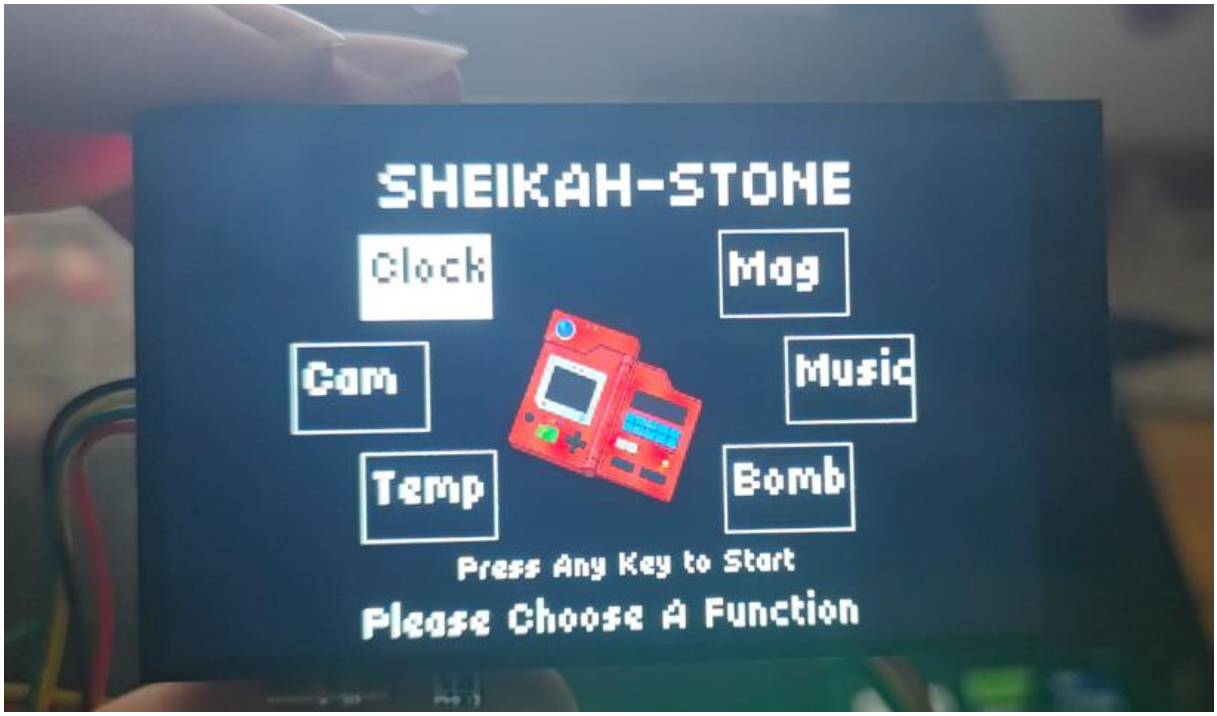


不足 10 个分为精确搜索和模糊搜索，透明度区分



搜索界面正常进入详情

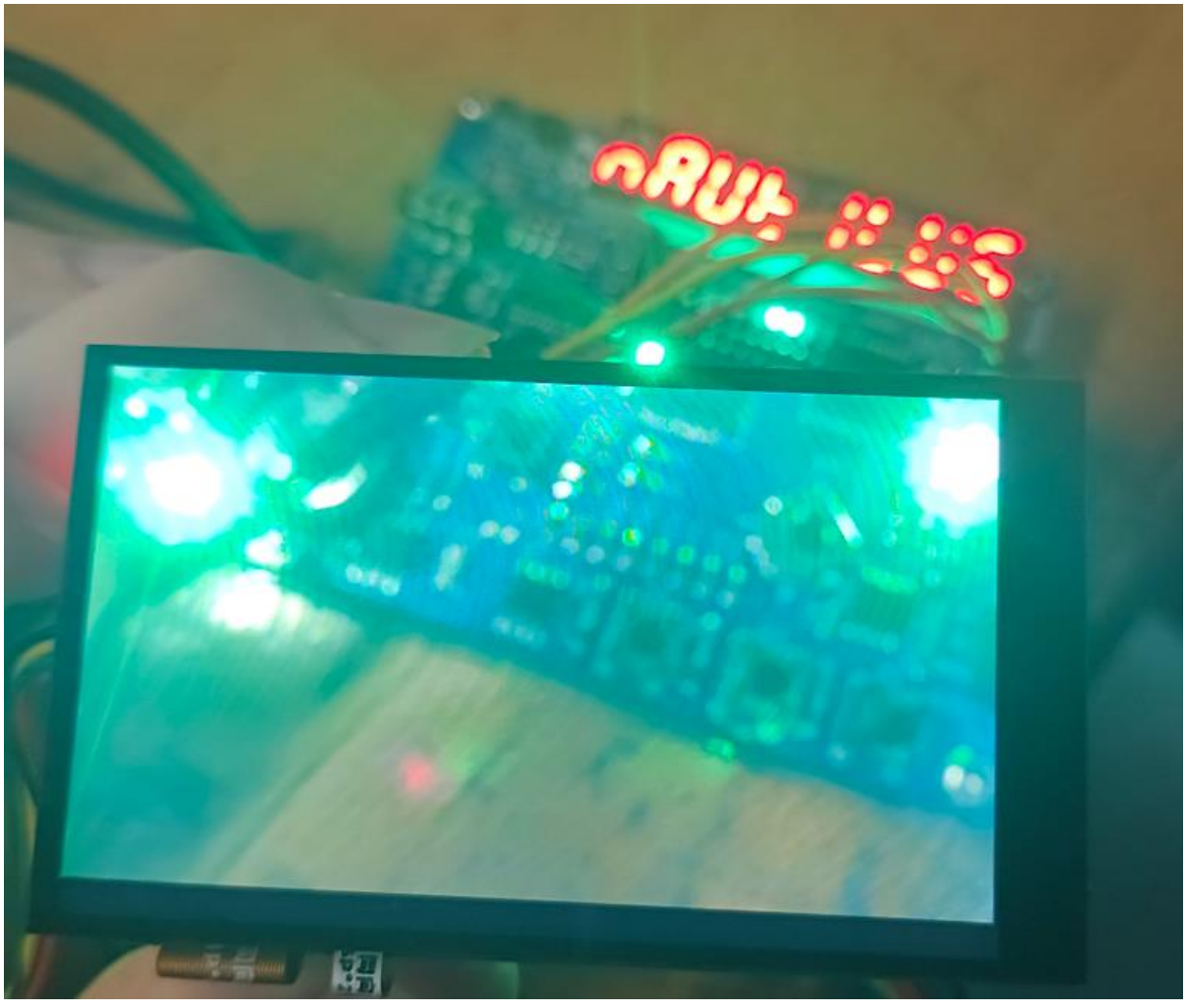
希卡之石界面



RTC 检测，一模一样啊，显示屏多了年月日星期


中国北京时间 UTC+8
16 : 13 : 09
距离长沙市日落18:30还有02小时16分
日出06:14
日落18:30
--现在的时间是--
16:13:10
16-13-10
相关搜索
手机桌面时钟
在线时钟显示到毫秒
北京校准时间到秒表
北京时间显示毫秒

相机模式

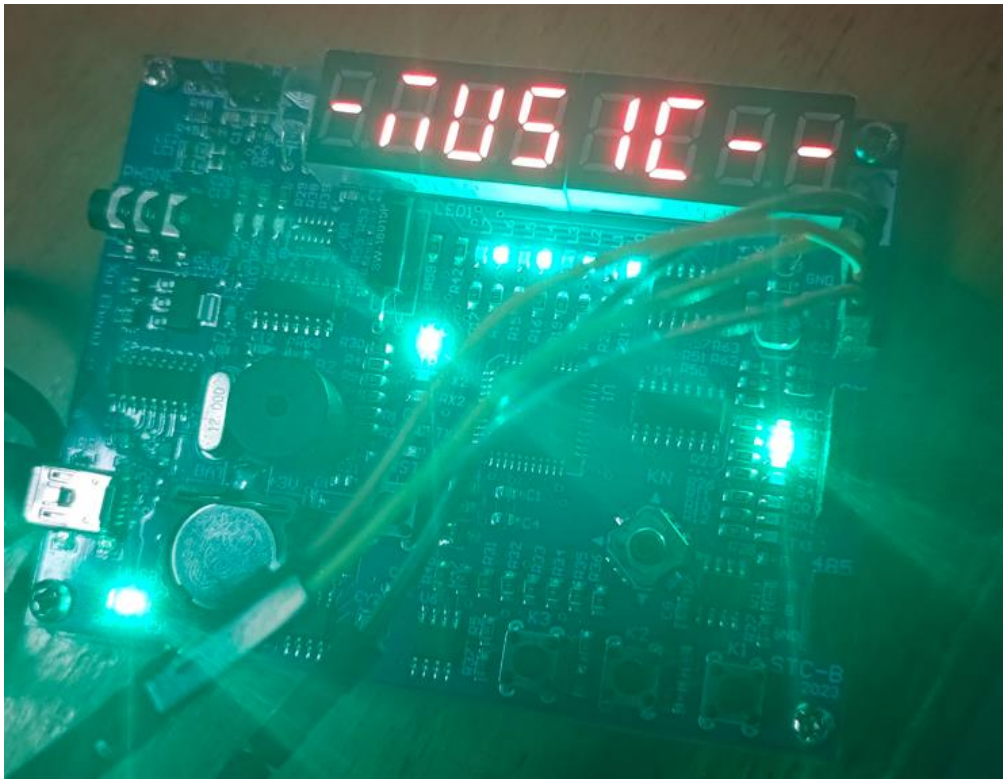


光温模式

磁力检测模式

霍尔 away 事件，靠近拍不了

音乐播放器



6. 总结与展望

项目总结

本项目围绕 “可视化交互集成图鉴终端” 核心目标，基于 STC-B 开发板与 K230-CanMV 开发板的硬件协同，融合 AI 图像识别、硬件传感交互、串口通信等技术，成功实现了 “硬件 + 软件 + AI” 一体化的智能图鉴系统，完成了预期设计任务。

该项目实现了 YOLOv5 分类模型的轻量化部署，在 Windows 环境配置 CUDA 加速训练，将 386 类宝可梦数据集的训练模型通过 NNCase 工具转换为 K230 兼容的 kmodel 格式，识别成功率在正拍、光照充足场景下可达 90% 以上；三是解决了双板通信同步问题，采用 0xaa 0x55 固定包头的串口协议，实现 STC 与 K230 的全双工数据交互，延迟控制在 100ms 以内，确保数码管显示、LED 提示与 K230 界面操作的实时同步。

项目以 “宝可梦图鉴” 与 “希卡之石” 为两大核心功能模块，实现了多维度交互体验：宝可梦图鉴模块支持 “图像识别查询” 与 “手动搜索查询” 双模式，前者通过摄像头采集图像并调用 YOLOv5 模型识别，自动显示宝可梦种族值、特性、身高体重等信息，后者通过虚拟 QWERTY 键盘输入拼音，支持精确匹配与模糊匹配（结果不足 10 个时自动补充），同时详情界面可通过左右按键切换临近宝可梦，并循环播放 BMP 格式动图；希卡之石模块则集成了 RTC 时钟、磁场探测、光温检测、音乐播放等硬件交互功能，如 RTC 模式下 STC 通过 DS1302 模块读取时间，同步在数码管显示 “时 - 分 - 秒”、K230 显示 “年月日星期”，磁场探测模式下通过霍尔传感器触发串口信号，在 LCD 提示 “磁力宝可梦靠近 / 离开”，并联动蜂鸣器报警。此外，项目还实现了 “摇一摇换人”（振动传感器触发随机切换宝可梦）、识别成功提示（LED 闪烁 + 蜂鸣器发声 + 数

码管显示结果）等特色交互功能，提升了系统趣味性。

项目充分发挥 STC-B 与 K230 的硬件特性，形成高效协同架构：STC-B 作为交互执行端，承担按键输入、传感采集、外设控制（数码管、LED、蜂鸣器）等实时性要求高的任务，通过 BSP 库函数快速调用振动、霍尔、DS1302 等模块；K230 作为算力与显示端，依托其 AI 算力实现图像识别，通过媒体库完成复杂 UI 渲染与图像输出，两者通过串口协议实现功能联动，真正实现 “双板合一” 的一体化体验。

项目不足

YOLOv5 分类模型的识别效果受环境与拍摄条件影响较大：外接摄像头采集图像需缩放至 224×224（模型输入尺寸），导致细节丢失，斜拍或小尺寸目标识别成功率降至 60% 以下；且数据集覆盖不足，当前仅包含 386 类宝可梦，且每类样本数量不均衡（部分稀有宝可梦样本少于 20 张），导致模型对小众宝可梦的识别泛化能力较弱；三是光照鲁棒性差，在逆光、暗光环境下，图像对比度降低，模型易将 “深色宝可梦” 误判为同类近似目标。

在硬件交互层面，存在部分功能延迟与操作冗余：触摸控制存在 误触问题，因触摸读取回调函数与界面闪烁循环存在资源竞争，长按触摸时易触发连续响应，需进一步优化定时器周期与触摸状态判断逻辑；在 UI 体验层面，动图播放存在 “帧卡顿” 现象，因 BMP 图像加载需占用较多内存，循环播放时未采用预加载机制，导致帧间隔不稳定，且详情界面的种族值进度条、特性描述等元素布局较为紧凑，在 800×480 分辨率下可读性有待提升。

系统长期运行存在资源管理与兼容性问题： K230 在频繁切换界面或加载图像时，image 对象与模型实例未及时释放，长时间运行可能导致界面

卡死。另外串口通信容错性差，在杜邦线接触不良或电磁干扰场景下，易出现数据丢包（如 STC 发送的时间数据未完整接收，导致 K230 显示日期错误）；三是扩展性局限，当前功能仅支持宝可梦图鉴与希卡之石，未预留模块化接口，新增功能时，需修改核心代码，不利于后续功能迭代。

未来展望

在现有功能基础上，可新增模块：一是手势识别模块，基于 K230 的触摸与媒体库通过触摸输入 “上→右→下→下→下” 等序列，触发对应的功能；二是 “NFC 门禁” 模块，在 STC-B 扩展 NFC 模块，结合希卡之石的 “传导认证信息” 功能，实现贴近 NFC 标签时，STC 数码管显示 “OPEN”；三是 “多图鉴兼容” 模块，扩展数据集，通过主界面切换 “图鉴类型”，实现多场景适配，同时新增 “用户收藏” 功能，通过 M24C02 非易失性存储器保存用户收藏的宝可梦列表，断电后不丢失数据。

资源管理方面，在 K230 代码中添加 gc.collect()主动内存回收，在界面切换前释放 image 对象与模型实例，同时优化串口通信协议，新增 “CRC 校验位” 与 “超时重发” 机制，确保数据传输可靠性。

硬件集成方面，根据 PCB 设计联系厂家定制外壳，将 K230、STC-B、LCD、摄像头、喇叭等模块集成一体，替换杜邦线为板载接口，提升便携性与稳定性，添加 “亮度自动调节”（通过光敏传感器实时调整 LCD 亮度），适配不同使用场景

综上，本项目通过双板协同与 AI 技术的融合，验证了嵌入式系统在 “智能交互图鉴” 场景的可行性，尽管存在部分不足，但为后续嵌入式智能终端开发提供了技术参考。未来通过功能扩展与性能优化，有望将其打造成兼具 “实用性、趣味性、扩展性” 的便携式智能图鉴产品，应用于儿童教育、IP 周边、嵌入式实训等领域。