南京理工大学

NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

计算机科学与工程学院

“嵌入式系统”实验报告书

题目：嵌入式系统作业 3——LCD

学号：922106840127

姓名：刘宇翔

成绩

日期：　　2025 年　　3 月　　14 日

1 题目要求

1. 题目设计要求

【1】作业目标：

（1）熟悉 FSMC 工作原理；

（2）能够熟练掌握 FSMC 嵌入式程序开发；

（3）独自完成 LCD、字库、图形、字模使用方法。

【2】作业内容：

设计程序将自己的中文姓名、学号和证据照显示在 LCD 屏上。

【3】完成要求：

（1）工程名称命名为：学号_lcd;

（2）学号 每隔 1 分钟自动换一个颜色

（3）中文姓名显示采用跑马形式形式，从左向右移动，正好 5 秒移动完后，从左再此循环显示；

（4）中文姓名不得用图片形式，须由字模生成；

（5）程序代码附有必要的注释，在开发板上完成验证。

2. 拟实现的具体功能

完成基于 FSMC 开发的，通过使用相关工具独自完成 LCD、字库、图形、字模使用方法，编程实现本实验任务，编译通过后下载程序到开发板上，能够成功在 LCD 上展现个人的学号、姓名以及证件照的内容，完成实验要求。

2 总体设计

2.1 硬件设计

1. 硬件设计思路

本次实验使用的是 RT-Tread Studio 为基础的工程模板，主要使用了 LVGL 库的头文件以及相关函数用于完成图像的显示功能。

MCU 的 LCD 驱动程序与 LVGL 库接口对接，实现图形缓冲区数据传输。通过 RT-Thread 提供的多线程机制，分配独立任务处理跑马灯动画与学号颜色切换，保证实时性和响应性。

通过 GPIO 和定时器确保各显示模块与 MCU 之间的信号控制，避免因数据传输延时造成显示异常。

在硬件设计中，我也充分考虑了调试信号和测试点布局，便于在开发过程中进行电路和信号质量检查，确保系统各模块正常工作。

本硬件设计思路旨在搭建一个以 MCU 为核心、LCD 为显示终端的嵌入式系统，通过 RT-Thread 操作系统和 LVGL 库，实现高效、实时的图形显示效果。设计过程中需兼顾性能、稳定性以及扩展性，确保系统在运行动画和颜色变换等动态效果时保持流畅、可靠的显示效果。该设计不仅满足实验要求，还为后续功能扩展预留了充足的接口和硬件资源。

2.2 软件设计

1. 系统总体设计

本程序基于 RT-Thread 实时操作系统和 LVGL 图形库，主要运行在嵌入式开发板上。程序的目标功能有两个：

（1）学号颜色自动切换：学号文本每隔 1 分钟更换一次显示颜色。

（2）中文姓名跑马灯效果：姓名文本以跑马灯的形式从屏幕左侧平滑移动至右侧，完成一次移动周期为 5 秒，移动结束后立即从左侧重新开始。

（3）在 LCD 屏幕上展示我的证件照图像。

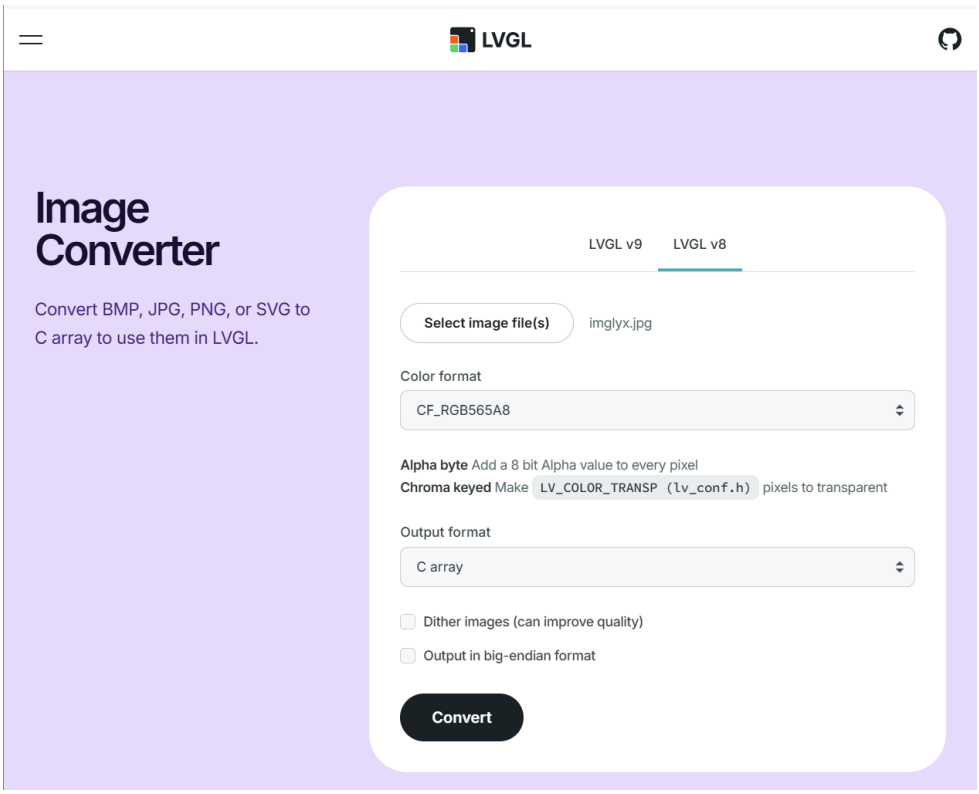
整个程序采用多线程设计，使用两个独立的线程分别处理这两个功能，从而互不干扰，保证界面动态效果流畅运行。

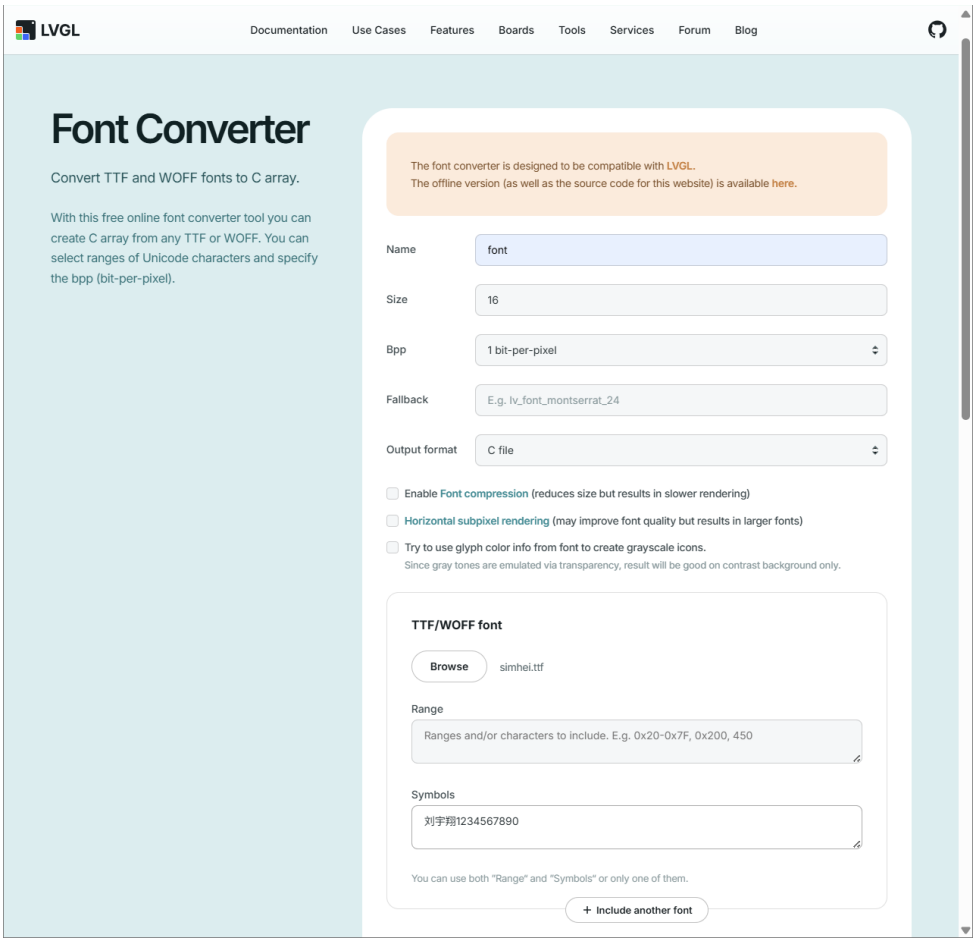
2. GUI 初始化与资源配置

（1）字体和图片资源加载

使用 LV_FONT_DECLARE(font) 和 LV_IMG_DECLARE(imglyx) 导入字体和图片资源，其中中文姓名和学号文本均依赖于该字体生成，不采用图片显示，从而满足作业要求。

我将我需要使用的证件照图片以及字体“simhei.tff”上传至 LVGL Image Converter 以及 LVGL Font Converter 生成对应的图模与字模文件，相关条件配置如下图所示：


LVGL
Image Converter
Convert BMP, JPG, PNG, or SVG to C array to use them in LVGL.
LVGL v9
LVGL v8
Select image file(s)
imglyx.jpg
Color format
CF_RGB565A8
Alpha byte Add a 8 bit Alpha value to every pixel
Chroma keyed Make LV_COLOR_TRANSP (lv_conf.h) pixels to transparent
Output format
C array
Dither images (can improve quality)
Output in big-endian format
Convert


LVGL
Documentation
Use Cases
Features
Boards
Tools
Services
Forum
Blog
Font Converter
Convert TTF and WOFF fonts to C array.
With this free online font converter tool you can create C array from any TTF or WOFF. You can select ranges of Unicode characters and specify the bpp (bit-per-pixel).
The font converter is designed to be compatible with LVGL.
The offline version (as well as the source code for this website) is available here.
Name
font
Size
16
Bpp
1 bit-per-pixel
Fallback
E.g. lv_font_montserrat_24
Output format
C file
Enable Font compression (reduces size but results in slower rendering)
Horizontal subpixel rendering (may improve font quality but results in larger fonts)
Try to use glyph color info from font to create grayscale icons.
Since gray tones are emulated via transparency, result will be good on contrast background only.
TTF/WOFF font
Browse
simhei.ttf
Range
Ranges and/or characters to include. E.g. 0x20-0x7F, 0x200, 450
Symbols
刘宇翔1234567890
You can use both "Range" and "Symbols" or only one of them.
+ Include another font

（2）创建界面元素

在 lv_user_gui_init 函数中，利用 LVGL 库在当前活动屏幕上创建了三个 UI 对象：

姓名标签：通过 lv_label_create 创建标签，并用 lv_obj_set_style_text_font 设置字体，lv_label_set_text 指定显示的中文姓名。初始位置设定在屏幕顶部较靠上的位置（如 (0, 10)）。

学号标签：类似方式创建，用于显示学号字符串，位置相对于姓名标签下移一点（例如 (0, 40)）。

图片对象：位置设定在屏幕中部偏下。

（3）线程创建与启动 在同一函数中，创建了两个线程：

名称为“rolling”的线程，其入口函数是 rolling_thread_entry，负责实现姓名跑马灯效果。该线程接收姓名标签对象作为参数。

名称为“changing”的线程，其入口函数是 changing_thread_entry，负责处理学号颜色的周期性变换。线程接收学号标签对象作为参数。 每个线程都分配了 1024 字节的堆栈、设定了优先级和时间片参数，确保系统调度时各线程能按预定时间进行处理。

3. 姓名跑马灯效果的实现

（1）运动轨迹设计 在 rolling_thread_entry 中，先定义了两个位置变量：

start_pos：起始位置，设定为负值（例如 -60），使得标签初始时完全位于屏幕左侧之外；

end_pos：结束位置，设定为一个大于屏幕宽度的值（例如 240），确保标签完全移出右侧。

（2）动画实现方式 采用一个无限循环来不断更新姓名标签的位置：

在每个循环内，通过一个 for 循环，每次将标签的 x 坐标增加 1 个像素，从起始位置逐步移向结束位置。

每一步之间加入延时，利用 HAL_Delay(interval) 控制刷新间隔。延时时间由整个移动周期 5 秒分成固定步数（代码中大致分为 300 步，每步间隔大约 16～20 毫秒），保证运动平滑且总时间为 5 秒。

当标签到达结束位置后，通过 lv_obj_set_x 将位置重置回 start_pos，从而形

成循环滚动的无缝动画效果。

4. 学号颜色定时变换的实现

（1）颜色数组的定义

在 changing_thread_entry 中，先定义了一个包含六种颜色的数组，颜色通过 lv_color_make 函数生成，包含黑色、红色、绿色、蓝色、黄色和紫色。

（2）定时切换逻辑

线程进入无限循环，在每个周期中，使用 lv_obj_set_style_text_color 将学号标签的文本颜色设置为数组中当前索引所对应的颜色。

随后通过简单的算术运算 (color_index + 1) % color_count 更新颜色索引，实现循环使用数组内的颜色。

最后调用 rt_thread_mdelay(60000) 延时 60 秒，使得颜色更换的周期为 1 分钟。

5. 线程调度与实时性保障

（1）线程优先级与时间片

代码中创建线程时均设置了相同的优先级，这保证了两个线程能获得 CPU 时间而不会互相抢占过多资源，同时时间片参数的设置使得 RT-Thread 能合理分配执行时间。

（2）延时函数的使用

在滚动线程中使用 HAL_Delay 控制动画帧率，在颜色切换线程中使用 rt_thread_mdelay 控制周期性任务。两种延时函数分别适用于硬件级别的延时和 RTOS 任务延时，保证了系统的实时响应和调度稳定性。

6. 代码模块化与可维护性

（1）功能分离

将姓名滚动和学号颜色变换分别放在独立线程中，有利于未来扩展和维护。每个线程只关注自己的任务，逻辑清晰，减少了代码间的耦合。

（2）注释和日志

在代码中添加了必要的注释，解释每个变量和逻辑的含义，便于开发人员和后续维护者快速理解程序运行流程。线程创建时也有错误日志输出，方便调试线程初始化失败的情况。

（3）资源管理

使用 LVGL 提供的 API 进行对象创建和样式设置，确保在图形显示方面调用的是稳定的底层接口。依赖 RT-Thread 的线程管理机制来保证多任务环境下的协作调度。

7．软件流程图：流程阶段分解

A. 初始阶段

开始 → 系统初始化

程序启动后首先执行系统及硬件的初始化。包括：

【1】RT-Thread 内核初始化、任务调度与系统资源配置。

【2】硬件外设初始化，如 LCD 显示模块、GPIO、串口等（如果有需要）。

B. GUI 初始化阶段

调用 GUI 初始化函数 → lv_user_gui_init()

进入 GUI 初始化阶段后，程序完成以下操作：

创建显示对象

***姓名标签*：**调用 lv_label_create 创建姓名显示对象，并用 lv_obj_set_style_text_font 设置自定义字体；初始位置设置为屏幕上方（如 (0, 10)）。

***学号标签*：**同样方式创建学号显示对象，设置字体和初始位置（如 (0, 40)）。

***图片对象*：**调用 lv_img_create 创建图片显示对象，加载资源并设置显示位置（如 (51, 60)）。

线程创建与启动

创建**“滚动线程”**，其入口函数为 rolling_thread_entry，负责控制姓名标签的跑马灯效果。

创建**“颜色切换线程”**，其入口函数为 changing_thread_entry，负责每隔 1 分钟切换学号文本颜色。

C. 姓名跑马灯线程处理（滚动线程）

进入滚动线程 → rolling_thread_entry()

具体流程如下：

初始位置设置

设置姓名标签的起始位置为屏幕左侧之外（例如 start_pos = -60）。

定义结束位置为屏幕右侧之外（例如 end_pos = 240）。

运动动画循环

采用 for 循环，每次将标签的 x 坐标增加 1 像素。

每次更新后调用延时函数（如 HAL_Delay(interval)，其中 interval 大约为 20 毫秒），确保整个运动周期为 5 秒（大致 300 步）。

重置与循环

当姓名标签移动到结束位置后，立即将位置重置回起始点，实现无缝循环滚动。

D. 学号颜色切换线程处理（颜色切换线程）

进入颜色切换线程 → changing_thread_entry()

具体流程如下：

颜色数组定义

定义包含六种颜色（黑、红、绿、蓝、黄、紫）的数组，用于轮换显示学号文本颜色。

颜色切换循环

在无限循环中，首先调用 lv_obj_set_style_text_color 将当前颜色应用到学号标签。然后调用延时函数 rt_thread_mdelay(60 * 1000)实现 60 秒延时。

延时结束后，更新颜色索引（通过循环索引实现数组内颜色轮换），继续下一次颜色切换。

E. 循环机制与实时响应

多线程独立运行

两个线程（滚动线程和颜色切换线程）并发运行，各自完成动画与颜色变换任务，不会互相阻塞。

系统保持实时响应，确保 LCD 上始终呈现平滑的跑马灯效果和定时颜色切换。

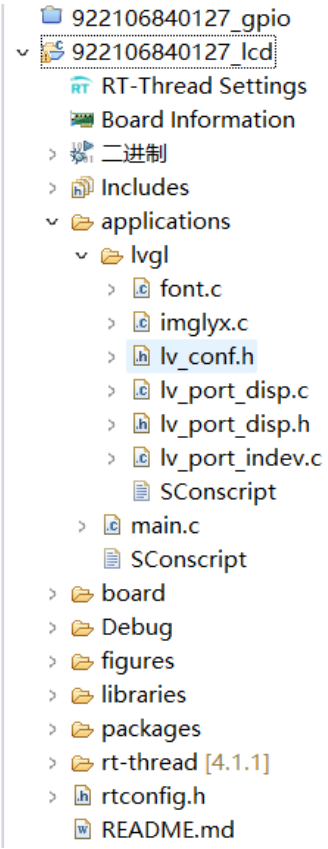
F. 结束阶段

系统退出或复位

当系统接收到关闭或复位信号时，各线程结束任务，系统资源释放，LCD 显示停止更新。

8. 具体功能设计与代码实现

（1）项目文件结构

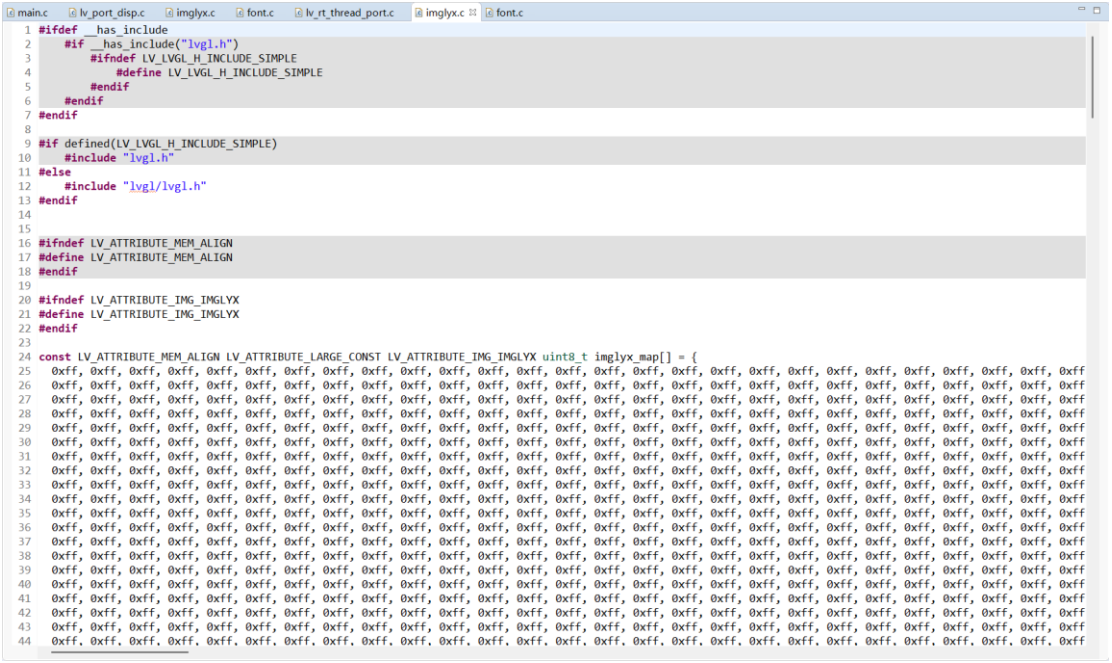


（2）font.c 以及 imglyx.c

功能：用于存放字体以及证件照导入生成的字模以及图模，由于代码内容过长故而截图一部分如下，具体就是给相关的内容进行编码用于 LCD 展示。

【1】 font.c

【2】 imglyx.c



9. main.c 代码实现

```
/*
 * Copyright (c) 2006-2021, RT-Thread Development Team
 *
 * SPDX-License-Identifier: Apache-2.0
 *
 * Change Logs:
 * Date           Author       Notes
 * 2023-5-10      ShiHao       first version
 */

#include <rtthread.h>
#include <rtdevice.h>
#include <board.h>

#define DBG_TAG "main"
#define DBG_LVL         DBG_LOG
#include <rtdbg.h>
```

```
#include <lvgl.h>

/* 导入字体和图片资源 */
LV_FONT_DECLARE(font)
LV_IMG_DECLARE(imglyx)

/* 线程函数声明 */
void rolling_thread_entry(void *ptr);
void changing_thread_entry(void *ptr);

/* GUI 初始化函数 */
void lv_user_gui_init(void)
{
    /* 创建姓名标签 */
    lv_obj_t *obj = lv_label_create(lv_scr_act());
    lv_obj_set_style_text_font(obj, &font, 0);
    lv_label_set_text(obj, "刘宇翔");
    lv_obj_set_pos(obj, 0, 10);

    /* 创建学号标签 */
    lv_obj_t *obj2 = lv_label_create(lv_scr_act());
    lv_obj_set_style_text_font(obj2, &font, 0);
    lv_label_set_text(obj2, "922106840127");
    lv_obj_set_pos(obj2, 0, 40);

    /* 创建图片 */
    lv_obj_t *obj3 = lv_img_create(lv_scr_act());
    lv_img_set_src(obj3, &imglyx);
    lv_obj_set_pos(obj3, 51, 60);
```

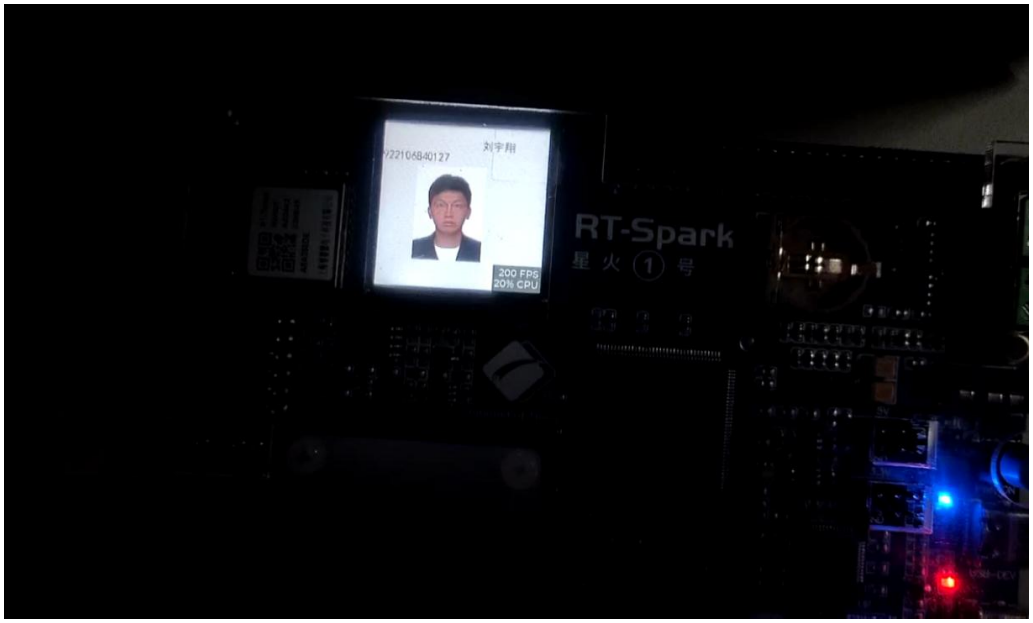
```
    /* 创建并启动滚动线程 */
    rt_thread_t thread = rt_thread_create("rolling", rolling_thread_entry, obj, 1024, 1,
5);
    if (thread != RT_NULL)
    {
        rt_thread_startup(thread);
    }
    else
    {
        LOG_E("Failed to create rolling thread!");
    }

    /* 创建并启动颜色变换线程 */
    rt_thread_t thread2 = rt_thread_create("changing", changing_thread_entry, obj2,
1024, 1, 5);
    if (thread2 != RT_NULL)
    {
        rt_thread_startup(thread2);
    }
    else
    {
        LOG_E("Failed to create changing thread!");
    }
}

/* 姓名滚动线程 */
void rolling_thread_entry(void *ptr)
{
    lv_obj_t* obj = (lv_obj_t*) ptr;
```

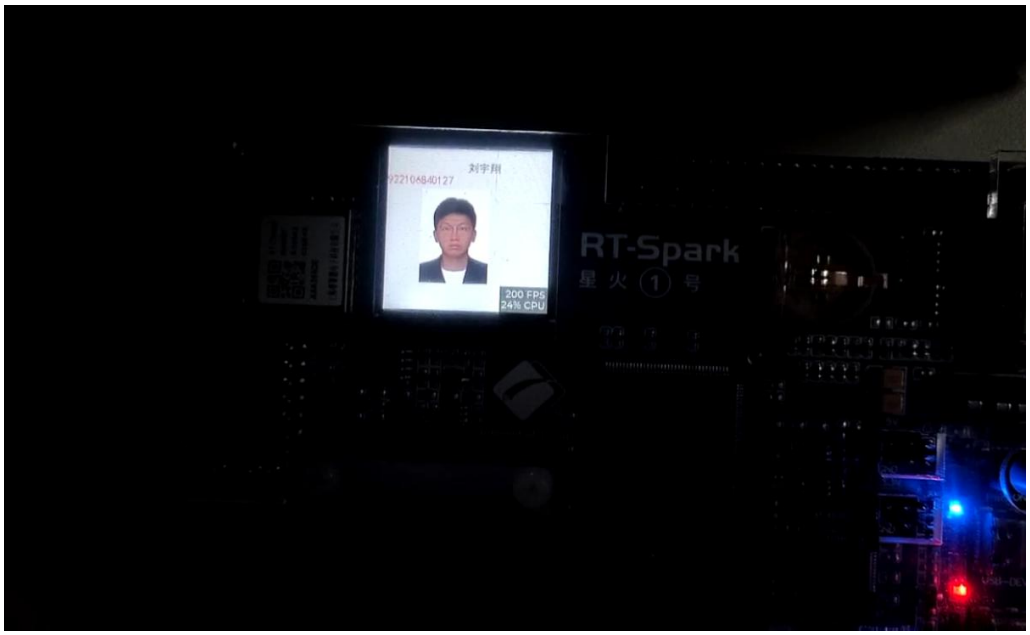
```
    int32_t start_pos = -60;     // 起始位置（完全左出）
    int32_t end_pos = 240;       // 结束位置（完全右出）
    const int interval = 5000 / 300; // 5 秒内移动 300 步，每步间隔 20ms

    while (1)
    {
        for (int16_t x = start_pos; x <= end_pos; x += 1)
        {
            lv_obj_set_pos(obj, x, 20);
            HAL_Delay(interval);
        }
        lv_obj_set_x(obj, start_pos); // 重置回左侧
    }
}

/* 颜色定义 */
#define COLOR_BLACK     lv_color_make(0x00, 0x00, 0x00)
#define COLOR_RED       lv_color_make(0xFF, 0x00, 0x00)
#define COLOR_GREEN     lv_color_make(0x00, 0xFF, 0x00)
#define COLOR_BLUE      lv_color_make(0x00, 0x00, 0xFF)
#define COLOR_YELLOW    lv_color_make(0xFF, 0xFF, 0x00)
#define COLOR_PURPLE    lv_color_make(0x80, 0x00, 0x80)

/* 学号颜色变换线程 */
void changing_thread_entry(void *ptr)
{
    lv_obj_t* obj = (lv_obj_t*) ptr;
    const lv_color_t colors[] = { COLOR_BLACK, COLOR_RED, COLOR_GREEN,
COLOR_BLUE, COLOR_YELLOW, COLOR_PURPLE };
    uint8_t color_index = 0;
```

```
    const uint16_t color_count = sizeof(colors) / sizeof(colors[0]);


    while (1)
    {
        lv_obj_set_style_text_color(obj, colors[color_index], LV_PART_MAIN); //
设置颜色
        color_index = (color_index + 1) % color_count; //  更新索引
        rt_thread_mdelay(60 * 1000); // 60 秒延时
    }
}
/*  主函数  */
int main(void)
{
    return 0;
}
```

3 实验结果分析与总结

以下是我实验结果的过程截图，可以看到学号的颜色变色了，由于 60 秒的视频占据存储空间过大，从而只截取了切换字体颜色的几秒钟作为实验视频验证。




刘宇翔
200 FPS
24% CPU
RT-Spark
星火①号