东南大学生产实习

乐鑫实习——项目报告

基于 ESP32-S2 的人机交互界面：拓展型 2048 小游戏

院（系）：　信息科学与工程学院

小组成员：　梁宇辰 04017414

谯　兵 04017411

祁栋华 04017417

完成日期：　2020 年 9 月 23 日

指导人：　詹昭城

目录

一、项目背景

1、乐鑫科技简介

乐鑫科技（www.espressif.com）成立于 2008 年，是一家专注于物联网和人工智能领域的无线通讯芯片及方案研发的上市公司，在全球多个地区设立了销售与研发中心，总部位于中国上海，为客户提供芯片、硬件、软件、APP、云端的一站式物联网和人工智能解决方案。其中，最受全球欢迎的有 ESP8266 和 ESP32 系列芯片。 乐鑫现已成为物联网领域的主流供应商，客户遍及全球物联网的各种应用领域，从消费级到工业级，从小型创业公司到大型老牌公司。近年来发展迅猛：

- 2015 年乐鑫被评为 “2015 年亚洲科技创新 100 强”；
- 2016 年乐鑫发布旗舰级产品 ESP 32 系列芯片并获复星集团领投 B 轮亿元级融资；
- 同年乐鑫荣膺 Gartner 2016 物联网 “最酷供应商” (Cool Vendors) 称号；
- 2017 年和 2018 年连续两年在 IoT 的 Embedded MCU Wi-Fi 领域出货量排名全球第一；
- 2018 年乐鑫完成 C 轮融资，由英特尔投资与芯动能基金联合领投；
- 2019 年 7 月 Github 上关于乐鑫芯片产品的开源项目超过 30000 个；
- 2019 年 7 月乐鑫科技（688018）成为科创板首发上市企业。

2、ESP32-S2 简介：

乐鑫的 ESP32-S2 系列开发板有丰富的外设接口和功能模块，方便用户快速构建原型，满足用户开发物联网应用的需求。

ESP32-S2 系列模组是通用的 Wi-Fi MCU 模组，功能强大，具有丰富的外设接口，是物联网、可穿戴电子设备和智能家居等应用场景的理想选择。

3、人机交互界面：

人机交互界面（HMI，Human Machine Interface）是系统和用户之间进行交互和信息交换的媒介，它实现信息的内部形式与人类可以接受形式之间的转换。凡参与人机信息交流的领域都存在着人机交互界面，2048 的游戏界面即是一个典型的 HMI 人机交互界面。

人机界面产品由硬件和软件两部分组成，硬件部分包括处理器、显示单元、输入单元、通讯接口、数据存贮单元等。在本项目中，ESP32-S2 开发板承担了其中大部分的职能。

4、2048 简介：

2048 是一款比较流行的数字游戏，最早于 2014 年 3 月 20 日发行。原版 2048 首先在 GitHub 上发布，原作者是 Gabriele Cirulli，后被移植到各个平台。

玩家每次可以选择上下左右其中一个方向去滑动，每滑动一次，所有的数字方块都会往滑动的方向靠拢外，系统也会在空白的地方随机出现一个数字方块，相同数字的方块在靠拢、相撞时会相加。不断的叠加最终拼凑出 2048 这个数字就算成功。

市面上的大多数 2048 游戏功能相当简单，只具有基本功能，玩法和难度的固定化使得其不能够适应不同年龄段，不同智力水平的玩家。对此，我们将尝试进行功能的优化和拓展。

二、技术路线

1、C 语言实现游戏算法

2048 规则容易理解，核心算法的描述也比较简单。我们将采取对二维数组的一维表示进行操作的方式进行 C 语言的编程，尽量地参照 C++、C#等编程语言面向对象的特点，实现算法的低耦合、高内聚。

2、ESP32-S2-LCD-DevKit

项目将使用如图 2-1 所示开发板进行开发.在 Github 上有针对开发板的专门的例程，我们将对其展开学习，并在此基础上搭建自己的工程所需的代码编译环境，最终能够跑出自己编写的能够达到项目目的的代码。

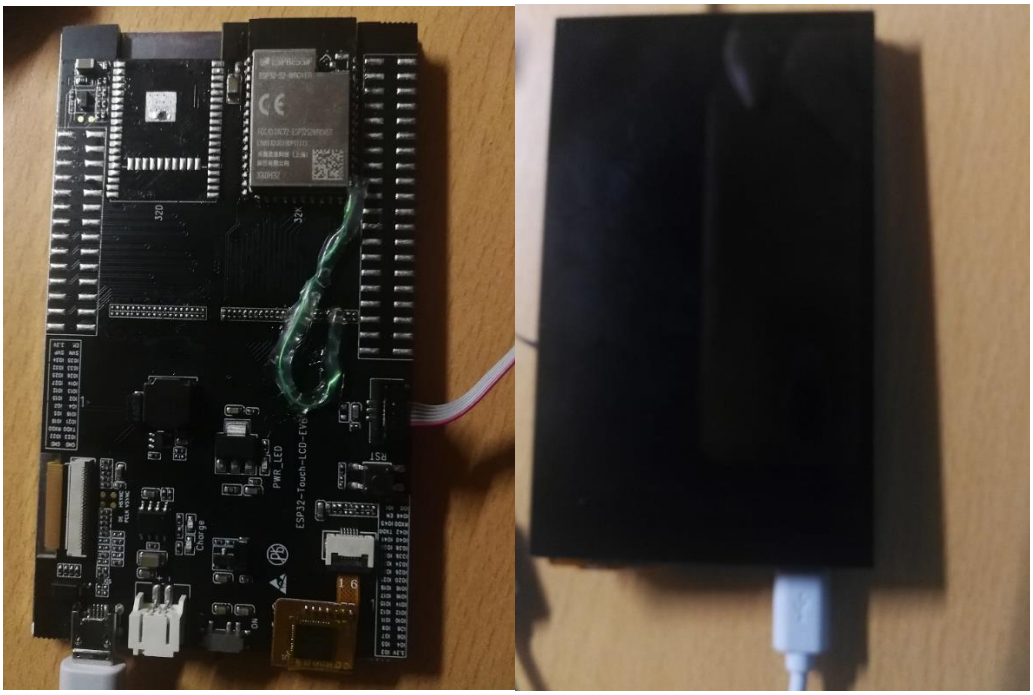

图 2-1 ESP32-S2-LCD-DevKit 开发板

3、LVGL

LVGL（Light and Versatile Graphics Library）是一个免费的开源图形库，它提供创建具有易于使用的图形元素，精美的视觉效果和低内存占用的嵌入式 GUI 所需的一切。

主要特点：

（1）功能强大的构建块，例如按钮，图表，列表，滑块，图像等。
（2）带有动画，抗锯齿，不透明度，平滑滚动的高级图形
（3）各种输入设备，例如触摸板，鼠标，键盘，编码器等
（4）支持 UTF-8 编码的多语言
（5）多显示器支持，即同时使用更多的 TFT，单色显示器
（6）完全可定制的图形元素
（7）独立于任何微控制器或显示器使用的硬件
（8）可扩展以使用很少的内存（64 kB 闪存，16 kB RAM）进行操作
（9）操作系统，支持外部存储器和 GPU，但不是必需的
（10）单帧缓冲区操作，即使具有高级图形效果

（11）用 C 语言编写，以实现最大的兼容性（与 C ++兼容）
（12）模拟器可在没有嵌入式硬件的 PC 上启动嵌入式 GUI 设计
（13）绑定到 MicroPython
（14）快速 GUI 设计的教程，示例，主题
（15）提供在线和离线文档
（16）在 MIT 许可下免费和开源

我们将根据 Github 上的例程和 lvgl.io 官网上的文档进行充分的实践性学习，充分发挥主观能动性，促进更深入的实践。通过基于 VS 的 PC 端模拟器，可以更方便地进行代码的编写和调试，如图 2-2 所示。

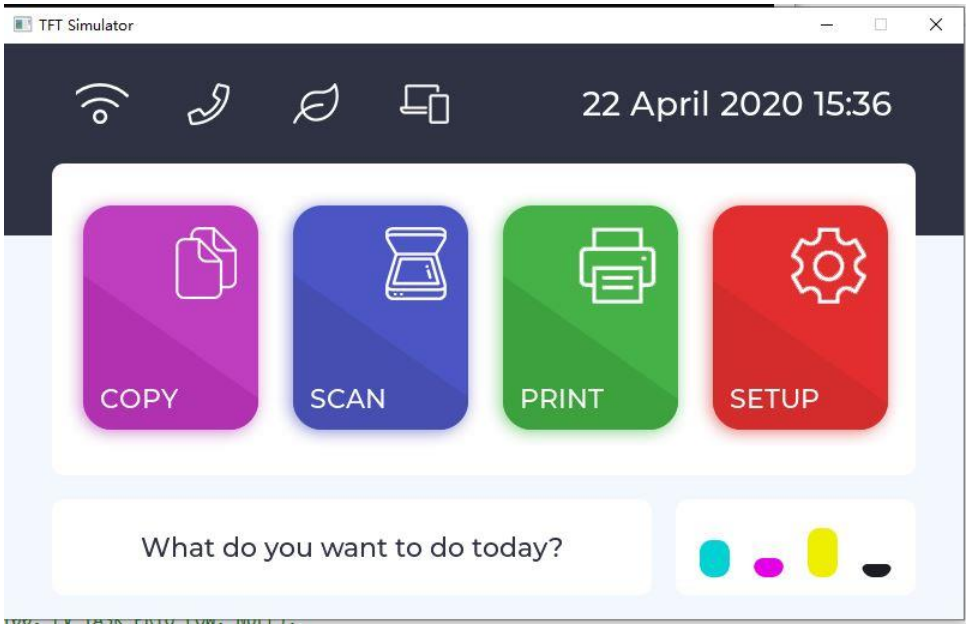

图 2-2 LVGL 模拟器调试界面

三、前期准备

1、环境搭建

下载最新版本 esp-idf 安装器 2.3，按照提示一步一步进行安装。选择电脑上本已安装的 git2.21.0 和 python3.7 版本，并将其添加到系统路径中。并在最后一步取消如图 3-1 所示选项（否则会失败）。

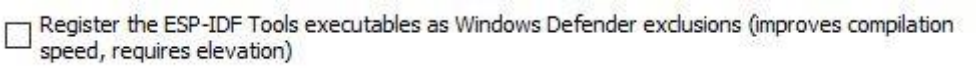

图 3-1 该选项需取消勾选

安装完成后，试运行，Build 到大约 5/908 时失败，日志报缺少编译文件（未截图，日志已缺失）。尝试手动下载 cmake，下载完成后再次尝试，Build 成功。

2、例程测试

（1）安装完成后，打开新创建的 ESP-IDF Command Prompt，自动完成 Esp 工具的路径添加。随后进入例程的工程目录：cd examples/get-started/hello_world。

（2）运行指令：idf.py set-target esp32s2，将 Build 的目标平台从默认的 ESP32 设置到我们所需要的 ESP32S2。

运行指令：idf.py menuconfig，进行图形化界面中的配置，选择所使用开发板和 lvgl 所需配置如图 3-2 所示。

```
#1) Serial flasher config --> Flash size --> 4MB
#2) Partition Table --> Partition Table --> Custom partition table CSV
#3) Component config --> ESP32S2-specific --> SUpport for external, SPI-connected RAM
```

图 3-2 menuconfig 配置

（3）运行指令：idf.py build，进行 Build，解决 3.1 中提到的编译问题后成功 Build，生成烧录所需的 bin 文件。

（4）安装 FDI 驱动，使用 USB2.0 数据线连接 ESP prog 编程板和 PC 端，并用合适的 JTAG 排线连接将编程板和 ESP32S2 开发板相连接，最后使用任意 USB 线和开发板相连进行供电，具体的接线方式如图 3-3 所示。

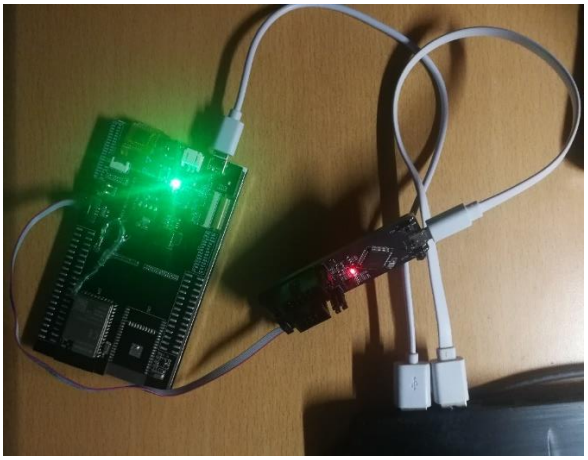

图 3-3 接线方式

（5）成功接线并安装驱动后，PC 端的设备管理器的端口一栏会出现两个新的设备号，如图 3-4 所示，根据编程板上连接主板的 JTAG 接口的描述 com n+1，我们认为这两个设备号中较大的数字就是我们需要的端口号，即图 3-4 中 COM4。

图 3-4 设备管理器界面，新端口 COM3，COM4

（6）运行指令：idf.py -p COM4 flash，程序开始烧写，此时必须将屏幕的开关打开，否则将会连接超时，如上述步骤均无误，将会提示 flash 成功。

（7）运行指令：idf.py monitor，可打开监视器，发现成功输出“HELLO WORLD”，测试成功，按下 ctrl+]，退出监视器。

3、基本功能分析

分析 2048 所需具备的基本功能如下：

（1） 刷新显示
（2） 上下左右移动
（3） 判定边界
（4） 判定游戏结束
（5） 碰撞计算
（6） 分数结算

4、多样化功能

在基本功能之外，我们将对 2048 进行一些功能的拓展，拓展功能大致包括：

（1） 难度分级
（2） 登陆模式
（3） 自定义模式（维度等）
（4） 作弊功能
（5） 排行榜
（6） 护眼模式

5、控制台算法验证

利用 C#语言在控制台中实现了 2048 游戏的基础模型，如图 3-5 所示。具体的实现方法将在下文呈现。

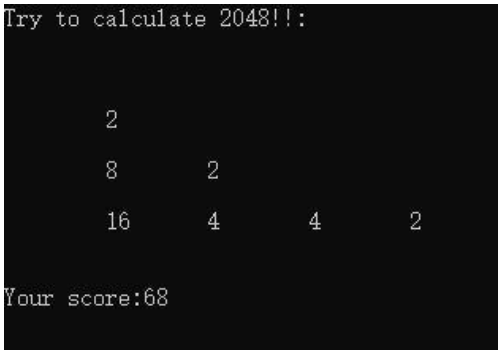

图 3-5 控制台验证 2048 实现算法

四、项目成果

1、2048 基础实现算法

我们决定首先在控制台中实现 2048 的基础功能，然后移植到 lvgl 的界面中，最终完成的代码写入编译烧录的 ESP32S2 程序框架中，并进行一些必要的修改。

要在控制台实现 2048 的基础功能，考虑到基于 lvgl 编程的一些面向对象的特性，我们决定使用相比 C、C++更容易的 C#语言进行基于控制台的算法编写，快速实现后再将代码转换为 lvgl 支持的 C、C++语言。

针对不同的功能，我们分别采用不同的静态函数来实现。

（1）总体

和 C、C++语言不同，C#语言支持数组的长度为可变量，于是我们设置全局变量 dimension 来表示方阵的维度，dimension 的值可变。方阵中的方块都被映射到一个全局的二维数组 gameArray 中，该整型二维数组的大小即为 dimension * dimension。gameArray[dimension][dimension]中存储每个方块对应的数字。下文将要提到的许多操作都将是对于这个二维数组的操作。整体流程图如图 4-1 所示。

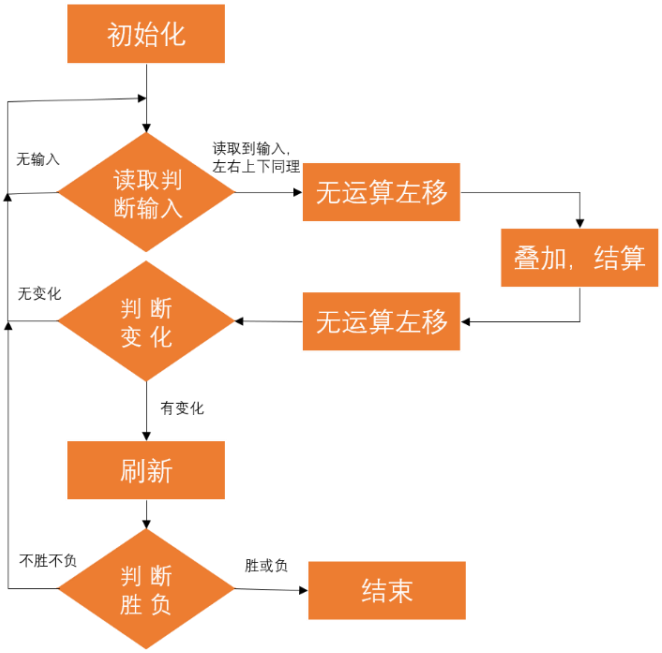

图 4-1 核心算法流程图

（2）主函数

此时的主函数与项目主体关系不大，也不会被移植，只是用于测试和参考，在此略加说明。

主函数中首先进行初始化，并且开启一次游戏（这些都会在下文提到），接着在一个无限的 while 循环中，读取键盘的输入，并通过 switch 语句判断，判断后进行各种不同的操作。最后在接收到结束信号（胜利或失败）时，强制退出无限循环。

（3）初始化

gameArray 进行清零，分数清零，各种控制标志变回默认值，最后随机生成带有数字 4 和 2 的两个方块作为下一局游戏的起始。

（4）显示

控制台中的显示与实际界面关系不大，在此忽略。

（5）矩阵复制

二维数组的复制在下文提到的函数中多次运用，故封装为函数，通过全局变量transmit[dimension][dimension]来承载复制的gameArray。

（6）判断变化

判断 gameArray 是否发生变化也是非常重要的步骤，不仅在一次操作判断无变化后不再随机生成新的数字的过程中需要，还可以减少界面不必要的刷新。

由于每次操作后，上一次的 gameArray 中的元素都被完整的保存在 transmit 中，故判断不变化的方法非常简单，将 transmit 和 gameArray 中的元素一一对照，全部一致则认为无变化。

（7）左移（上、下、右移同理）

这里的左移指的是单纯的左移（下文都称单纯左移），不做任何计算，即把每行的所有数字都依次移到最左端。这里直接将算法贴出，如图 4-2 所示。其中 k 是临时数组，用于存储每行实际不为 0 的元素数，Trans()即是上文提到的矩阵复制函数。

```
private static void LeftMove()
{
    for (int m = 0; m < dimension; m++) k[m] = 0;
    Trans();
    for (int i = 0; i < dimension; i++)
        for (int j = 0; j < dimension; j++)
        {
            if (transmit[i, j] != 0)
            {
                gameArray[i, k[i]] = transmit[i, j];
                k[i]++;
            }
        }
}
```

图 4-2 单纯左移算法

（8）附带计算的左移（上、下、右移同理）

这里的左移是完全的左移，即向左滑动后游戏做出的反应。首先进行单纯左移，然后进行列间相邻元素的叠加判断，若单纯左移后列间相邻元素相同，则叠加，即被叠加的数字乘以二，叠加的元素清零，叠加的同时进行分数的结算，并且判断是否获胜。叠加完成后，有可能出现非零元素并不是每行依次靠左排列的情况（比如有一行一次出现了 2 组叠加），此时须再调用单纯左移的函数，不再进行叠加，使排列符合逻辑。

（9）“数零”

每次操作后，如果 gameArray 发生了变化，需要在空白处随机生成一个新数字，此时我们就需要直到空白的数量，进行数零的操作。“数零”返回一个整型数，表示空白的数量，逻辑简单，在此略过。

（10）判负

首先通过“数零”判断是否还有空白，若无空白，再判断是否有行列间相邻的元素相同，若无，则认为没有继续操作的空间，判断为负。

（11）随机生成数字

每次使二维数组有变化的操作后，需要在空白处生成新数字。若还有空白（二维数组中还存在 0），则以 1 到“数零”返回值为界（边界可取），产生随机数，记为 gi。遍历二维数组的各个元素，第 gi 个为零的元素即为将要生成新数字的空白。生成的新数字可由函数的参数进行传递。

随机数通过伪随机数的方式实现。

2、基于 lvgl 模拟器的界面及功能设计

（1）显示

I．游戏主体

游戏主体是平方数个正方形排列的方块，最少为 4 个，最多为 64 个。方块以按钮的形式存在。当呈现不同数字时，方块呈现为不同的颜色。当方块数字为 0 时，方块内不显示数字。

有函数 void lv_2048(int dimension)作为刷新方块显示的函数，其中 dimension 是方块阵的维度。调用该函数时，若表示方块的静态按钮对象未创建，则首先利用循环创建并根据维度进行合理地布局，随后利用 API 函数 lv_obj_set_hidden(lv_obj_t* obj, bool b)设置按钮是否被隐藏。一般维度发生改变时，会隐藏一些多余的按钮并重新布局。布局完毕后，根据按钮对应的不同的数组元素，对按钮重新设置 style，以改变背景颜色。另有计分板上的分数刷新。

对于上文提到的 style，除了 0 对应的 style 因为要单独处理设置为全局变量（护眼模式下，该变量需要影响其他的 C 文件），都为静态变量，每个 style 在界面初始化函数 void lv_2048_others()中初始化，并分别赋值。

每当游戏界面需要刷新时，即调用 lv_2048(dimension)，即可完成刷新，且多余的刷新较少。

II．主界面其他

上文提到的 lv_2048_others()，该函数进行主界面的整体初始化，包括左侧的按钮、标签、滑块等。同时调用第一次 void Initialize()以及 lv_2048(dimension)，完成游戏的初始化和游戏方块的初始化。整体主界面情况如图 4-3 所示。

（2）代码移植

上文提到的 C#代码涵盖了游戏核心的基本功能，大体变化较少。但由于编程语言之间的一些区别，做出一些改动。原二维数组 gameArray，由于 C 语言不支持变化长度的数组，全程使用指针又比较麻烦，我们采取了确定长度的一维数组 gameArray[64]，以定长一维数组的方式表示变长二维数组，即当 dimension=x 时，原来的 gameArray[i][j]，就等于现在的 gameArray[i+j*x]。定长数组中没有对应关系的元素即是 4.2.1.1 中提到的需要隐藏对应方块的元素。

在输入检测方面，控制台中是以键盘的上下左右控制，实际开发板中无法使用键盘，故采取两种方式实现输入。一是在右侧添加 4 个不同的按键，表示键盘的上下左右；二是类比

手机触摸屏，以触屏滑动的方式呈现，这部分内容由于不是在模拟器中完成的，将会在下一部分，基于 lvgl 的 ESP32S2 程序框架中详细描述。

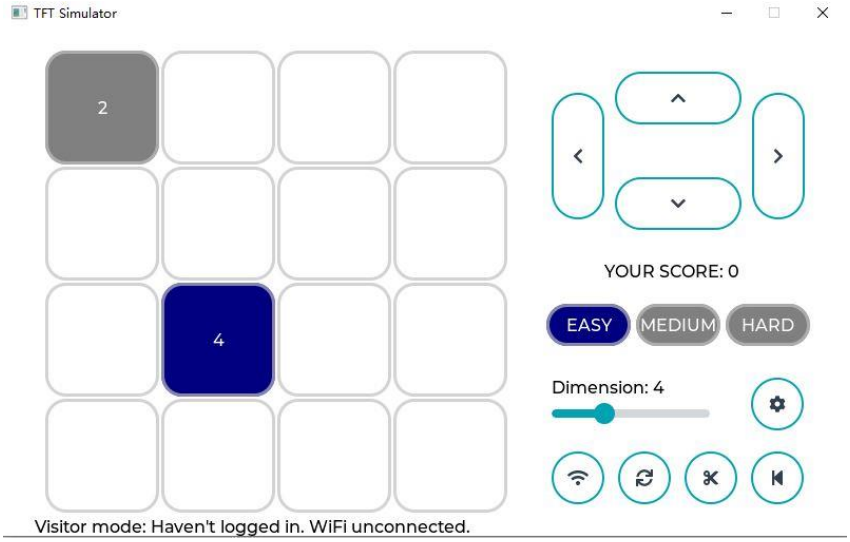

图 4-3 游戏主界面

（3）基本功能实现

基本功能在代码的移植完成后已基本实现。添加了在胜利或失败时出现弹窗，选择记录分数并重新开始或者继续游戏（失败时继续游戏可以通过高级功能中将要提到的作弊手段扭转败局）。

通过对 lvgl 例程中关于弹窗的例程的学习，我们将弹窗这个对象的各种功能熟练地运用到了项目中。首先生成一个遮盖整个屏幕的黑色 obj 对象，对其设置动画，弹窗出现时由全透明到半透明，随后以这个 obj 为父对象创建消息窗对象，此时由于被覆盖，只有消息框能够活动。初始化消息窗，添加消息和按钮阵。同时绑定事件到消息窗，但按下按钮时，消息窗消失；当消息窗消失时，删除消息窗的所有父对象。

最后将生成消息窗的代码封装成函数 void msgbox_type_1(const char* key[3], char* info)，对于不同的需求，生成不同的消息窗，方便多次调用。值得注意的是，在消息窗绑定的事件函数中，我们发现，由于按钮的内容确定时，需要的响应基本相同，所以我们通过字符串比较按钮内容的方式来确定按下的按钮，实现不同按钮对应的不同的功能，相同按钮对应的相同的功能。

（4）高级功能实现

高级功能分为 3 种，即主界面高级功能，设置界面高级功能和登陆界面高级功能。

I．主界面高级功能

A．难度调节

如图 4-3 所示，计分板下的三个按钮，能够控制游戏的难度。游戏难度由全局变量 difficulty 表示，这是一个枚举变量，有 EASY、MEDIUM、HARD 这 3 种取值。

这个功能只有在登陆后才能够正常使用。

一个按钮在选中状态时（如图中 EASY），其他按钮会进入未选择状态。

按下其中一个按钮后，若未登陆，游戏会提示你进行登陆，可以选择跳转到登陆界面；若已登陆，游戏会提示更改难度需要重新开始游戏，若确认重新开始，游戏会重新启动并修改难度，若取消，游戏会恢复原来的状态。

这里有一些实现的细节不再赘述，可以在附件中的代码里自行摸索。

B. 维度调整

参照滑块的例程，添加滑块用于表示维度，最小为 2，最大为 8，默认为 4。

该功能只有登陆后才能正常使用。

拉动滑块后，若未登陆，则可以跳转到登陆界面；若已登陆，会提示需要重新开始游戏，若确认重新开始，游戏会重新启动并修改维度，若取消，游戏会恢复原来的状态。图 4-4 为 8 维游戏界面。

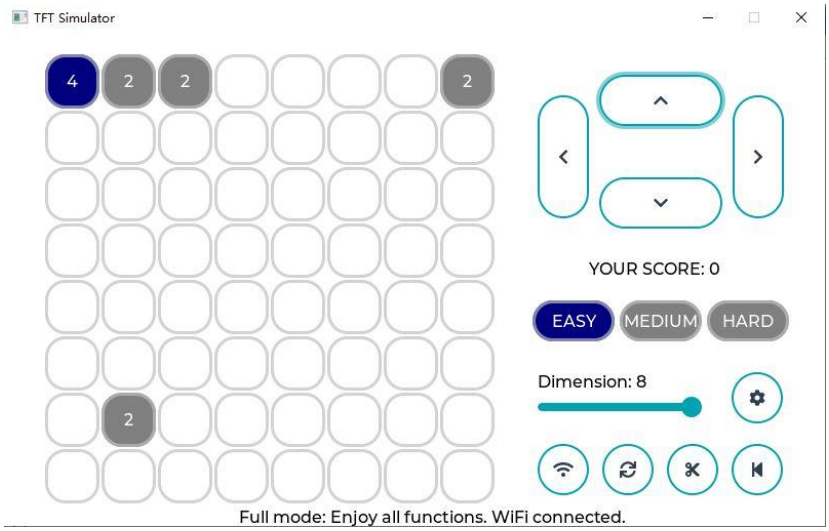

图 4-4 8 维游戏界面

C. 设置

滑块右侧的按钮表示设置，点击后即可进入设置界面。

D. 伪 WiFi 连接

右下角的第一个带有 WiFi 图标按钮。

此处的 WiFi 并不是真的 WiFi，但连接 WiFi 是登陆的必要条件。

点击按钮后，若未连接，会自动连接 WiFi，如图 4-5 所示，进度条的变化以动画的方式实现；若已连接，则可以选择断开或不断开连接。

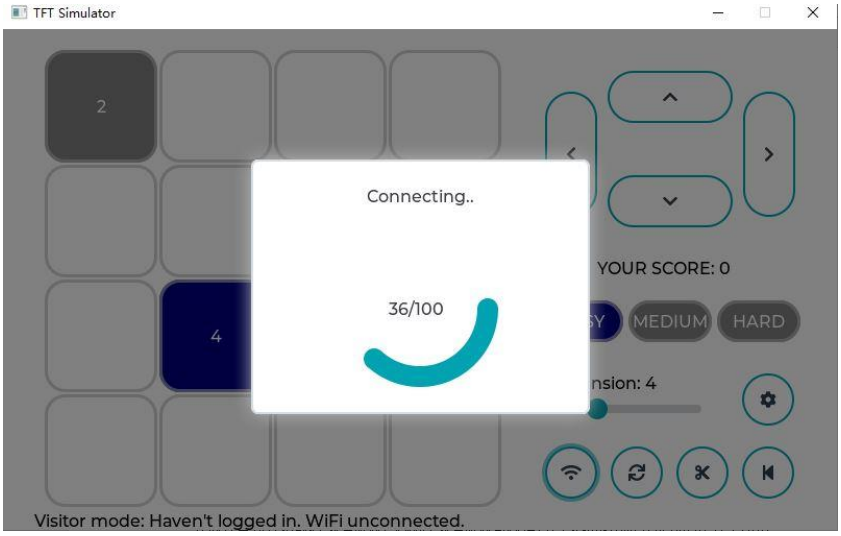

图 4-5 伪 WiFi 连接中

E. 重新开始

右下角的第二个带有刷新图标的按钮。按下后即可确认重新开始游戏或取消，该功能无需登陆后的权限。

F. 破坏

右下角的第三个带有剪刀图标的按钮。

游戏时，当你想直接消除一个碍事的方块时，你可以登陆后使用该功能作弊。登陆后，每局游戏享有 3 次机会，每次将会扣除 1024*次数的分数。当没有剩余次数或没有方块可以破坏时，游戏也会加以提示。

进入破坏状态后，右半屏幕将会被透明的黑色 obj 对象遮盖，只有游戏方块部分会显示，此时按钮和事件绑定，点击非零方块对应的按钮后，会弹出是否确认破坏的消息框，确认后破坏该方块，无法反悔。

G. 撤销

右下角的第四个带有回退图标的按钮。

游戏时，当你不慎操作失误想要撤回到上一步时，你可以登陆后使用该功能作弊。登陆后，每局游戏享有 3 次机会，每次将会扣除 1024*次数的分数。当没有剩余次数时，游戏也会加以提示。

H. 提示消息

主界面最下方有一行提示信息，初始化时从左向右移动，每当刷新时，如果该消息处于静止状态，则会被赋予向另一侧移动的动画。

提示消息的内容主要与登陆权限和 WiFi 连接有关。

II. 设置界面高级功能

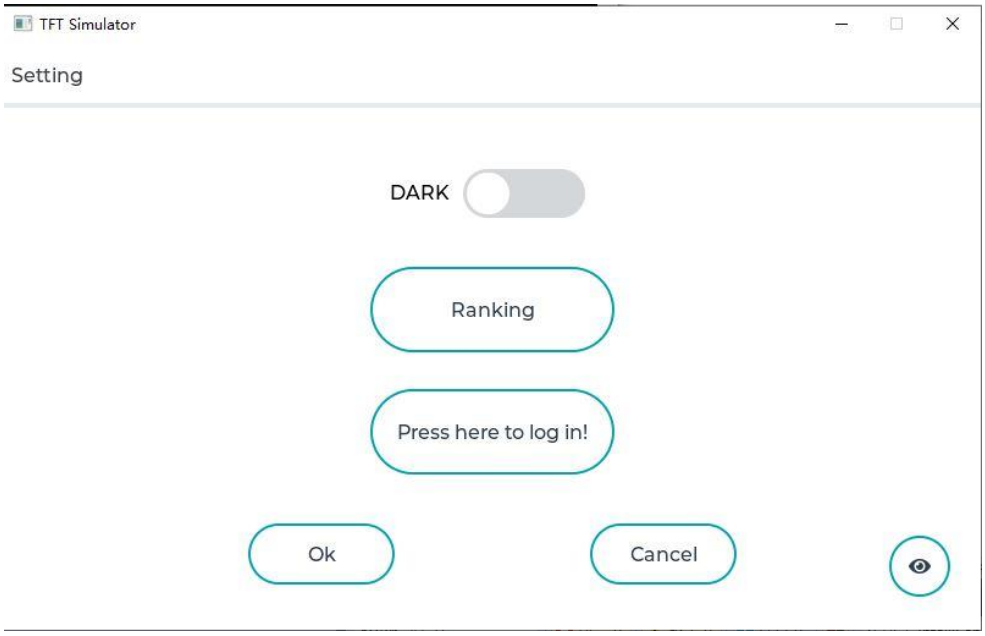

图 4-6 设置界面

设置界面如图 4-6 所示。该节面主要有 4 个功能，例如：

A. 护眼模式

标签 DARK 后的开关即为护眼模式开启开关。护眼模式开启后，各界面背景替换维黑色，字体替换为易于识别的颜色。图 4-7 展示了护眼模式下的主界面。

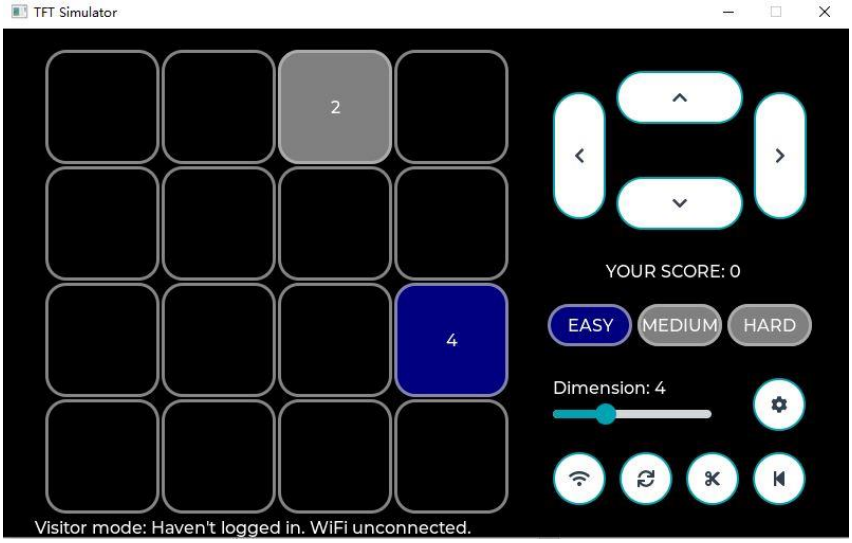

图 4-7 护眼模式下的主界面

B. 排行榜

开关下的按钮表示此功能。按下按钮后弹出消息窗，消息窗中带有表格，显示排名。排行榜中只显示用户的用户名以及分数，最多显示 3 个，是一个简单的功能呈现，有很大的继续开发的空间。

每次开启开发板电源和开关后，之前的分数排行将会依旧存在。这样的存储功能是参照 esp-idf 例程 spiffsgen 实现的，基于烧录镜像中文本文件的读写。但这部分的调试不在模拟器上进行。

C. 登陆

按下中间的按钮“Press here to log in”，可以进入登陆界面。

D. 版权显示

右下角带有眼睛图标的按钮表示此功能。按下该按钮后弹出显示版权信息的消息窗。

III. 登陆界面高级功能

登陆界面如图 4-8 所示，主要负责登陆的功能。

登陆需要用户名和密码，有默认密码“2048”，暂无注册的功能，用户可以使用任意非空且长度小于 10 的用户名进行登陆。登陆需要 WiFi 连接，登陆完成后可享有全部权限。

轻触文本输入框，会在下半屏幕弹出键盘，如图 4-9 所示。键盘是参照 lvgl 的例程实现的。输入完成后，轻触空白部分即可退出键盘。但是值得注意的是，键盘上的×键，按下后会出现 bug，这是一个在例程中就存在的问题，暂未解决；以及键盘上的√，按下后并不会视作输入完毕，仍需轻触空白部分完成。

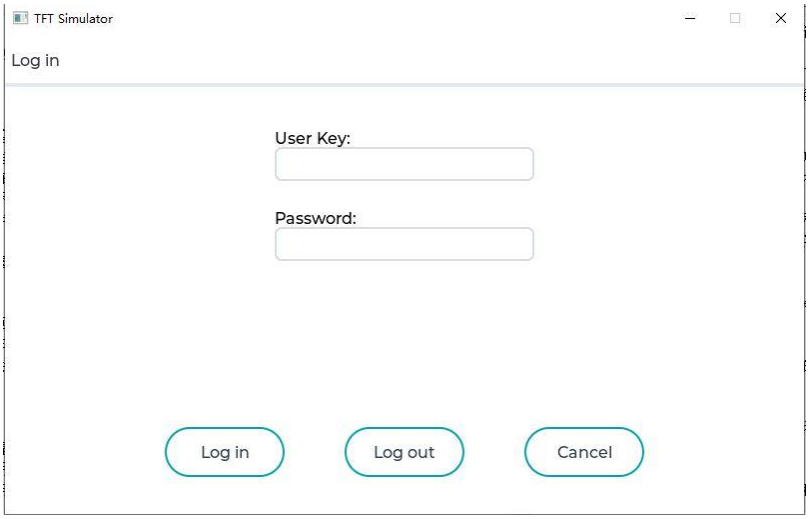

图 4-8 登陆界面

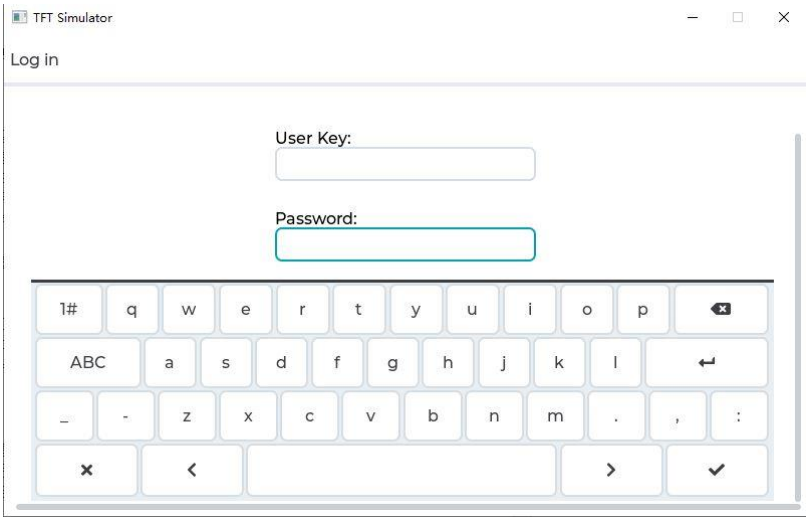

图 4-9 键盘弹出

图 4-8 最下方 3 个按钮表示登入、登出和取消，按下后会根据不同的状态提示或跳转，逻辑清晰，可以在模拟器或开发板上自行探索。

3、基于 lvgl 的 ESP32S2 程序框架

（1）简要分析原框架

在主函数 app_main(void)中，首先初始化 lcd 和 ft5x06，最后通过 freeRTOS 的 API 函数 xTaskCreate(gui_task, "gui_task", 20 * 1024, NULL, 10, NULL)创建 gui 任务，gui 任务是整个程序整个框架的核心，通过修改 gui 任务，我们可以实现想要的功能。

在在 gui 任务中，lv 初始化后，最高优先级的是利用 xTaskCreate 创建 tick 任务，lvgl 需要 tick 才能知道动画和其他任务的经过时间。本项目中的计时以 10ms 为单位，在 static void hmi_tick_task(void *arg)中呈现，如图 4-10 所示。

```
static void hmi_tick_task(void * arg)
{
    while(1) {
        lv_tick_inc(10);
        vTaskDelay(10 / portTICK_RATE_MS);
    }
}
```

图 4-10 tick 任务调用函数

第二步是创建显示缓存，初始化并注册显示驱动。

第三步是初始化输入驱动，确定输入类型和回调函数，根据驱动注册输入设备（在此指指针型输入触摸），并对触摸点加上指示光标。在本项目中，我们认为只是光标的加入影响界面的美观，故删去了这一段代码。

第四步创建了一个 lv_task，用于开启监视器。在实践中发现，开启该监视器会使开发板在烧录成功过后一小段时间后卡死，原因未详细查明，且在本项目中对调试帮助不大，故也删去。

第五步是调用用户函数，我们采取 lv_task_once(task)的方式调用函数 lv_2048_others()完成 hmi 界面的整体的首次初始化，防止后面被卡死。

最后将会进入一个无限循环，反复调用 lv_task_handler()，处理和调用其他的 lv_task（例如第五步中创建的 lv_task）。

（2）触屏滑动

触屏滑动的功能能够最大化地利用触摸屏的可触摸特性，区别触摸屏和传统 PC，是本项目的核心功能之一。由于输入需要在无限的循环中进行检测，故我们将这段代码放在 4.3.1 提到的 lv_task_handler()的后面。

在注册输入设备完成后，我们可通过 lvgl 的 API 函数_lv_indev_read(touchpad_indev, &touchpad_data)读取输入设备数据（包括指针和状态），并根据这指针和状态的前后对比判断触屏滑动的方向，做出相应的响应。图 4-11 展示了其中一小部分功能的代码实现。

另外，触屏滑动的功能需要在时间和空间上进行约束。在空间上，需要确定起始或终止指针的坐标在游戏方块的范围之内；在时间上，只有在游戏主界面的游戏进行时才能够触发这个功能，否则会出现类似在设置界面操作暂不可视的主界面游戏的混乱情况。

（3）存储功能

存储功能是连接游戏多次启动之间的桥梁，仿照 spiffsgen 的例程实现，主要用于排行榜功能，也可将此原理应用于注册功能，但由于时间有限，未编写此功能。

下面介绍 spiffsgen 文件系统的开发流程：首先按照需求复制例程中的头文件到 main.c 中，并仿照例程编写所需的代码，spiffs 配置和例程保持一致即可（在本项目中，即编写初始化时从文件中读取排名数据和排名变化时写入文件的代码）；编写完成后，打开目录下的 partitions.csv 文件，在最后一行加入 Name：storage，Type：data，SubType：spiffs，Size：0xF0000；再打开 main 文件夹中的 CMakeLists.txt，在最后一行加入 spiffs_create_partition_image(storage ../spiffs_image FLASH_IN_PROJECT)，以指定镜像对应的文件夹；最后在目录中新建名为“spiffs_image”的文件夹，即可像例程中一样，在项目中读取或者写入该文件夹中的文件。

```
_lv_indev_read(touchpad_indev, &touchpad_data);
if (touchpad_data.state == LV_INDEV_STATE_REL)
{
    pressing = false;
    deltax = touchpad_data.point.x - pre_point.x;
    deltay = touchpad_data.point.y - pre_point.y;
    if (touchpad_data.point.x < 500 || pre_point.x < 500)
    {
        if (fabs(deltax) > SENSITIVE || fabs(deltay) > SENSITIVE)
        {
            if (fabs(deltax) > fabs(deltay))  //Horizontal
            {
                if (deltax > 0)  //Right
                {
                    if (isGaming && !destroying)
                    {
                        RightShift();
                        lv_2048(dimension);
                    }
                }
```

图 4-11 部分触屏滑动代码

4、编译及烧录

（1）编译

编译部分主要参照了 ESP32-S2-LCD-DevKit 的一个官方的 printer 例程，然后用 cMake 工具，它可以用简单的语句来描述所有平台的安装(编译过程)。在本工程中，主要的编译结构有如下几个方面：

CMakeLists.txt（CMake 配置文件）

```
# The following lines of boilerplate have to be in your project's
# CMakeLists in this exact order for cmake to work correctly
cmake_minimum_required(VERSION 3.5)

add_compile_options(-fdiagnostics-color=always)
include($ENV{IDF_PATH}/tools/cmake/project.cmake)
project(esp32s2-i2s-lcd)
```

图 4-12 CMakeLists.txt

这是 cMake 加载的配置文件，首先通过 cmake_minimum_required(VERSION 3.5)定义了 cMake 所支持的最小版本。然后使用 include($ENV{IDF_PATH}/tools/cmake/project.cmake)来包含 esp-idf 下的 project.cmake 文件，这个文件包含许多需要加载的组件。最后采取 project(esp32s2-i2s-lcd) 来说明工程名为 esp32s2-i2s-lcd。

Makefile（make 配置文件）

```
PROJECT_NAME := parallel

include $(IDF_PATH)/make/project.mk
```

图 4-13 Makefile

Makefile 文件是 make 命令加载的配置文件。

main（工程主代码目录）

main 文件夹用来放置工程主要源文件。包括：

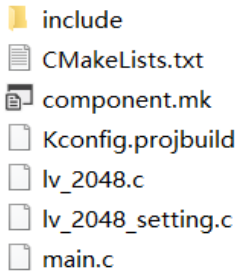

图 4-14 main 工程目录

include 目录中包含了需要的几个头文件，有 gui.h，lv_2048.h，lv_2048.setting.h。

CMakeLists.txt 用来编译 main 组件的配置，函数 set(COMPONENT_SRCS…) 和 set(COMPONENT_ADD_INCLUDEDIRS…) 用来设置相关源码的位置，最后通过 register_component() 注册组件。如图所示：

```
set(COMPONENT_SRCS  "main.c"
                    "lv_2048.c"
                    "lv_2048_setting.c"
                    "mouse_cursor_icon.c"
                    )

set(COMPONENT_ADD_INCLUDEDIRS "include")

register_component()
```

图 4-15 main 目录下的 CMakeLists.txt

component.mk 用来编译 main 组件，一般是个空文件，在这种情况下会默认编译该目录下的所有代码。

lv_2048.c，lv_2048.setting.c，main.c 三个 C 文件就是只要的代码文件，在这里实现主要的功能。

sdkconfig（通过 make menuconfig 生成的配置文件）

sdkconfig.old（通过 make menuconfig 生成的备份配置文件）

（2）烧录

烧录过程参见前面第三部分中例程测试部分。

5、最终呈现

（1）机制

整体游戏机制与常规 2048 相仿，以下说明一些具体区别。

I. 难度

难度分为 3 种，EASY 难度每次随机只生成 2；MEDIUM 难度每次有一半的几率随机生成 4；HARD 难度会随机生成 2，4，8，几率比是 1：2：1.每种难度的分数累计机制相同，下文将阐述。

II. 分数计算

每次叠加合成新的方块时，新方块的数字即为本次的得分。

III. 作弊功能惩罚

作弊功能不仅有次数限制，也需要付出一定代价，每使用一次相同种类作弊功能，都会失去 1024*已使用次数的分数。

IV. 排行榜

排行榜只是简单功能的实现，不区分游戏的难度、维度等因素，只排行分数。

（2）完成度

参照项目计划书，完成度较高，所有拟定的基本功能和高级功能都已实现。工程源代码及相关说明已上传至 https://github.com/cyGnail/esp32-s2-hmi。

（3）效果展示

测试同学发挥坚持不懈的优良精神，在 PC 模拟器上对游戏进行长时间测试，如图 4-12 所示，游戏稳定性较高，未出现明显 bug。

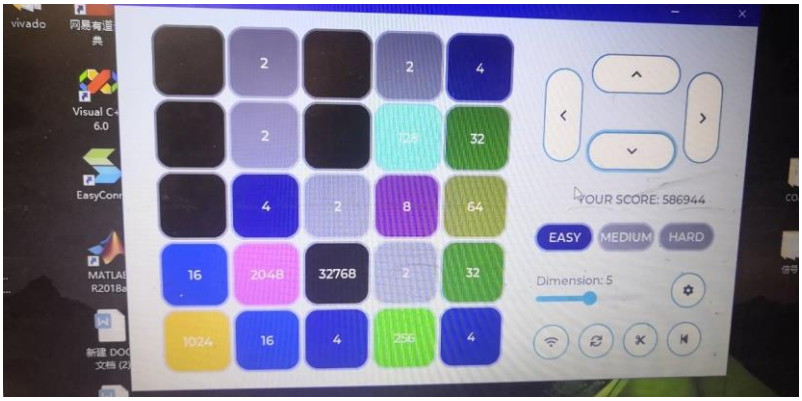

图 4-16 测试画面，一直到 58.7w 分都未出现 bug

烧录到开发板后，画面也较为稳定，且短时间测试后各功能也暂未发现明显 bug，如图 4-13 所示。

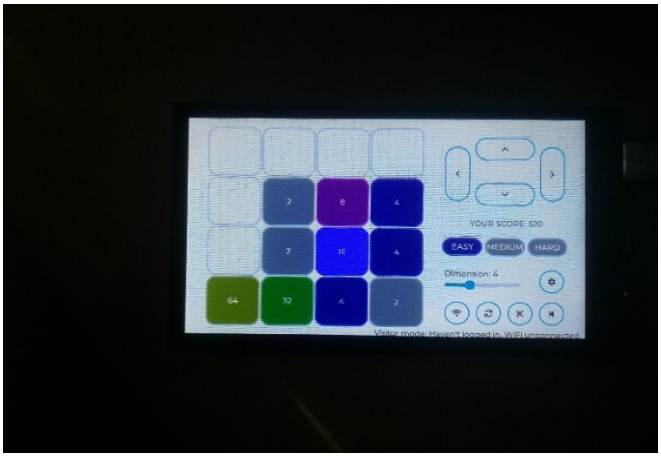

图 4-17 开发板上运行效果

6、存在问题

虽然完成度较高，但仍有许多缺陷。以下列出几点：

（1） **功能不够完美：**排行榜过于简单；注册功能没有多余时间开发；设置界面内容过少；WiFi 不能够真正连接；键盘上有一些小 bug 未解决。

（2） **代码冗长：**一开始代码量较少时，代码书写较为美观，也十分注意代码的整洁性、低耦合性和注释的完备性。但是后期代码量逐渐增大后，由于懒惰，很多地方都没有添加注释，且代码的耦合程度逐渐提高，以后需要注意。

（3） **界面未做美化：**界面中大多数控件都是默认的 style 和字体，未做修改，有些地方明显觉得放大字体会比较美观。且大多数控件的坐标都是直接赋值，个人认为使用 obj 对象的对齐比较合理。

五、进度与分工

第一周（2020.8.31—2020.9.6）：

搭建环境，完成项目计划。

第二周（2020.9.7—2020.9.13）：

完成 2048 的控制台编写，实现 2048 基本功能。

第三周（2020.9.14—2020.9.20）：

完成高级功能的编写。

第四周（2020.9.21—2020.9.27）：

完成前几周未收尾的工作，对项目进行优化，总结工作。

时间安排如上，项目实际进度与之相仿。具体进度可详见实习日报，周报。

分工：

谯兵：代码的移植、编译与烧录；撰写报告；

梁宇辰：基于 lvgl 的界面开发；调试；撰写报告；

祁栋华：研究算法；调试；撰写报告

六、总结与体会

04017411 谯兵：

总的来说，本次生产实习是一次非常好的体验，同时非常感谢乐鑫给了我们这一次实践的机会，从中学到了很多，也领悟到了很多。从最开始的环境都不知道怎么搭建，到最后开发板上一个完整的 2048 游戏，虽然在此期间遇到了很多的问题，但是在乐鑫工程师的帮助和上网自行查找解决方案下，都成功被解决了。

04017414 梁宇辰：

本次实习收获很多。感谢学校和乐鑫科技给与了我们这次宝贵的机会。在实习过程中，我们克服重重困难，从无到有，在组员间的齐心协力和工程师的悉心指导下，完成了整个项目的开发过程，取得了不错的结果。

在此期间，我切身体会到了将理论知识运用到实践中的不易，也认识到，只有不断地接触新事物和不懈地实践，才能够收获成功和能力的提升。

04017417 祁栋华：

这次的生产实习，首先感谢乐鑫和学校能给我们这么一个宝贵的机会，让我们能在团队项目开发过程中学习更多的实践知识。在项目刚开始，我们在乐鑫工程师的指导下搭建环境，烧录测试示例。当时，我们还是较为困惑的，有关于 esp32S2 板子的，有搭建环境过程中的各种报错，以及后续如何开展项目，做什么项目等等各种问题。因此，前期进度显得十分缓慢，四五天时间都只是刚刚搭建好环境。所幸，后来在工程师发放的各类文档指导，以及学院生产实习指导老师的协调下，我们逐步形成自己探索为主，同时请求工程师辅导指点疑难问题的项目开发模式，成功追上了预期进度，在第二周内成功实现了 2048 游戏的各项基本功能。在项目实施过程中，首先学习了 printer 例程，在成功烧录测试后，修改 project 例程，尝试在 main 目录中添加 lv_demo_widgets 的 h 和 c 文件，遇到编译问题后，在工程师的指导以及组长组员的帮助下成功解决问题。这是项目开始以来解决的比较主要的问题。在这种逐步学习，逐步尝试，逐步反思直至解决问题的过程中，我进一步体会到团队协作对项目的重要性。

项目开发过程中，我们学会了如何搭建 esp-idf 环境，如何烧录测试历程，学习并配置了 esp-iot-solution，在控制台中编写项目，学习了 LVGL 开发，界面设计，并自主进行调试、修复 bug，测试各项功能能。同时，我也进一步理解了与先进者主动寻求交流，团队协作的重要性。

七、附件

1. 简要设计文档
2. 工程源文件
3. 实习日报，周报
4. C#控制台代码