

目录

1 项目总体设计	3
1.1 需求分析	3
1.2 功能目标	3
1.3 总体结构图	3
2 项目硬件配置	4
2.1 主板配置	4
2.2 时钟配置	4
2.3 LED 配置	5
2.4 蜂鸣器配置	6
2.5 按键	6
2.6 LCD 显示屏	7
2.7 电阻触摸屏设置	7
2.8 串口通信	8
2.9 FLASH	8
2.10 TIM 模块	9
2.11 总体引脚展示	10
3 项目模块设计	11
3.1 主要功能模块	11
3.1.1 游戏逻辑模块	11
3.1.2 显示模块	15
3.1.3 输入检测模块	16
3.2 辅助功能模块	17
3.2.1 LED 模块	17
3.2.2 蜂鸣器模块	17
3.2.3 串口通信模块	17
3.2.4 FLASH 模块	18
3.2.5 TIM 模块	18
4 系统实现	19
4.1 开发环境与开发工具	19
4.2 项目文件介绍	21

5 项目成果与测试	23
5.1 项目成果	23
5.2 项目展望	26

1 项目总体设计

1.1 需求分析

出于疫情期间的物流限制，我们小组很难购买到额外的外设部件，因此，我们依据手头的野火开发板为依托进行设计，以液晶显示屏和电阻触摸屏为核心进行项目构思。

同时，由于疫情隔离给社会带来了很大的心理压力，我们将构建轻松有趣的嵌入式设备益智游戏作为选题方向。经过分析与讨论，我们选择构建一款基于 STM32 的探索式迷宫游戏。

1.2 功能目标

我们构建的迷宫游戏实现了以下功能：

- 作为一款嵌入式设备游戏，本项目实现了基本的显示界面 UI，使用触摸屏进行主要控制，使用按键进行辅助控制，实现了音频反馈。
- 与传统的迷宫相区别，本项目的迷宫是探索式的迷宫，迷宫的大部分区域对玩家不显示，玩家只能获悉自身附近以及起点与终点附近的迷宫地图信息。
- 为了提高趣味性，本项目用玩家完成迷宫时行动的步数作为分数进行显示，并保存玩家的最佳成绩。

1.3 总体结构图

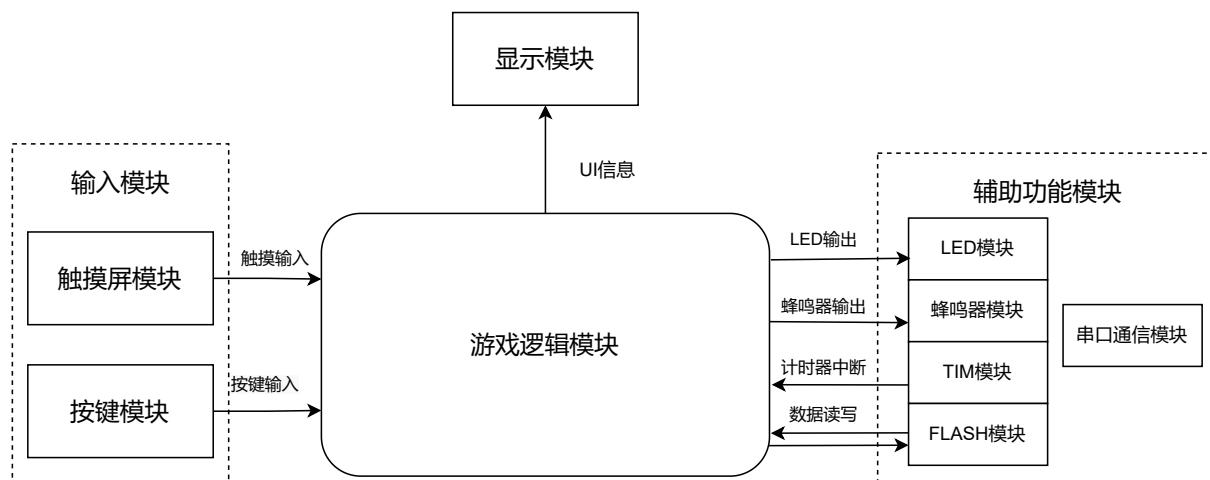


图 1: 总体架构图

2 项目硬件配置

2.1 主板配置

本项目使用的主板是老师提供的野火指南者开发板,CPU 系列为 STM32F103VET6。

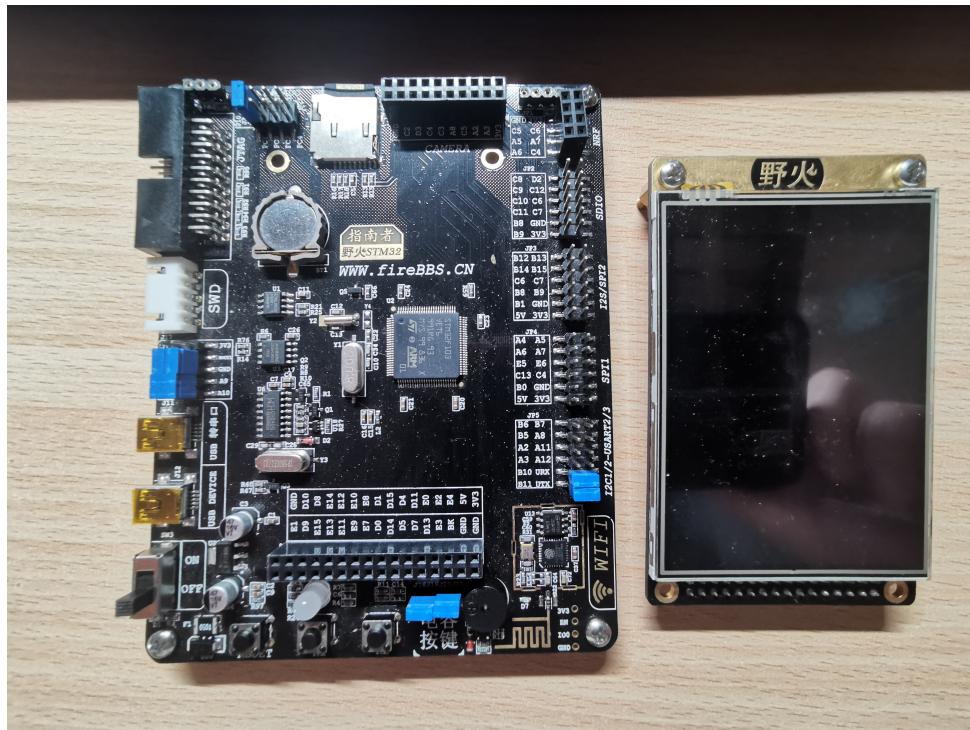


图 2: 主板与显示屏

2.2 时钟配置

本项目使用 8MHz 的外部高速时钟源 HSE，配置系统主时钟为 72MHz。时钟源电路原理图如图 3 所示，使用的两个端口分别为 OSC_IN 与 OSC_OUT。

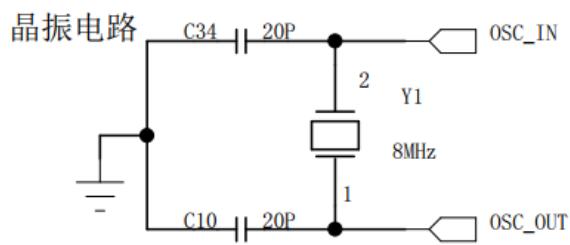


图 3: 晶振电路

完整的时钟树配置如图 4 所示。

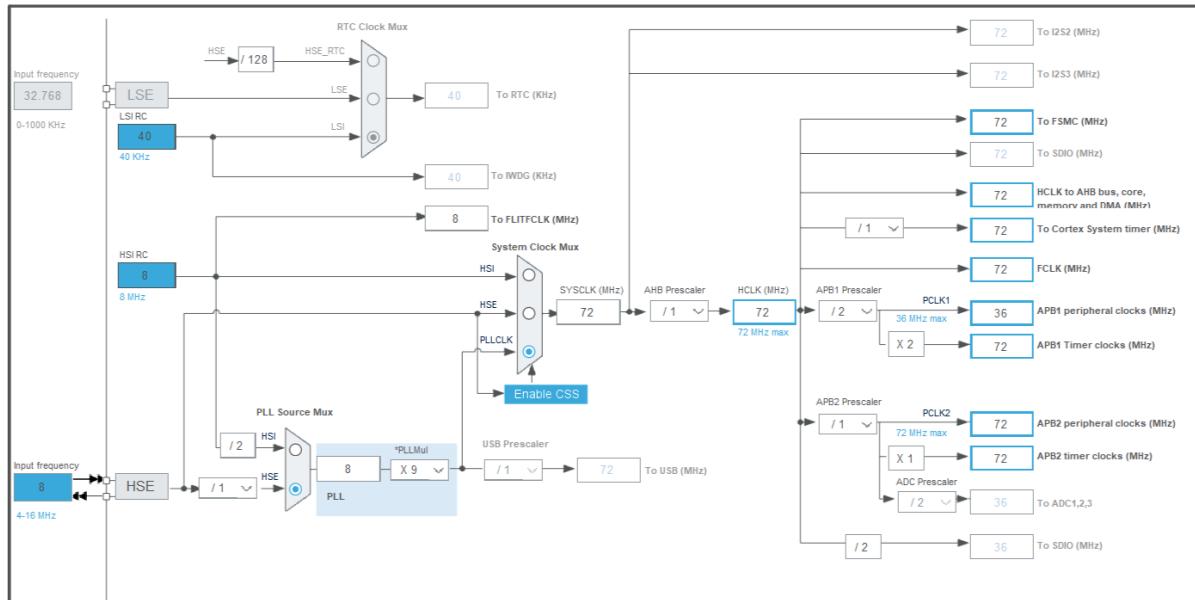


图 4: 时钟树

2.3 LED 配置

开发板上带有三个 LED 灯，分别为红色、蓝色、绿色，均使用推挽输出模式，当输出为低电平时，led 点亮。

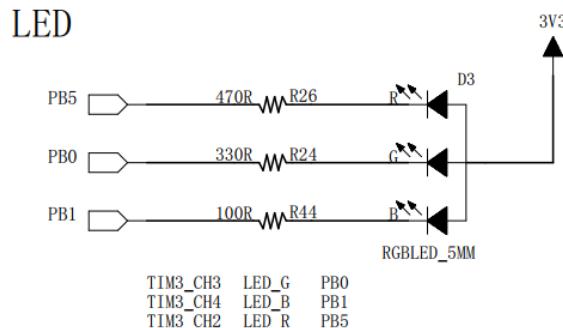


图 5: LED 模块

表 1: LED 端口配置

标签	端口	功能
LED_G	PB0	绿色 led
LED_B	PB1	蓝色 led
LED_R	PB5	红色 led

2.4 蜂鸣器配置

开发板上携带一个有源蜂鸣器，当输出为高电平时，蜂鸣器发声，蜂鸣器对应的端口为 PA8，配置为推挽输出模式。

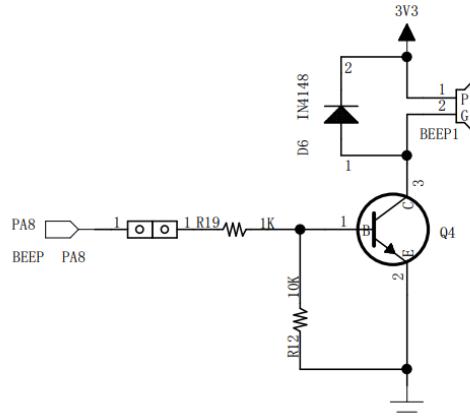


图 6: 蜂鸣器模块

表 2: 蜂鸣器配置

标签	端口	功能
BEEP	PA8	蜂鸣器

2.5 按键

开发板上携带两个按键，按键按下时可以产生中断，我们通过中断回调函数实现按键的功能，因此将两个按键都配置为中断端口，按键电路原理图及端口配置如下所示：

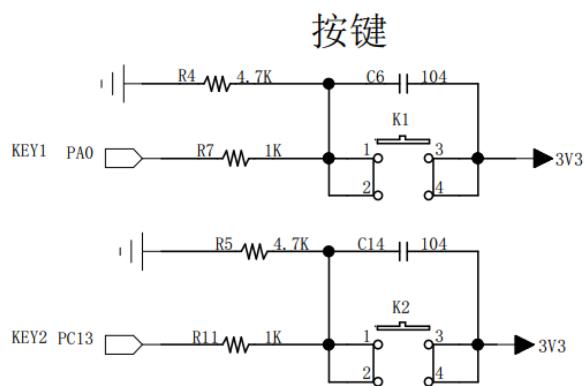


图 7: 按键模块

表 3: 按键配置

标签	端口	功能
KEY1	PA0	按键 1
KEY2	PC13	按键 2

2.6 LCD 显示屏

实验板携带一个 3.2 寸液晶显示屏，使用 ILI9341 液晶控制芯片进行控制，我们使用 STM32Cube 的功能辅助配置，设置如下：

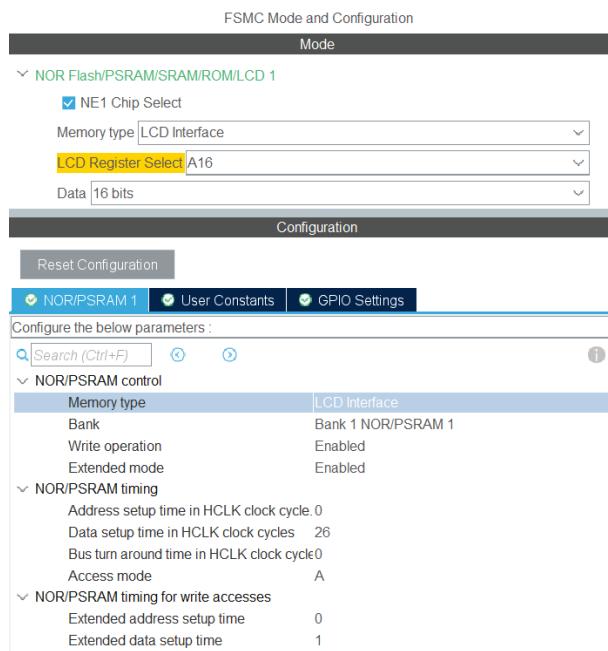


图 8: LCD 模块 FSMC 配置

除此之外，使用 GPIO 配置如下：

表 4: LCD GPIO 配置

标签	端口	模式
LCD_BL	PD12	推挽输出
LCD_RST	PE1	推挽输出

2.7 电阻触摸屏设置

电路板上携带的 3.2 寸显示屏同时包含电阻触摸屏 XPT2046，使用模拟 SPI 时序通信，配置端口如下：

表 5: XPT2046 配置

标签	端口	模式
XPT2046_CS	PD13	推挽输出
XPT2046_CLK	PE0	推挽输出
XPT2046_MOSI	PE2	推挽输出
XPT2046_MISO	PE3	悬空输入
XPT2046_PENIRQ	PE4	悬空输出

2.8 串口通信

使用 USART1 配置串口通信，配置如下：

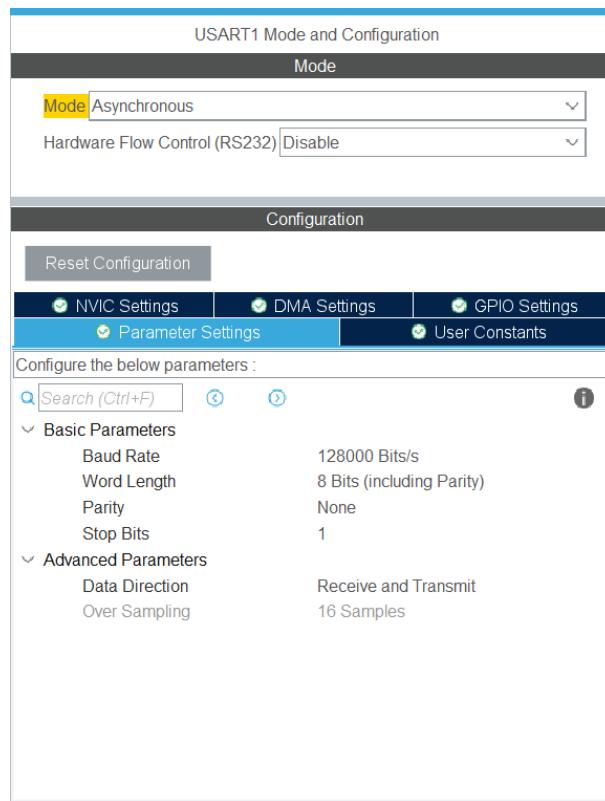


图 9: 串口通信 USART1 配置

使用两个 GPIO 端口进行通信，分别记 PA0 为 USART1_TX,PA10 为 USART1_RX。

2.9 FLASH

本项目使用 FLASH，一方面用于触摸屏的矫正参数保存，另一方面，保存玩家的最高分。本实验板上外置 FLASH 使用 SPI1 进行配置，配置信息如下：

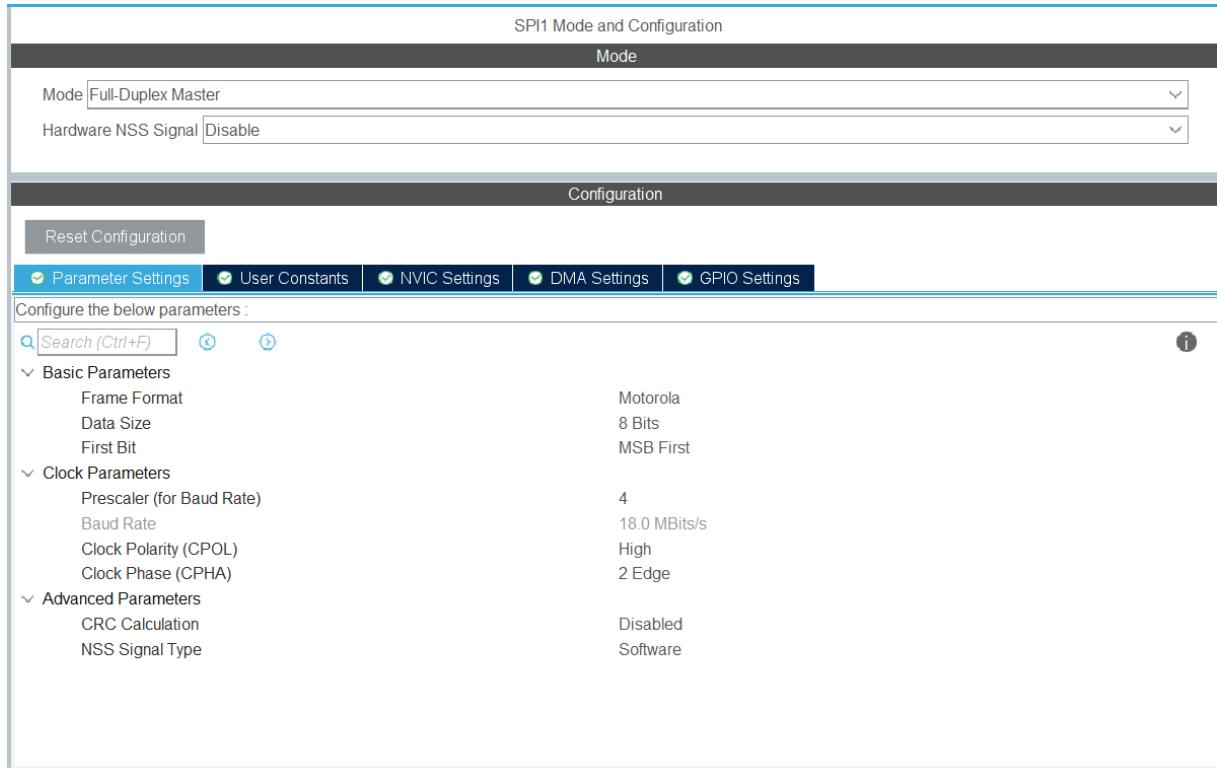


图 10: 外部 FLASH SPI1 配置

由于使用软件控制片选信号，还需要额外配置 GPIO 端口如下：

表 6: XPT2046 配置

标签	端口	模式
FLASH_SPI_CS	PC0	推挽输出

2.10 TIM 模块

本项目使用两个 TIM 模块，TIM6 与 TIM7 实现定时器中断，两个基本定时器配置一致。

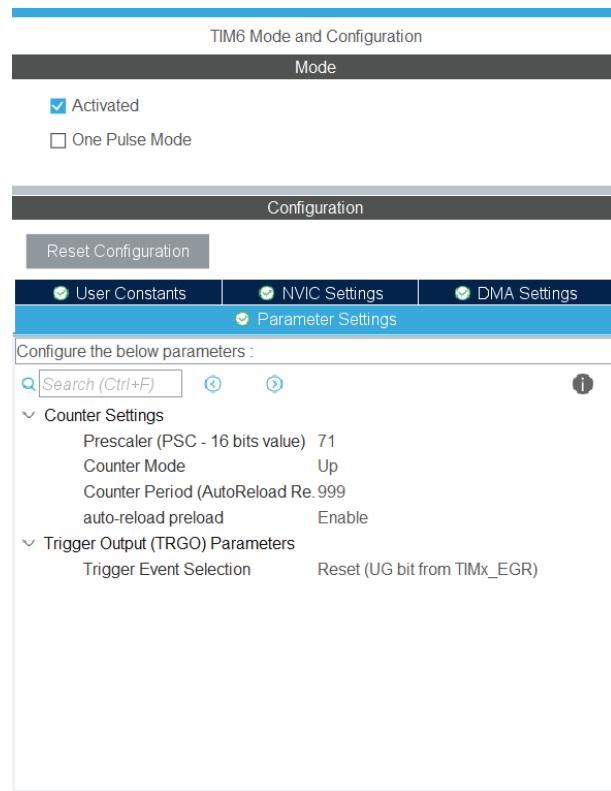


图 11: TIM6 与 TIM7 配置

每隔 1 微秒产生一次中断。

2.11 总体引脚展示

配置结束后，使用的外部引脚如下图：

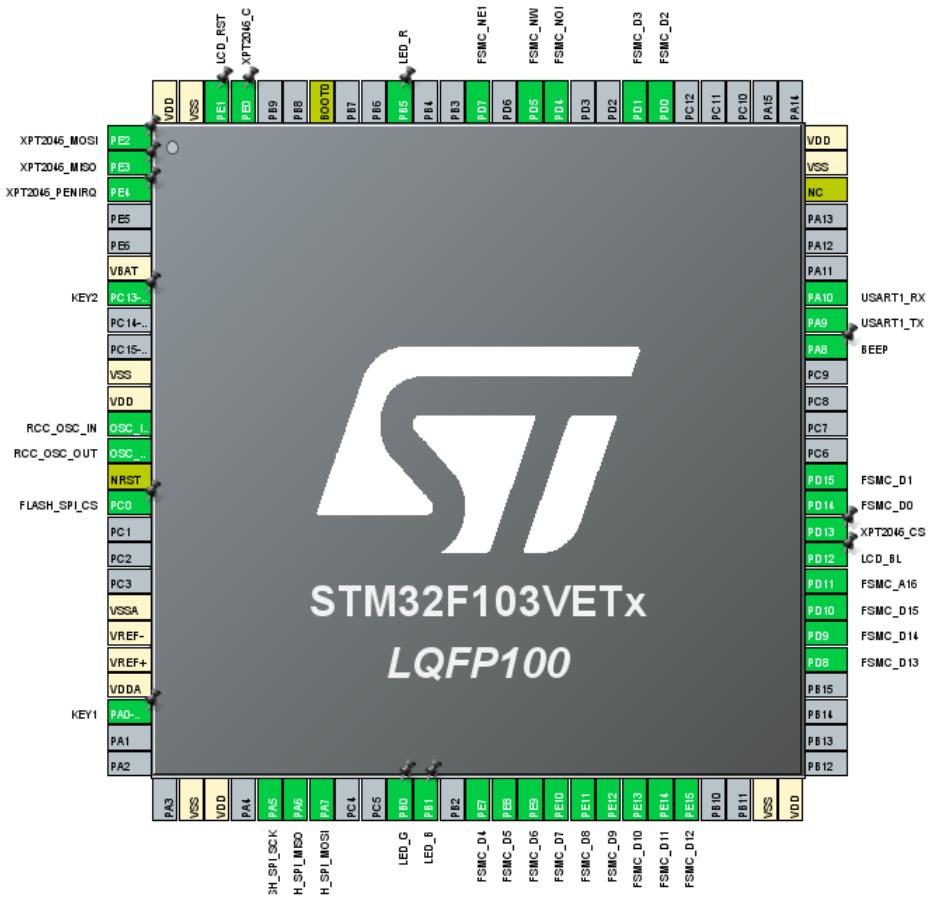


图 12: 引脚配置

3 项目模块设计

本项目分为主要功能模块与辅助功能模块，主要功能模块实现游戏逻辑控制等主体功能，辅助功能模块包括支持项目实现的其他全部模块。

3.1 主要功能模块

3.1.1 游戏逻辑模块

游戏逻辑模块是本项目的核心，本项目将主要功能分布在三个 UI 上，分别为 WELCOME_UI、PLAY_UI 与 WIN_UI，他们分别实现不同的功能，通过 UI 之间进行跳转支撑起完整的游戏体验。游戏逻辑模块会调用其他模块实现功能。本节将分别介绍三个 UI 实现的主要功能与跳转逻辑。

3.1.1.1 全局状态变量

本项目使用一些全局变量来记录迷宫状态和 UI 状态，其中迷宫状态的全局变量实例化位于 main.c，但结构体声明位于 maze.h 中：

```
typedef struct _maze_t
{
    int32_t step; // 当前玩家消耗的步数
    int32_t best; // 玩家的最佳记录
    uint8_t map[MAZE_X][MAZE_Y]; // 迷宫地图
    int32_t player[2]; // 玩家的位置
    int32_t enter[2]; // 入口位置
    int32_t exit[2]; // 出口位置
} maze_t;
```

其中迷宫地图格子分为墙、路、入口、出口四种类型，取值范围定义在 maze.h 中：

```
#define WALL_S (uint8_t)0
#define ROAD_S (uint8_t)1
#define ENTER_S (uint8_t)2
#define EXIT_S (uint8_t)3
```

本项目使用一个全局变量记录 UI 状态，定义在 main.c 文件中：

```
uint8_t ui_stat;
```

该变量的取值定义在 maze.h 中：

```
#define WELCOME_UI 0
#define PLAY_UI      1
#define WIN_UI       2
```

3.1.1.2 UI 跳转逻辑图

UI 跳转关系如下图所示：

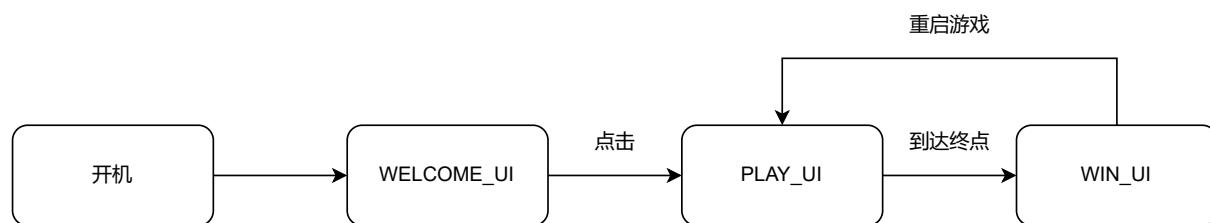


图 13: UI 跳转逻辑

3.1.1.3 WELCOME_UI

WELCOME_UI 是开机之后的欢迎界面，在整个程序运行流程中只显示一次，该 UI 上方显示本项目的项目名称：“MAZE GAME”，下方闪烁“START！”标志，用户点击后会跳转到 PLAY_UI。



图 14: WELCOME_UI 界面

3.1.1.4 PLAY_UI

PLAY_UI 是本项目的主要 UI，用户在这里通过点击下方按钮操作蓝色小点上下左右移动，探索迷宫地图以到达出口；在屏幕的右侧上方用红色字体显示人物已经移动的步数，用黑色字体显示玩家到达终点的最佳步数记录；在右侧下方显示“restart”按钮，玩家可以点击“restart”按钮回到黄色格子的起点，步数清空并重新开始。若玩家操控的蓝色小点到达绿色格子的终点，则会转换 UI 状态到 WIN_UI。

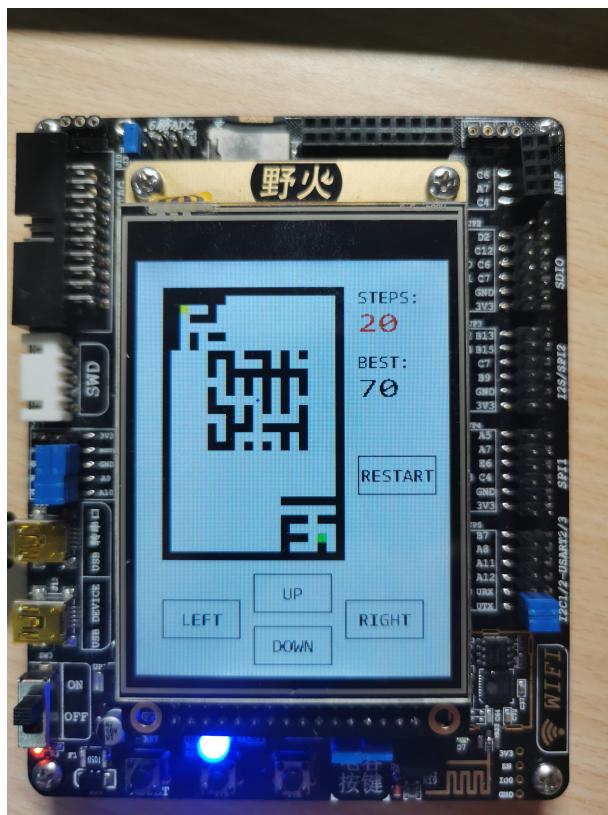


图 15: PLAY_UI 界面

3.1.1.5 WIN_UI

WIN_UI, 保持 PLAY_UI 的到达终点状态不变, 下方的操作按钮消失, 显示“YOU WIN!”标识, 玩家可以通过点击“restart”按钮回到 PLAY_UI 状态重新开始游戏, 若玩家的成绩突破了最好成绩, 则重新游玩时, 会显示更新后的最好成绩, 且这个最好成绩不会因为芯片重启被清空。

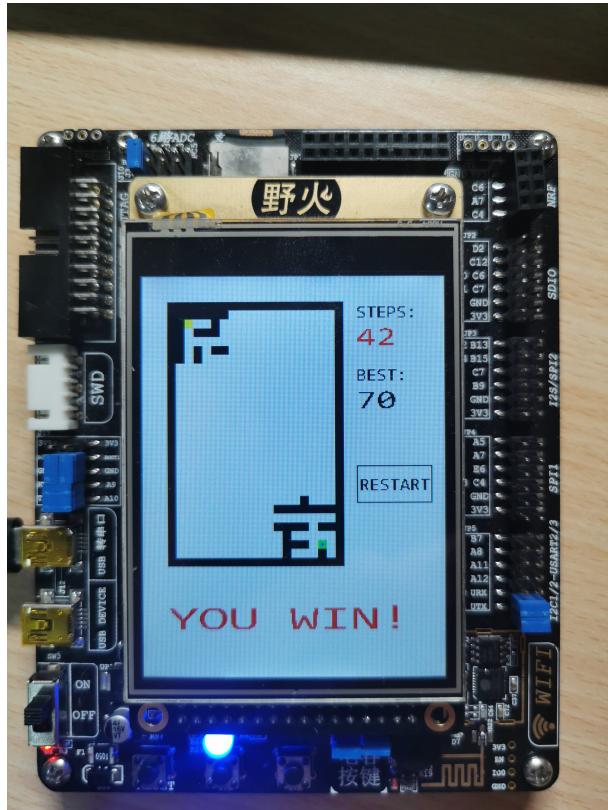


图 16: WIN_UI 界面

3.1.2 显示模块

上一小节介绍了本项目的 UI 以及 UI 逻辑模块，这一小节介绍为了实现这些 UI 模块所使用显示模块，介绍显示模块调用的基本函数以及基于基本函数实现得到主要功能函数。

3.1.2.1 基本函数

本项目在野火提供的液晶显示屏固件库的基础上进行了修改，通过修改使得基于标准库的固件库转换为基于 HAL 库的固件库。因此，我们使用的基本函数也是基于该固件库的基本函数，使用的基本函数及功能如下，这些函数在 lcd.h 文件中声明，在 lcd.c 文件中定义：

```

void  ILI9341_Clear (uint16_t usX, uint16_t usY, uint16_t usWidth, uint16_t usHeight ); // 清空某一片区域
void  ILI9341_DrawRectangle ( uint16_t usX_Start, uint16_t usY_Start, uint16_t usWidth, uint16_t usHeight,
    uint8_t ucFilled ); // 绘制矩形
void  ILI9341_DisPlayString_EN ( uint16_t usX, uint16_t usY, char * pStr ); // 显示英文字字符串
void  LCD_SetFont      (sFONT *fonts); // 设置字符串字体
void  LCD_SetTextColor (uint16_t Color); // 设置前景色
void  LCD_SetColors   (uint16_t TextColor, uint16_t BackColor); // 设置前景色与背景色

```

3.1.2.2 主要功能函数

借助修改后的固件库，我们实现了一些显示用功能函数，声明在 lcd.h 文件中，定义在 lcd.c 文件中，如下：

```
void LCD_WELCOME_Draw(); // 绘制 WELCOME_UI

void LCD_WIN_Draw(); // 绘制 WIN_UI

void LCD_MAZE_Refresh(); // 根据游戏状态更新 PLAY_UI
void LCD_MAZE_DrawMap(); // 根据迷宫状态更新迷宫
void LCD_MAZE_DrawPlayer(); // 根据玩家位置更新玩家显示
void LCD_MAZE_DrawButton(); // 绘制方向按钮
void LCD_MAZE_DrawStep(); // 绘制已消耗的步数
void LCD_MAZE_DrawRestart(); // 绘制restart按钮
void LCD_MAZE_DrawBest(); // 绘制最佳步数

void LCD_MAZE_BUTTON_Down(uint8_t type); // 当按钮被按下时，修改按钮显示
```

3.1.3 输入检测模块

本项目使用的输入检测模块主要包含两个部分——按键输入与触屏输入，输入检测模块检测对应输入，执行逻辑功能。

3.1.3.1 按键输入检测

本项目使用按键中断来进行按键的检测，在按键中断触发后，在中断的回调函数中处理按键功能，按键的中断的回调函数位于 gpio.c 文件，其中调用的函数在 key.h 中声明，在 key.c 中定义。

```
void KEY1_EXIT_Callback(); // key1按下后，更换LED的颜色
void KEY2_EXIT_Callback(); // key2在PLAY_UI按下后，会将最佳记录清零并重新开始游戏，该过程中进行了FLASH中的读取与重写
```

3.1.3.2 屏幕输入检测

与显示模块类似，本项目在野火提供的电阻屏触屏固件库的基础上进行了修改，通过修改使得基于标准库的固件库转换为基于 HAL 库的固件库。

屏幕输入检测并没有使用中断函数，而是使用了固件库中提供的有限自动机的方法，对屏幕输入的检测分为两个部分——屏幕按下与屏幕按下抬起，所有的按键功能都是在抬起的动作时进行检测与完成，检测到按下动作后仅修改被按下按钮的显示状态，处理函数在 maze.h 中声明，在 maze.c 中定义：

```
void MAZE_UI_HandelDown(int16_t x, int16_t y); // 处理按下
void MAZE_UI_HandelUp(int16_t x, int16_t y); // 处理抬起
```

每个处理函数都会根据所处 UI 状态 (ui_stat) 的不同采用不同的处理函数进行处理。总体来说，在按下后会进行判断是否按在了按键上，如果是，则将按键转换为被按下的显示形式；当抬起时，检测是否在按键上抬起，如果是，则调用相关的函数执行按键所代表的逻辑动作。

3.2 辅助功能模块

辅助功能模块主要是用于为主要功能模块的实现提供支持，包含项目中的其他所有模块，也是本项目不可或缺的一部分。

3.2.1 LED 模块

LED 模块主要用于进行 LED 操作，在 led.h 中实现，包含了三个 LED 的亮起、熄灭、反转操作：

```
// 蓝色LED操作
#define LED_B_Up      HAL_GPIO_WritePin(GPIOB, LED_B_Pin, GPIO_PIN_RESET);
#define LED_B_Down    HAL_GPIO_WritePin(GPIOB, LED_B_Pin, GPIO_PIN_SET);
#define LED_B_Toggle  HAL_GPIO_TogglePin(GPIOB, LED_B_Pin);

// 红色LED操作
#define LED_R_Up      HAL_GPIO_WritePin(GPIOB, LED_R_Pin, GPIO_PIN_RESET);
#define LED_R_Down    HAL_GPIO_WritePin(GPIOB, LED_R_Pin, GPIO_PIN_SET);
#define LED_R_Toggle  HAL_GPIO_TogglePin(GPIOB, LED_R_Pin);

// 绿色LED操作
#define LED_G_Up      HAL_GPIO_WritePin(GPIOB, LED_G_Pin, GPIO_PIN_RESET);
#define LED_G_Down    HAL_GPIO_WritePin(GPIOB, LED_G_Pin, GPIO_PIN_SET);
#define LED_G_Toggle  HAL_GPIO_TogglePin(GPIOB, LED_G_Pin);
```

3.2.2 蜂鸣器模块

蜂鸣器模块封装蜂鸣器操作，控制蜂鸣器的响停，在 beep.h 中实现：

```
#define BEEP_Up      HAL_GPIO_WritePin(GPIOA, BEEP_Pin, GPIO_PIN_SET);
#define BEEP_Down    HAL_GPIO_WritePin(GPIOA, BEEP_Pin, GPIO_PIN_RESET);
```

3.2.3 串口通信模块

串口通信是嵌入式设备调试的重要方式，虽然串口通信在本项目最后的成品并没有功能性的用途，但是在本项目开发过程中起到了至关重要的作用。在本项目中，对于串口通信，除了最基础的配置，我们主要重映射了 printf 函数，让程序在调试过程中可以很方便的使用 printf 实现串口信息的传输。

3.2.4 FLASH 模块

本项目中，FLASH 模块主要用于两个方面——触摸屏的校正参数保存与玩家最高分的保存。校正参数的保存重用了固件库的代码，我们主要实现了读取与重写 FLASH 中的玩家最高分功能。

本项目中使用一个全局 32 位 int 数组 score，定义位于 main.c，如下：

```
int32_t score[2];
```

该数组存储从 FLASH 中读取的数据，有两个元素，第一个元素是读取到的校验元素，第二个元素是最高分。

在初始化过程中，从 FLASH 读取指定地址的 8 字节数据，存储到 score 数组中，作为两个 32 位的 int 变量。若读取到的 score[0] 与程序中编入的校验元素 (SCORE_FLAG，定义在 maze.h) 相同，则认为该存储区域存储了之前存储好的最佳成绩，在对最佳成绩的值进行范围校验后，将最佳成绩的值读入迷宫状态的全局变量中，并且用于显示。若校验位错误或者最佳成绩超过了范围，则将最佳成绩清零并将正确的校验元素与清零后的最佳成绩重新写入 FLASH。

当玩家到达终点时，对玩家的成绩进行检测，若成绩优于最佳成绩，则将正确的校验元素与最佳成绩写入 FLASH 中的地址。

3.2.5 TIM 模块

本项目使用两个 TIM 计时器模块产生中断。使用计时器模块主要是为了实现 UI 闪烁与按键音频反馈。两个 TIM 计时器配置相同，均每隔 1ms 产生一次中断，中断处理回调函数功能位于 tim.h 与 tim.c，函数名为 HAL_TIM_PeriodElapsedCallback。该函数同时处理 TIM6 与 TIM7 的中断，TIM6 用于按键音频反馈，TIM7 用于 UI 闪烁。

TIM7 控制 UI 界面的闪烁，具体包括 WELCOME_UI 中“START！”的闪烁与 PLAY_UI 中玩家操作的蓝点的闪烁，闪烁周期均为 1s。为此，我们定义了一个全局变量以记录当前的显示状态：

```
uint8_t light;
```

light 仅取 0 与 1，当 TIM7 计时到 500ms 时，切换显示状态，并且将 light 取反以记录状态的改变。通过这种方式，在不同的显示状态调用不同的显示函数以实现动态闪烁。

TIM6 在检测到屏幕点击按键的抬起动作后被使能，同时使能蜂鸣器，在经过 2ms 后，停止 TIM6，并将蜂鸣器同时停止。通过这种方法，实现每次按键的音频反馈。

4 系统实现

4.1 开发环境与开发工具

本项目使用 STM32CubeIDE 开发，因此并未使用 STM32 标准库，而是使用了 Cube 提供的 HAL 库，因此在进行开发的过程中，我们对野火提供的固件库中的代码进行了修改，使其可以使用 HAL 库提供的接口，进而实现我们需要的功能。

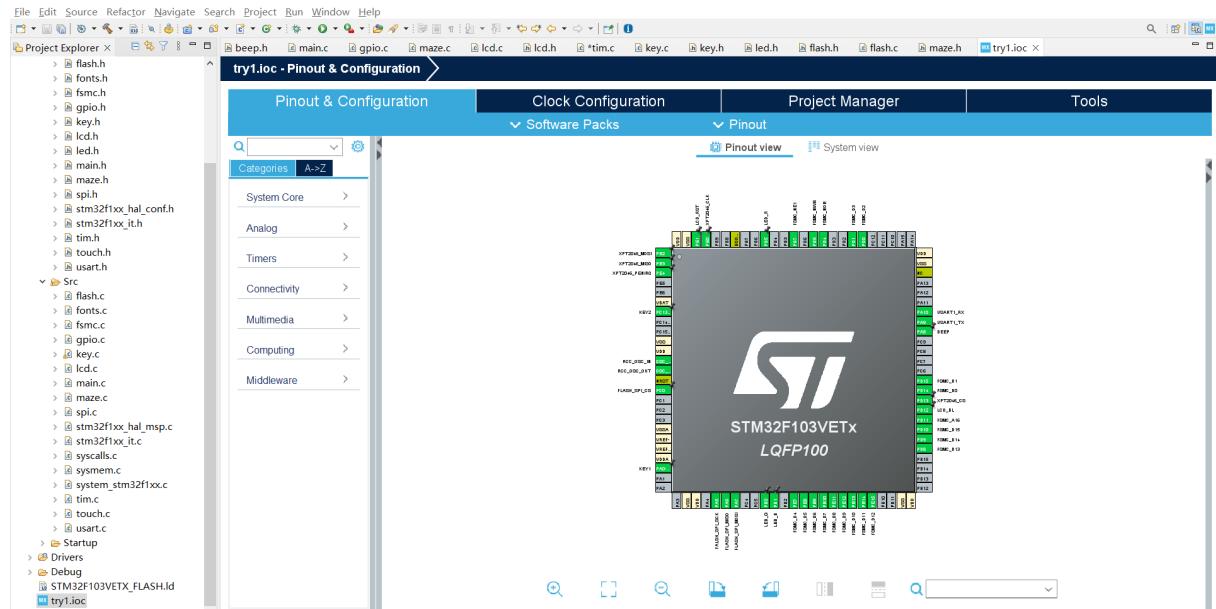


图 17: cube 界面

使用 STM32CubeIDE 开发，我们将项目编译为.hex 文件，使用 mcuisp 将程序写入单片机。

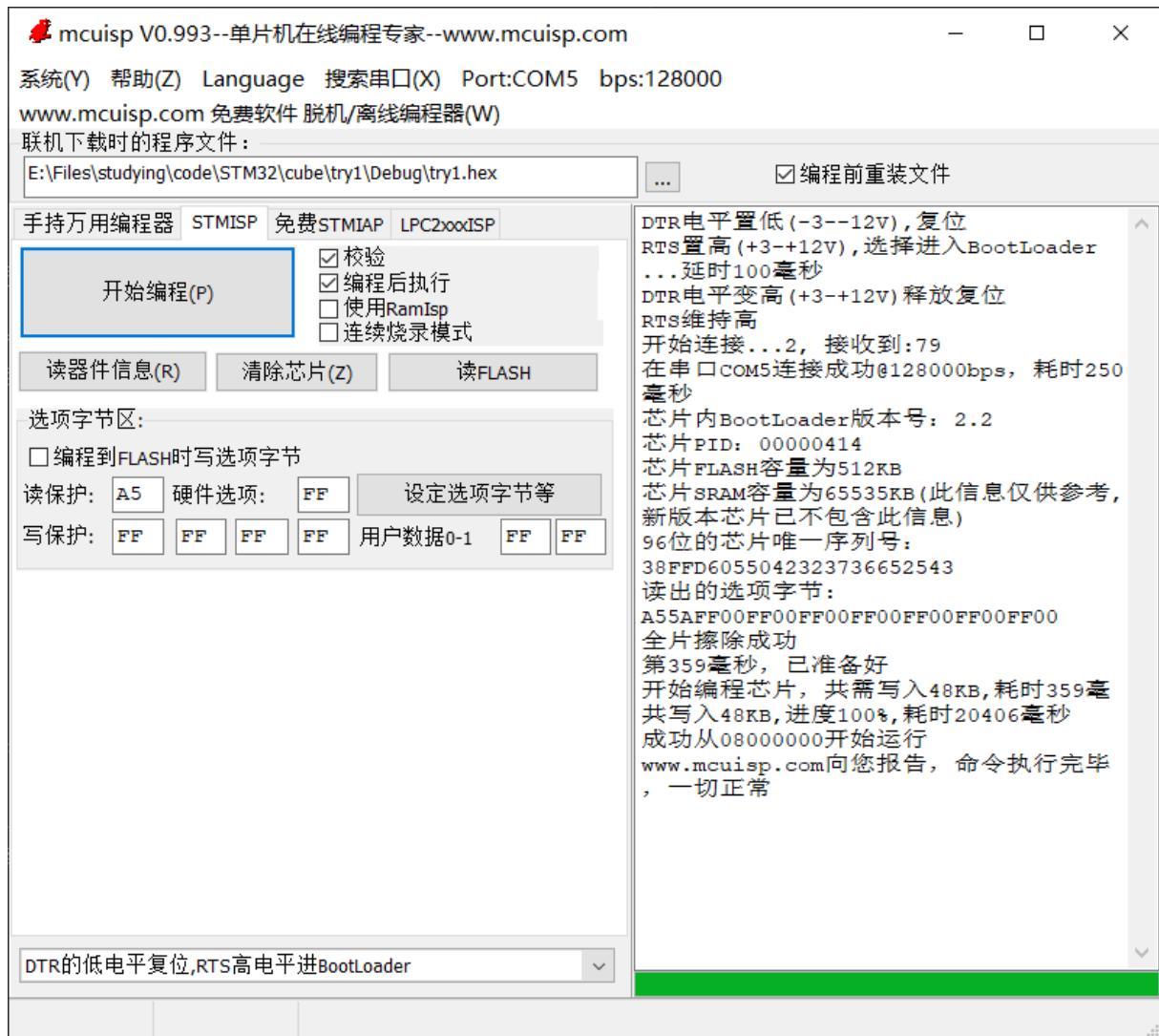


图 18: mcuisp 界面

我们使用野火串口调试助手进行串口调试。

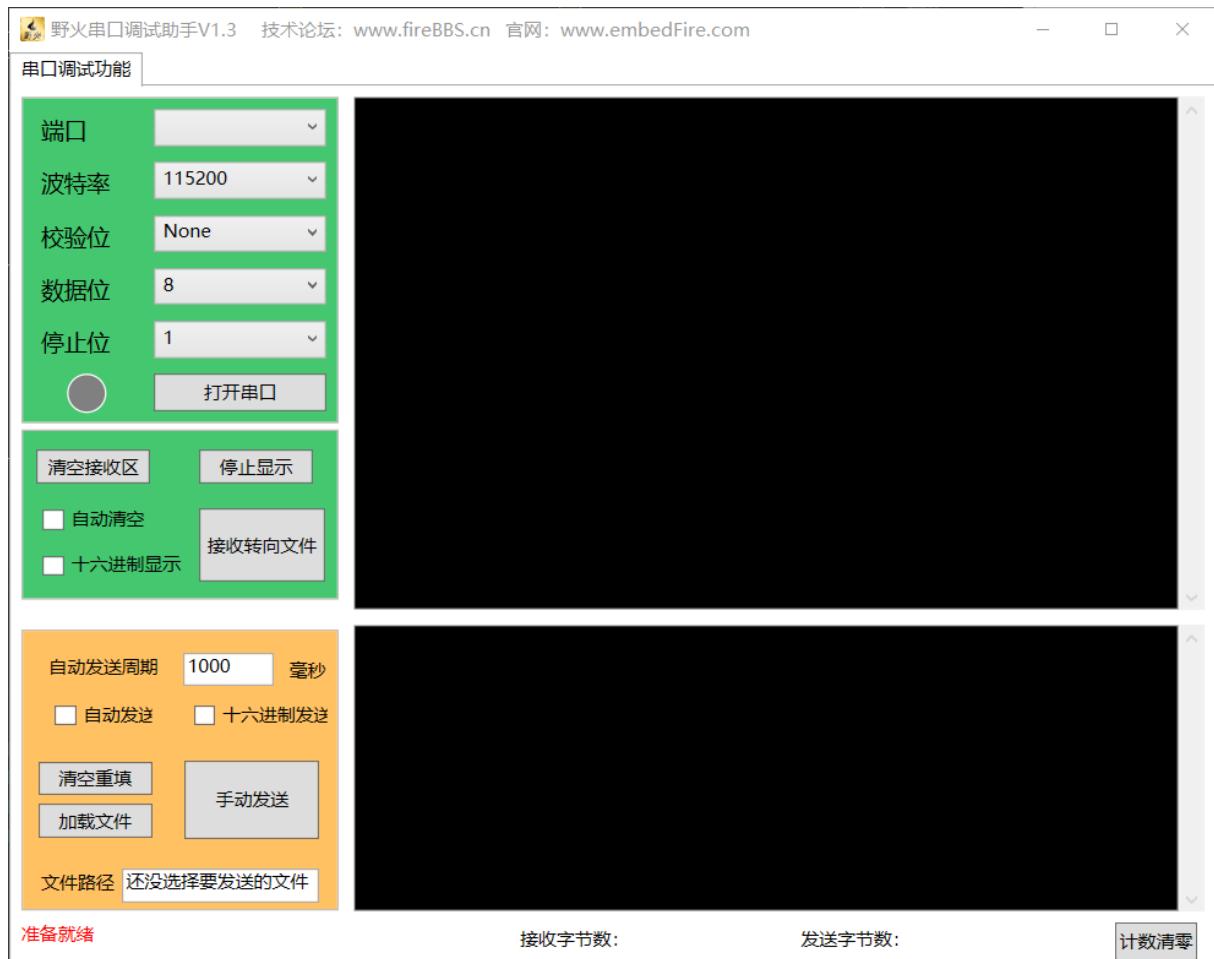


图 19: 野火串口调试助手界面

4.2 项目文件介绍

我们生成的项目文件树如图所示：

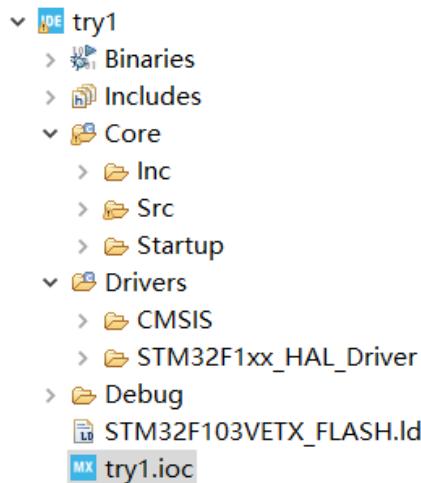


图 20: 文件树

我们编写的文件位于 Core/Inc 与 Core/Src 目录下, Core/Inc 存储头文件, Core/Src 存储源文件。

主要的源文件以及头文件介绍如下:

- main.h/main.c

程序执行的主函数, 在经过时钟、硬件、迷宫状态、UI 的初始化后, 进入循环函数, 在循环函数中调用 maze.h 中声明的 MAZE_CHECK() 函数检查玩家位置以判断是否到达终点, 使用 touch.h 中声明的 XPT2046_TouchEvenHandler() 函数检查是否有屏幕触摸事件。

- beep.h

蜂鸣器模块, 实现蜂鸣器的功能封装

- flash.h/flash.c

FLASH 模块, 实现外部 FLASH 的调用, 重用自野火提供的固件库, 我们使用 HAL 对库进行了改写, 使原先基于标准库的函数可以在基于 HAL 库使用。

- fonts.h/fonts.c

字体字模的实现。

- gpio.h/gpio.c

我们在该文件的 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) 函数中完成了按键中断的处理跳转。

- key.h/key.c

按键模块，我们在这两个文件中实现了按键中断的最终处理函数。

- lcd.h/lcd.c

LCD 显示屏模块，重用自野火提供的固件库，我们使用 HAL 对其进行了改写，是原先基于标准库的函数可以基于 HAL 库使用。除此之外，本项目的 UI 显示函数也在这两个文件中声明并实现。

- led.h

LED 模块，实现 LED 功能的封装。

- maze.h/maze.c

游戏逻辑模块，在这两个文件中定义并实现了迷宫游戏的逻辑功能，对 lcd.h 中声明的 UI 绘制功能函数进行调用，实现逻辑操作。

- tim.h/tim.c

计时器模块，TIM6 与 TIM7 的中断处理回调函数在这里实现。

- touch.h/touch.c

触摸屏模块，重用自野火提供的固件库，我们使用 HAL 对其进行了改写，是原先基于标准库的函数可以基于 HAL 库使用。除此之外，我们修改屏幕按下与抬起的处理函数，调用 maze.h 中定义的函数进行处理。

- usart.h/usart.c

串口通信模块，我们在这里对 printf 进行了重映射。

其余没有提到的源文件由 STM32CubeIDE 自动生成，我们没有进行功能性的修改。

5 项目成果与测试

5.1 项目成果

我们项目开发成果是一款探索式的迷宫游戏，以运行在野火开发版上的应用程序展示，本节展示主要界面与主要功能。



图 21: 欢迎界面

开发板上电后进入欢迎界面。该界面下方闪烁“START!”文字，LED 灯默认显示蓝色，玩家点击屏幕任意区域进入游玩界面。

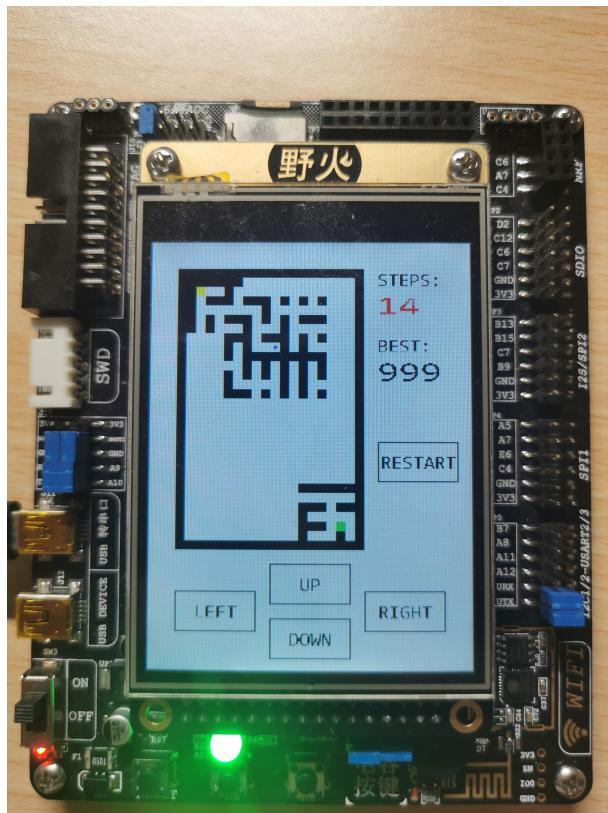


图 22: 游玩界面

进入游玩界面后，初始状态迷宫仅显示边界、黄色起点与绿色终点的部分区域以及玩家操作的蓝色小点附近的区域。

玩家可以进行的操作如下：

- 点击屏幕下方方向按钮，操作蓝色闪烁小点移动，随着蓝色小点的移动，探索迷宫区域，同时右上角记录走过的步数；
- 点击“restart”按钮，将蓝色小点返回开始位置，重置走过的步数；
- 按下 K2，重置游戏并且重置最佳成绩为 999；
- 按下 K1，改变 LED 灯的显示颜色。

如果玩家成功抵达了终点，则会进入胜利界面。

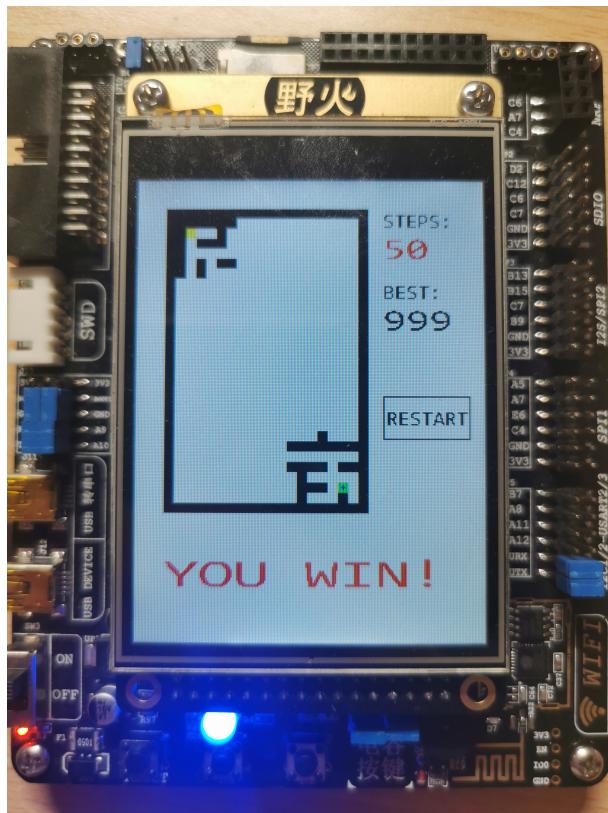


图 23: 胜利界面

进入胜利界面后，蓝点停止闪烁，方向按键消失，下方改为显示胜利语句，玩家可以通过点击“restart”重新游戏。若玩家的成绩优于最佳成绩，则会更新最佳成绩。

5.2 项目展望

本项目实现了一个基于嵌入式系统的探索式迷宫游戏，我们实现了迷宫游戏的基础功能，在本项目的基础上，有以下几点完善方向：

- (1). 迷宫存储：当前我们的迷宫是编码在程序中的，下一步可以将迷宫文件存储在 FLASH 中。
- (2). 随机迷宫：当前我们的迷宫仅有一款，为了提高游戏性，下一步可以添加随机生成迷宫算法，随机生成并保存游戏，为玩家提供更多选择。
- (3). 迷宫创作：同上一条，下一步可以考虑添加允许玩家自行创作迷宫的功能，让玩家可以直接在嵌入式设备上创作迷宫并保存。
- (4). 外设添加：受疫情影响，我们使用的外设都是片上自带的外设，如果有更多的外设就可以有更多的可能性。