N32G4FRx 系列

智能门锁解决方案用户手册 V1.1

修订历史记录

版本号	修订人	修订日期	修订描述
V1.0	产品体系	2019/11/28	初版
V1.1	产品体系	2020/05/28	根据最新硬件修正相关描述

目 录

1.1 概述

智能门锁采用高集成度、高性能、低功耗的 N32G4FRx 系列作为主控 MCU，内部集成算法组件、触控组件、安全组价、密码组件、语音组件、低功耗控制、可选蓝牙组件等多种核心资源（N32WB452x 带 BLE5.0 蓝牙组件）。方案软件搭载了 freeRTOS 实时操作系统，流畅运行 OLED 菜单显示、语音导航、指纹、触摸等多种人机交互接口，并配套可选非接触功能芯片，最大程度的满足成本和性能需求。

适用于公寓、家庭锁、智能锁、挂锁等智能家居领域。

1.2 规格和主要技术参数

技术参数：

- 电源直流： DC Max Input6.5V
- 开锁方式： 指纹、密码、钥匙、APP、卡片
- 是否搭载系统： freeRTOS
- 指纹像素： 支持多种分辨率的指纹 sensor，如 160x160、192x192
- 指纹识别性能： 最快 1:1 比对约 10ms
- 指纹模板提取： 最快 180ms
- 指纹分辨率： 508DPI
- 密码容量： 可支持 1000 个(数量可配)
- 指纹容量： 可支持 200 个(160x160，数量可配)
- 刷卡容量： 可支持 1000 个(数量可配)
- 触控按键： 12 个
- 电池续航: 18 个月
- 密码长度： 6~16 位密码，支持虚位密码开锁
- OLED 显示： 0.96 英寸，中英文显示
- 语音提示： 中文语音导航
- LED 灯： 12 个
- 开锁日志： 85 条
- 工作电流： 约 150mA
- 低功耗： MCU+TOUCH<=7uA/3.3V；
  MCU+TOUCH+NFC(3 次/S) <=28uA/3.3V；
  MCU+TOUCH+FP+NFC <=50uA/3.3V；
- 调试接口： SWD

规格：

- 运行温度： -20~60℃门锁/-40℃~105℃ MCU 芯片
- 存储温度： -45℃~85℃
- 相对湿度： 20～85%(无凝结)
- 自动待机： 15s 无操作自动待机。
- 加密方法： 支持硬件 AES/3DES/SM4 等多种加密方法
- 自动唤醒时间： 小于 300ms。
- 静电： 接触+-8KV;空气+-15KV

1.3 智能门锁方案实物图

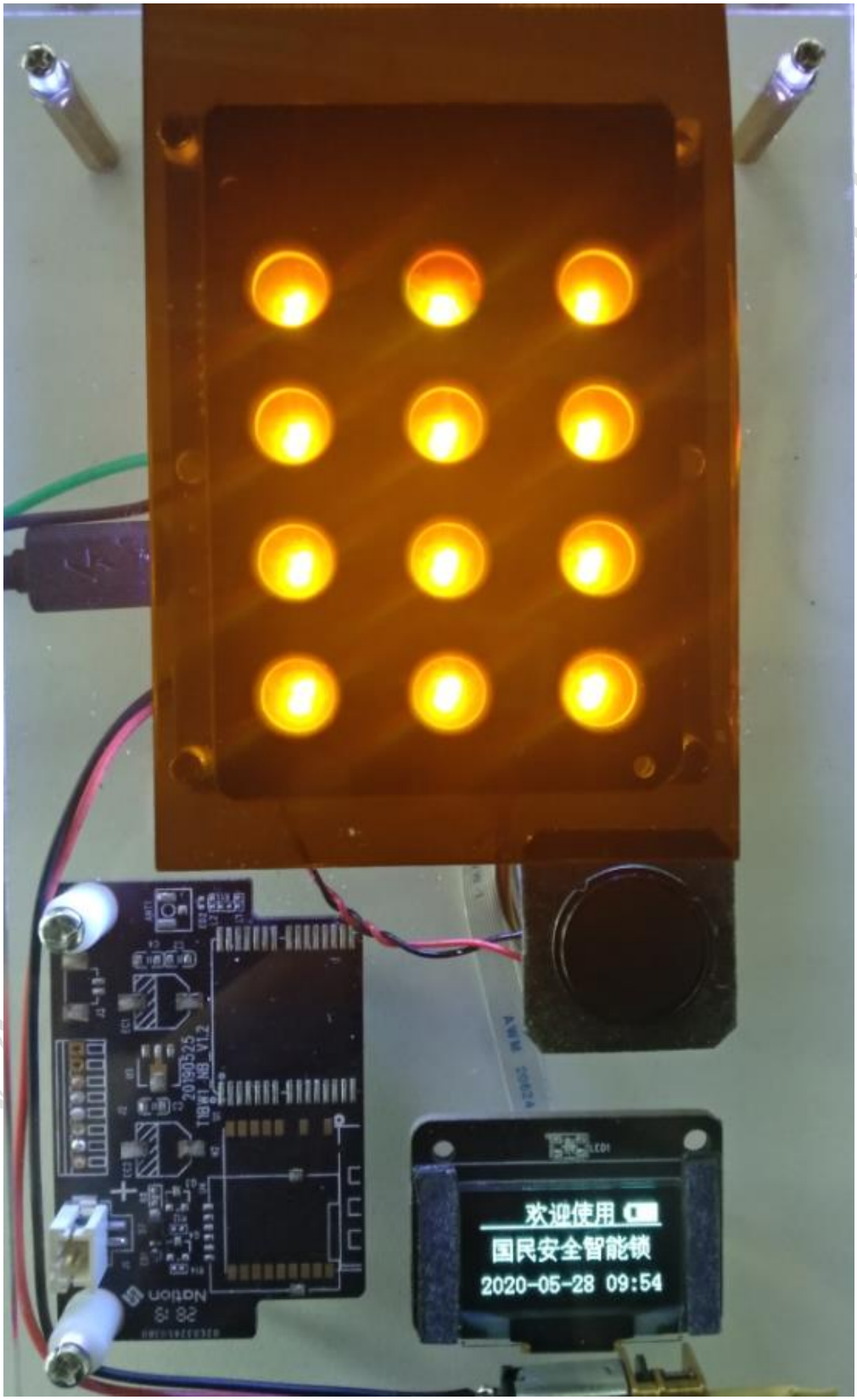


图 1 智能门锁解决方案实物

1.4 常用术语

术语	说明
FP 算法	Fingerprint 指纹算法
TSC	Touch Sensor Controller 触摸传感控制器

1.5 参考资料

《N32G4FRx 系列 32 位 ARM®Cortex-M4 微控制器用户手册》

《N32G4FRx 系列 32 位 ARM®Cortex-M4 微控制器数据手册》

2. 系统框图及功能介绍

2.1 系统框图

本方案应用的系统框图，请参考下图所示：

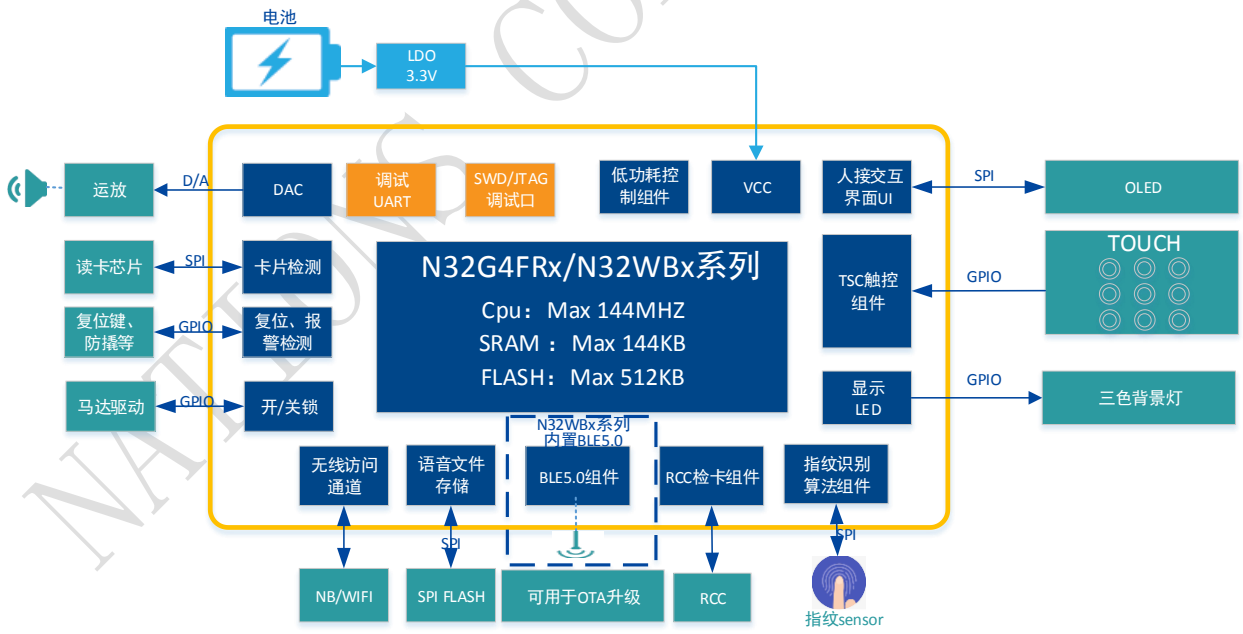


图 2 方案系统框

2.2 硬件主控板

- 主控 MCU80Pin，其中可用 67 个 GPIO;
- 尺寸：63*78MM;
- 两层板;
- BLE 板载天线，可节省结构空间
- OLED 屏 SPI 接口通讯;
- 指纹模组 SPI 接口通讯;
- 蓝牙模块 UART 串口通讯;
- NFC 检卡模块 SPI 接口通讯;
- 电机模块由 GPIO 口控制电机驱动 IC;
- 12 个触控按键由 GPIO 口控制;
- 12 个触控按键背光源由 GPIO 口控制;
- 红绿蓝 3 个 GPIO 控制的三色灯
- NB 模块 UART 串口通讯;
- 外部 FlASH SPI 接口通讯;
- 语音模块由 GPIO 控制语音驱动 IC;
- 支持低成本 DAC 语音输出
- SWD 接口;
- 防撬开关;

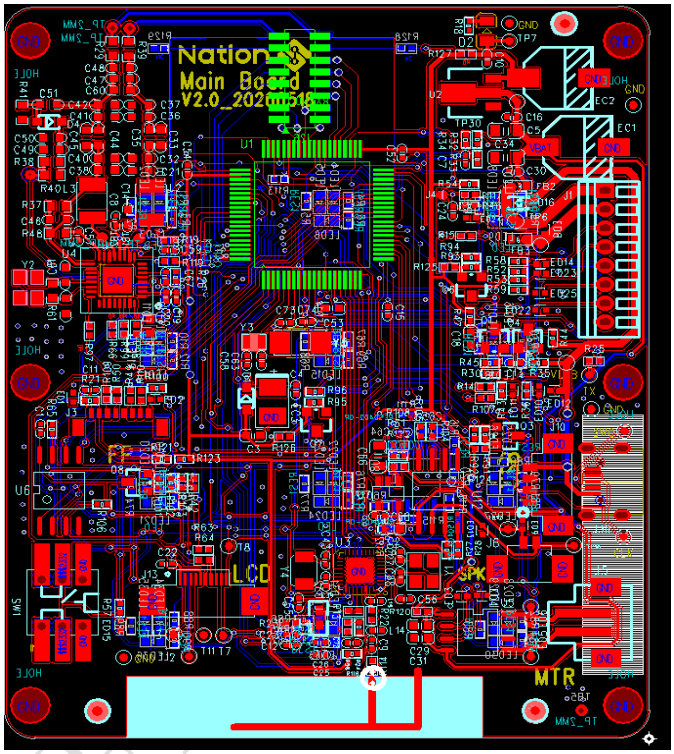
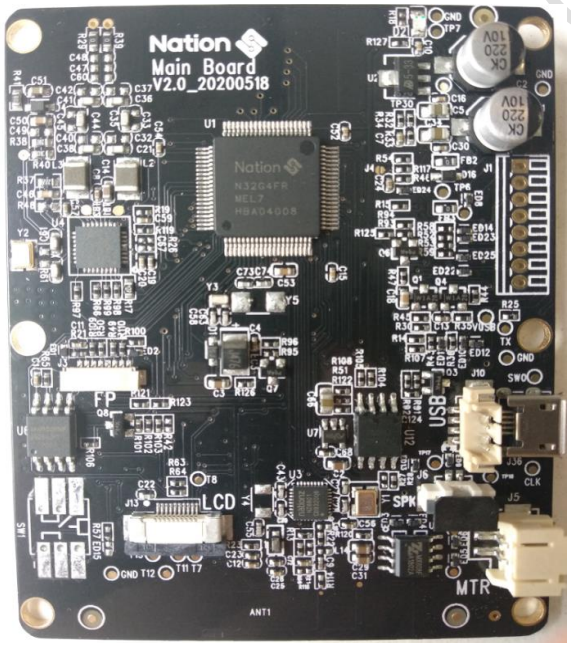
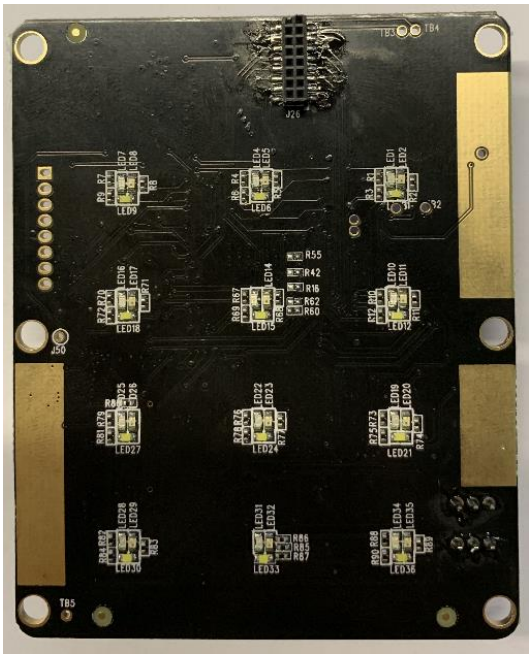


图 3 方案主控板





2.3 触控板

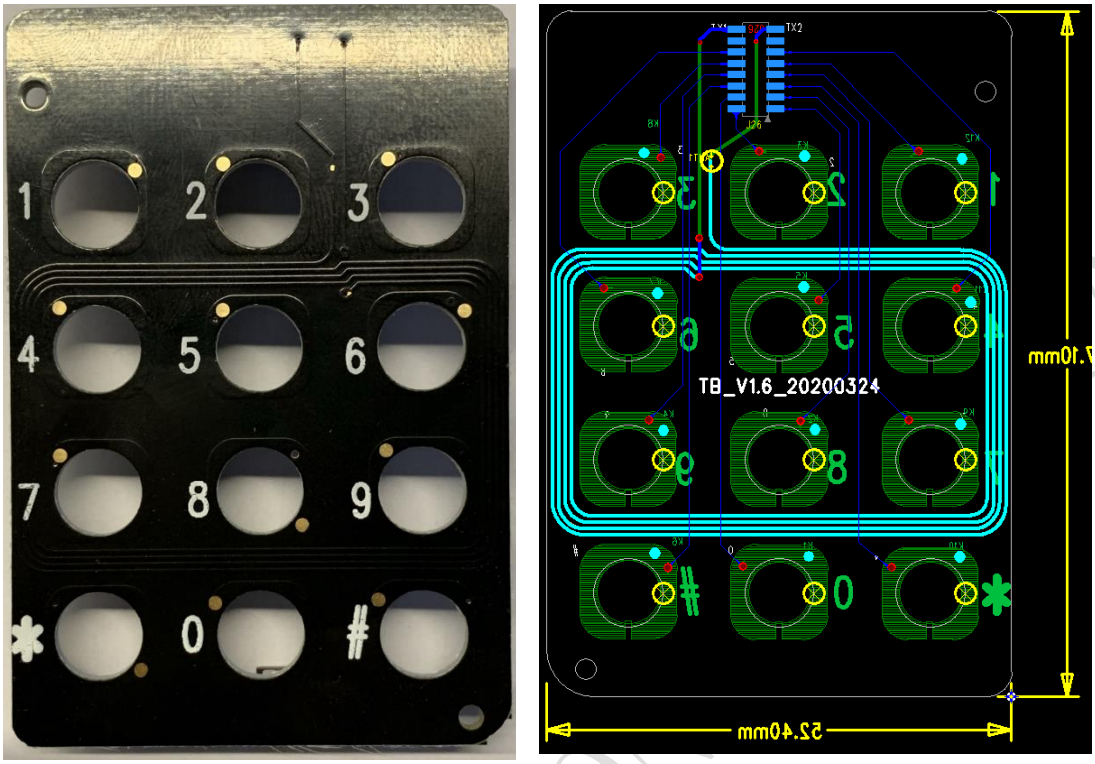


图 4 方案触控板

2.4 功能介绍

本方案为智能门锁的整体解决方案，包括主控芯片、外围模块、周边硬件电路、方案软件、以及配套设计工具、开发套件等。

具有高集成度的一体化设计，功能包含指纹、密码、NFC 卡片、蓝牙、一次性密码、机械钥匙开锁等多种开锁方式。芯片自带金融级加密算法、MPU FLASH 保护方法，提供可靠的安全保障。

体贴的人接交互界面 OLED、语音导航、及三色 LED 灯；搭载了嵌入式 freeRTOS 操作系统，加入了画面迁移引擎及画面参数数据，更加方便的画面修改、增添操作。

2.4.1 方案主要功能列表

- 具备 freeRTOS 系统、内部 TSC 触控、集成 FP 指纹算法、集成 NZ3802 NFC 检卡。
- 具备系统低功耗及唤醒(指纹唤醒/触控唤醒/NFC 唤醒/RTC 定时唤醒)。
- 具备对密码/指纹/卡片的数据管理及片上 FLASH 读写操作：

包括管理员权限下的 UI 菜单导航、LED 控及设置（包括密码、卡片、指纹账户的增加、删除）

管理员和普通账户添加/删除

恢复出厂设置、开锁记录保存。

- 具备正常模式下的 NZ8801 蓝牙连接开锁功能
- 具备电池电量检测、RTC 日历/时间显示
- 具备 SPI 接口 OLED 屏幕显示
- 具备高质量单线语音 IC 播放和省成本的 DAC 语音播放两种方式。

在 DAC 语音播放方式下：有配套的语音文件样本、制作工具、语音文件合并工具以及语音文件 SPI FLASH 下载工程、工具。

- 具备双线电机开锁/关锁控制、菜单开锁/关锁显示。

2.4.2 方案样机操作流程

本系统上电后，自动开机硬件检测，如一切检测正常则系统开始启动。

启动后可进行相关操作。默认出厂无录入任何数据，可按下“#”,添加管理员信息。

再通过管理员权限，按照 OLED 界面和语音导航提示下，添加其他用户信息，录入指纹、密码、卡片等信息。

系统默认 10s 无操作情况下，自动待机。可通过指纹、刷卡、触摸、APP 等方式唤醒操作。

详细的操作流程，请参考方案操作指南。

3. 硬件配置及电路说明

3.1 电源电路

电源电路原理图，请参见下图所示：

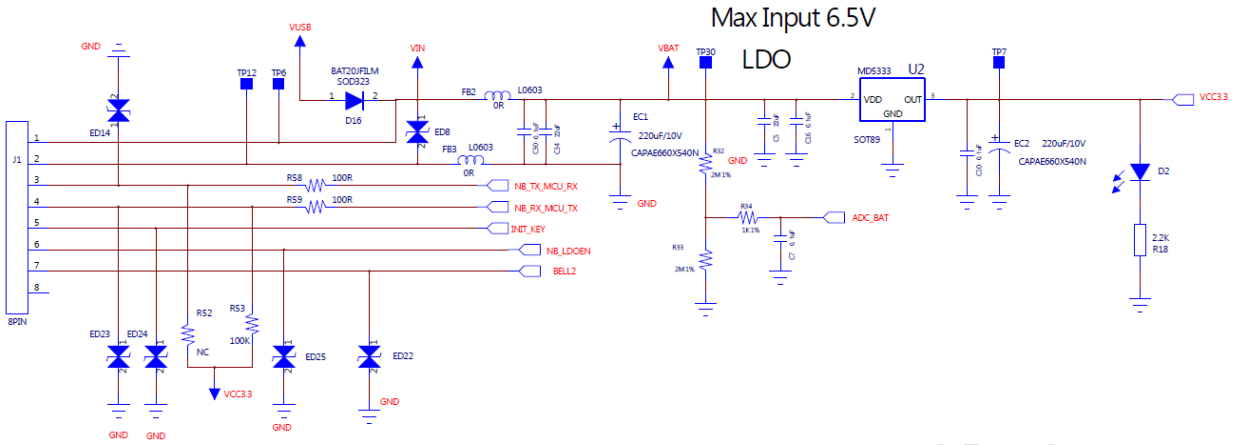


图 5 电源电路

系统上电后，软件检测电池电量及周围硬件。启动完成后，OLED 进入主页面，主页面 10s 无操作则进入待机。

本方案的电源由 4 节 1.5V 电池串联成 6V 电源供电如上图中的 VIN，经过 LDO 降压到 VCC3.3 给主控 MCU 芯片以及外围电路器件供电。

如电量不足时，可通过 VUSB 接口，由外部充电宝/移动电源，提供临时电源开锁。

注：上图电路中 NB_TX_MCU_RX/NB_RX_MCU_TX 未连线，开发者可根据实际情况将 NB 模块与 MCU 连线调试。

3.2 触控及触摸面板

本方案的触摸采用 N32G4FRx 系列自带 TSC 硬件触控，配合软件算法库实现按键触摸。触控由 MCU 主控板和触控板组成。

MCU 主控板：连接 MCU，负责触控检测；

触控板：提供触摸 PAD，接收手指感应信号。

本方案的触控由 12 个触控通道(即支持 12 触摸按键)，每个通道独立连接到 MCU 主控中的一个 PIN 脚，共 12 个 PIN 脚。

MCU 主控电路如下：

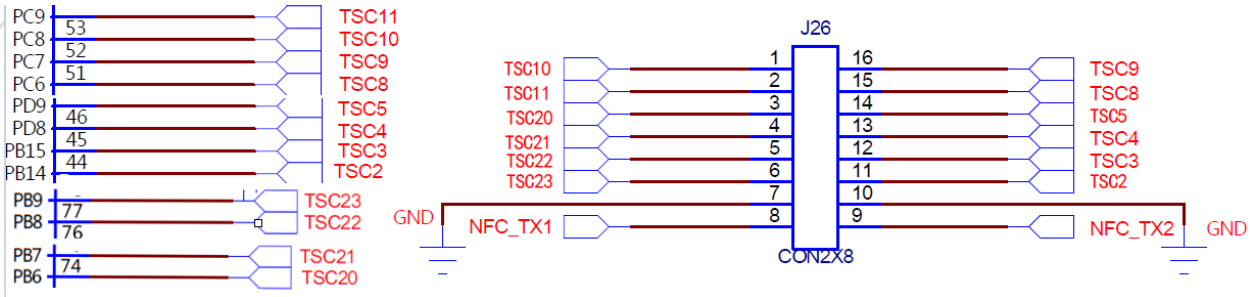


图 6 触控 MCU 主控板及连接接插件

前置面板电路如下：

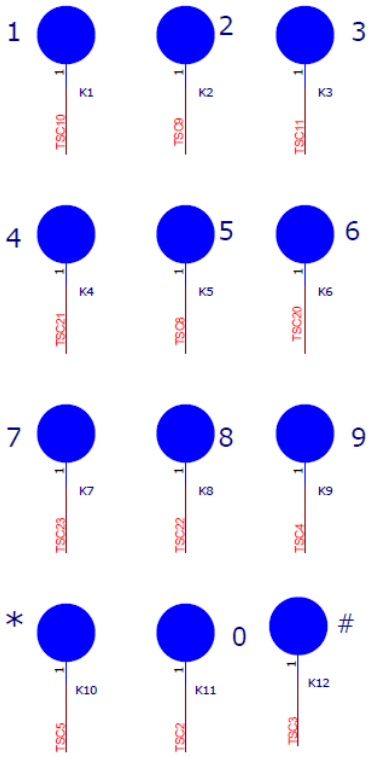


图 7 触控板

3.3 按键及 LED 灯电路

本方案在触控面板下面每个按键对应一个三色灯，每个三色 LED 灯独占 1 个 PIN，控制同时打开/关闭。而三色灯的电源网络由三个 GPIO 口进行控制。

红、蓝、白三色灯电源网络：

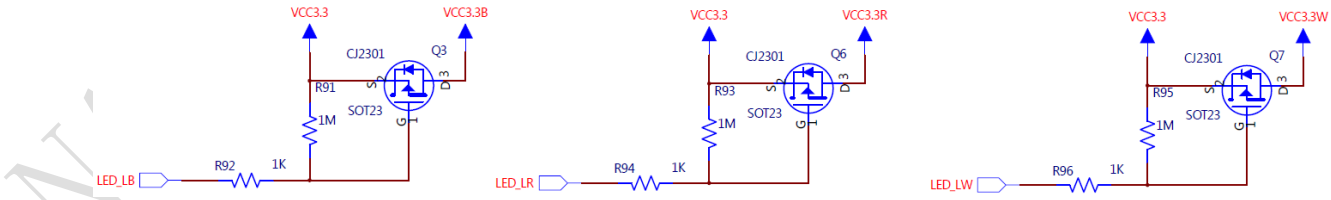


图 8 三色灯电源控制

每个灯电路如下：

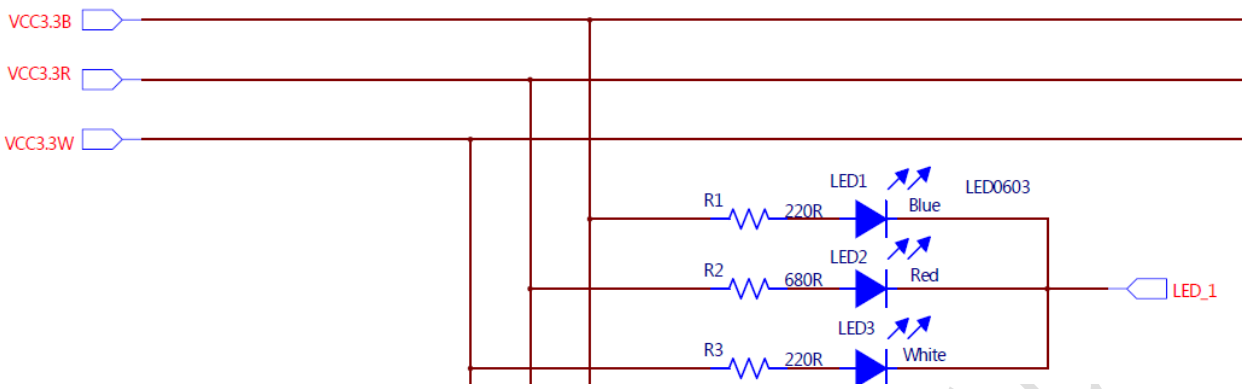


图 9 灯控制

通过 LED_1 连接到 MCU GPIO，通过 GPIO 拉低则导通。打开 LED_LB 时，LED1 会亮蓝色；打开 LED_LR 时，LED2 会亮红色；打开 LED_LW 时，LED3 会亮白色。

本方案带有一个复位孔按键，方便恢复出厂设置。电路如下：

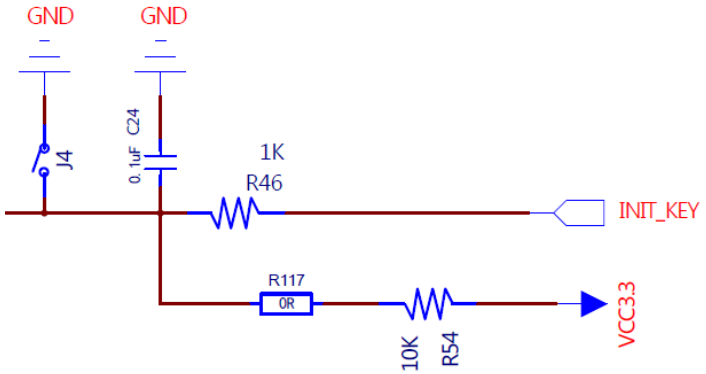


图 10 复位电路控制

当用户长按 INIT_KEY 时，MCU 端配置 GPIO 口为输入，当检测到长按时，清除系统中所有数据，恢复到出厂状态。

注：INIT_KEY 按键的安装位置为门锁的内侧。方便管理员忘记密码时，可恢复设置。

3.4 界面 OLED 显示

本方案采用 SSD1306 OLED 进行界面输出，方便用户录入指纹、设置等操作。OLED 采用无背光的方式，通过 SPI 发送相关命令控制显示。

电路如下：

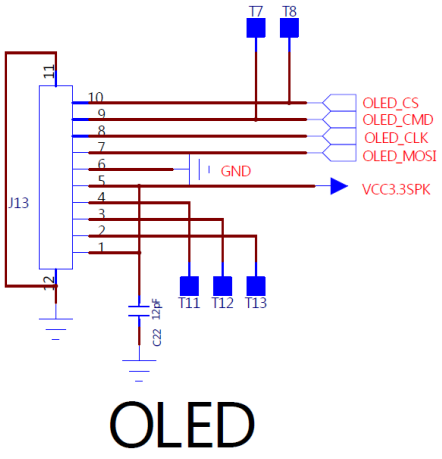


图 11 主控板 OLED 显示屏 SPI 接口电路

OLED 与 MCU 通过 SPI 进行命令控制和数据发送，其中用到了 OLED_CS（SPI 片选）、OLED_CMD(GPIO 口控制命令或数据)、OLED_CLK(SPI 时钟)、OLED_MOSI（MCU 发送数据给 OLED 数据线）

OLED 基板电路，用于与 OLED 连接，同时有固定的作用：

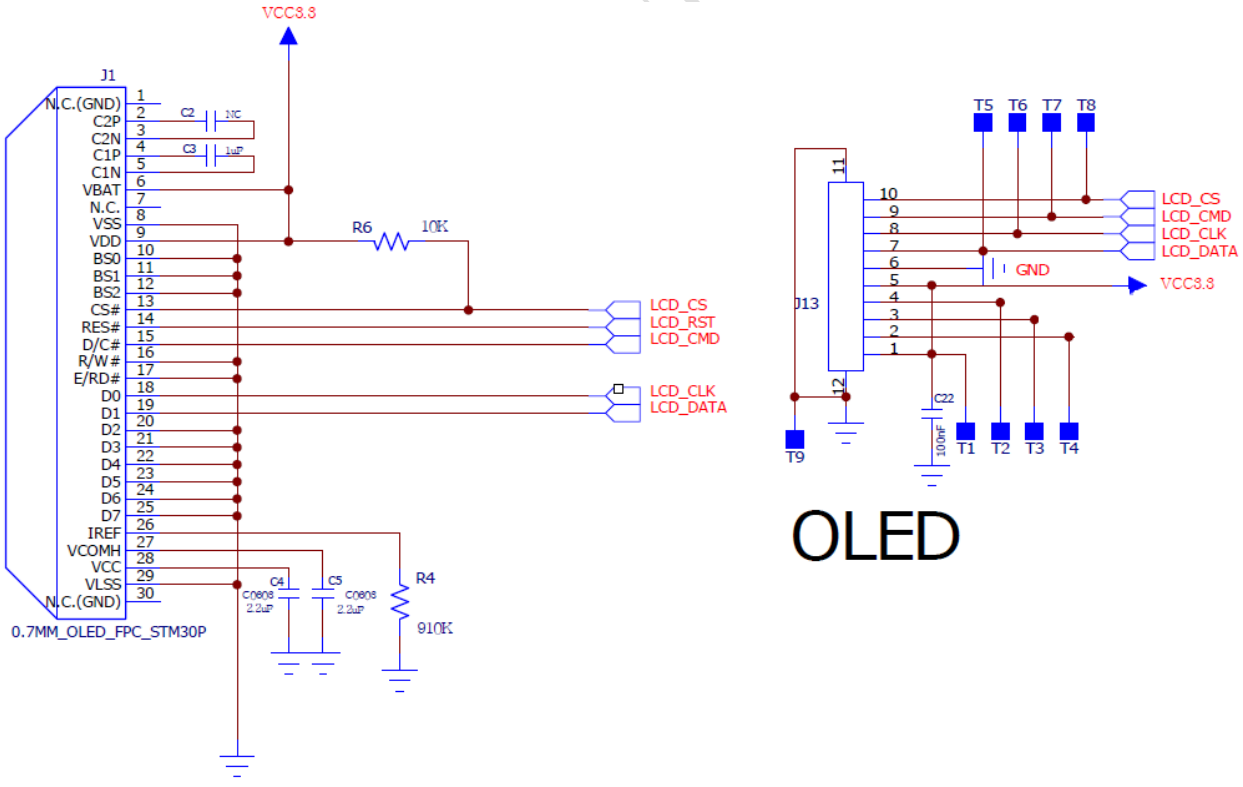


图 12 OLED 显示基板电路

OLED 屏可以支持多种接口，包括 SPI，I2C，并口线，所以 OLED 的连接线较多。本方案通过 SPI 连接，所以有效的连接线为：GND、VCC、CS#,RES#,D/C#,D0,D1

接插件图：

图 13 OLED 显示基板接插件

3.5 语音输出

3.5.1 语音 IC 方案

本方案电路支持两种方式进行选择。

如采用 SC5180B IC 输出语音导航，电路如下：

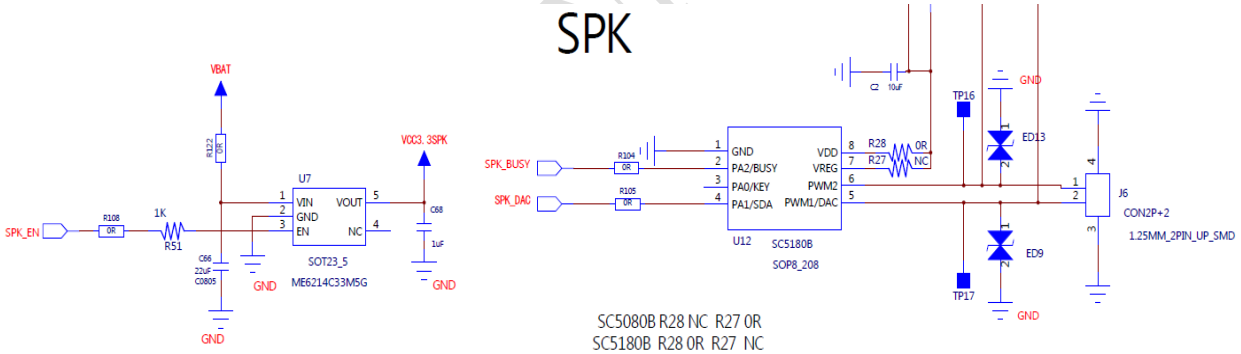


图 14 语音播放电路

选择语音 IC 播放时，将硬件电阻 R111 去掉，焊接上电阻 R105。

该语音 IC 通过单线时序，控制播放的语音序号。单线数据线为 SPK_SDA，播放一首后判断 SPK_BUSY 线是否为空闲(高电平空闲)。空闲后，继续播放下一首。

注意：在播放前，打开语音芯片电源 SPK_EN。

单线控制时序如下：

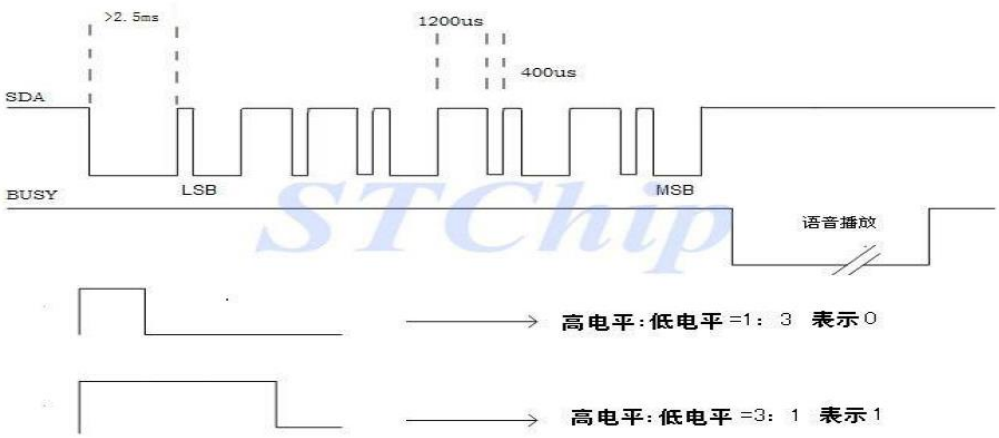
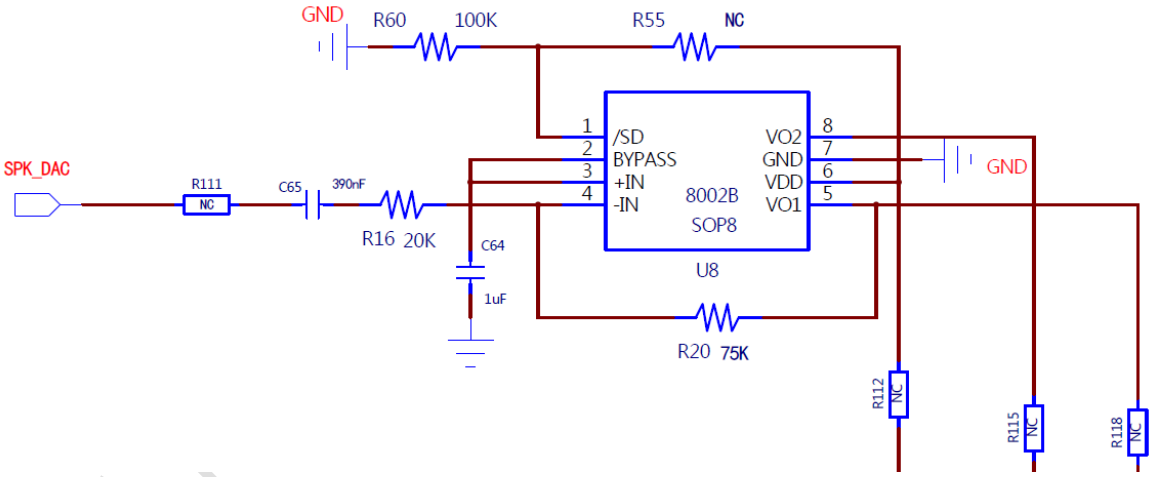


图 15 语音控制的单线协议

注：开发者可根据实际情况通过主控 MCU 的内部 12bit DAC 输出语音，尽可能的降低方案成本。

3.5.2 低成本 DAC 方案

如采用低成本 DAC 播放方案时，电路如下：



在 MCU 内部通过 DMA 读取 FLASH 语音文件，同时启动另外 DMA 将语音输出到 SPK_DAC，再经过外部 8002B 运放进行信号放大，然后 VO1 输出给喇叭发声。

3.6 指纹算法

本方案采用主控 MCU 中集成指纹算法，通过 SPI 外接指纹 sensor 就可满足指纹识别需求。sensor 为 160X160 的规格。

连接电路如下：

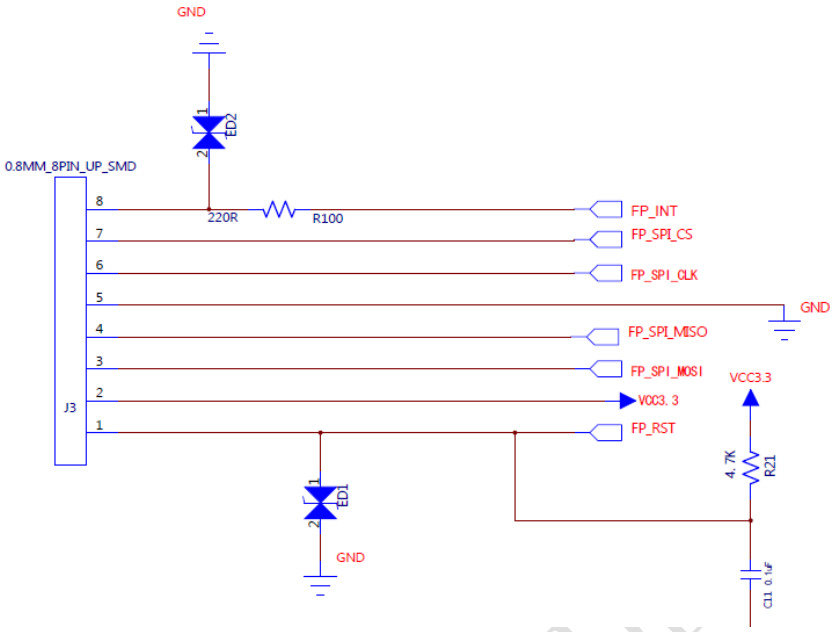


图 16 指纹算法接口电路

其中 FP_RST 为指纹 Sensor 复位脚、FP_SPI_MOSI 为 MCU 发送数据给 Sensor、FP_SPI_MISO 为 Sensor 发数据给 MCU、FP_SPI_CLK 为 SPI 时钟、FP_SPI_CS 值 SPI 片选线、FP_INT 为指纹 Sensor 中断脚。

3.7 开锁电机

采用直流电机 HSJ08，通过两个 GPIO 口(M1、M2)控制马达的正转和反转。

电路如下：

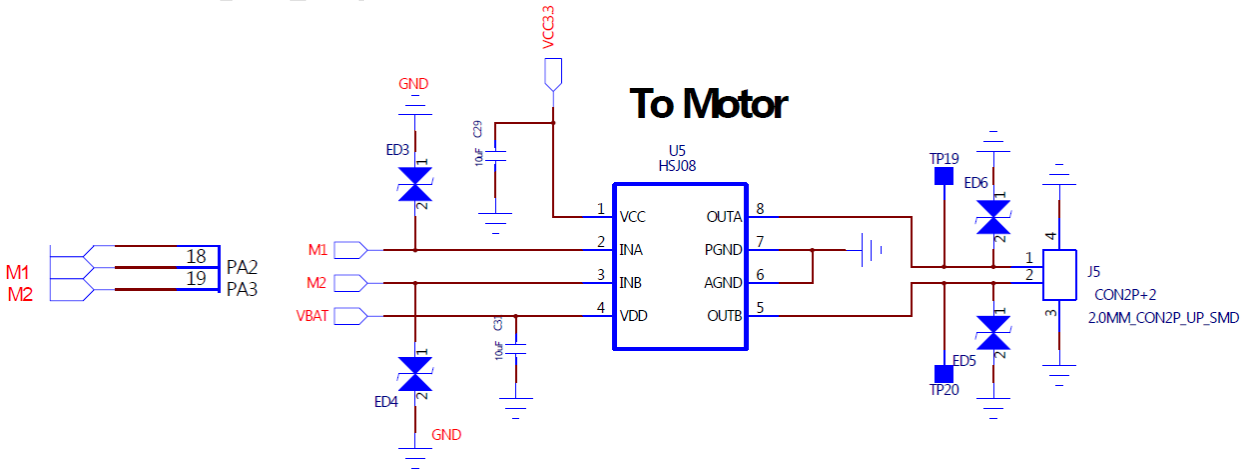


图 17 电机驱动电路

通过 GPIO 口 M1 和 M2，控制电机。真值表如下：

M1	M2	OUTA	OUTB	功能
L	L	Z	Z	待机
H	L	H	L	正转
L	H	L	H	反转
H	H	L	L	刹车

3.8 检卡

卡片检卡通过 NZ3802 芯片实现非接功能。主控 MCU 通过 SPI 与 NZ3802 进行通讯，获取卡片的信息。

电路如下：

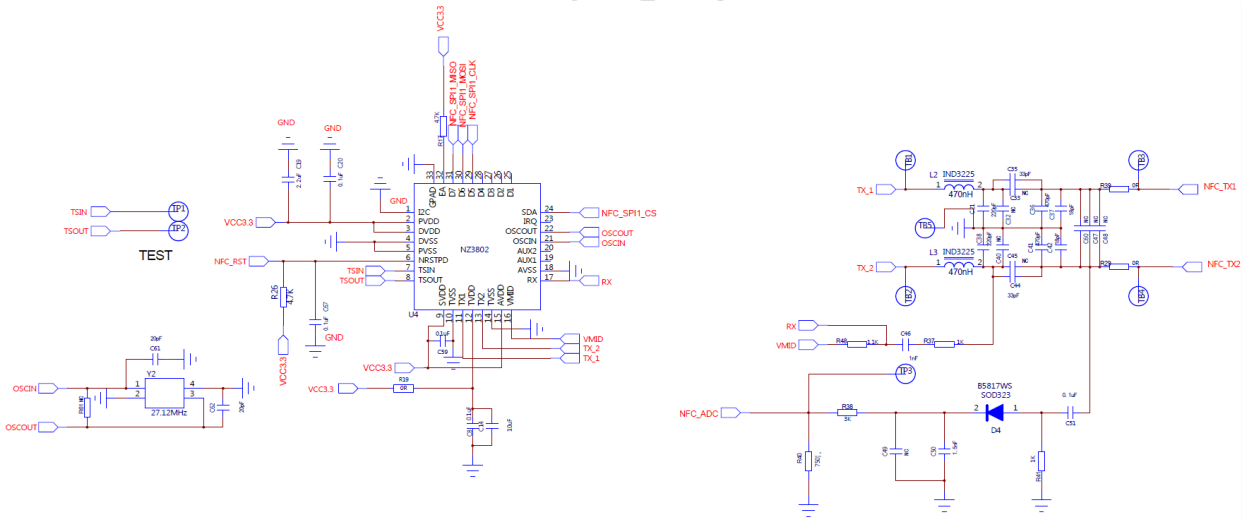


图 18 非接电路

其中的 NFC_SPI1_MISO、NFC_SPI_1_MOSI、NFC_SPI1_CLK、NFC_SPI1_CS 为 SPI 通讯接口；NFC_RST 为 NZ3802 的复位管脚。

注：NZ3802 的射频天线与 TSC 前置触控面板布局在一个 PCB 上，所以在低功耗检卡时，需要注意错开，防止有干扰。

3.9 蓝牙模块通讯

本方案支持蓝牙协议栈集成在主控 MCU 中，提高方案集成度、降低方案成本。外部通过 UART 与接 NZ8801 蓝牙射频芯片进行通讯：

电路如下：

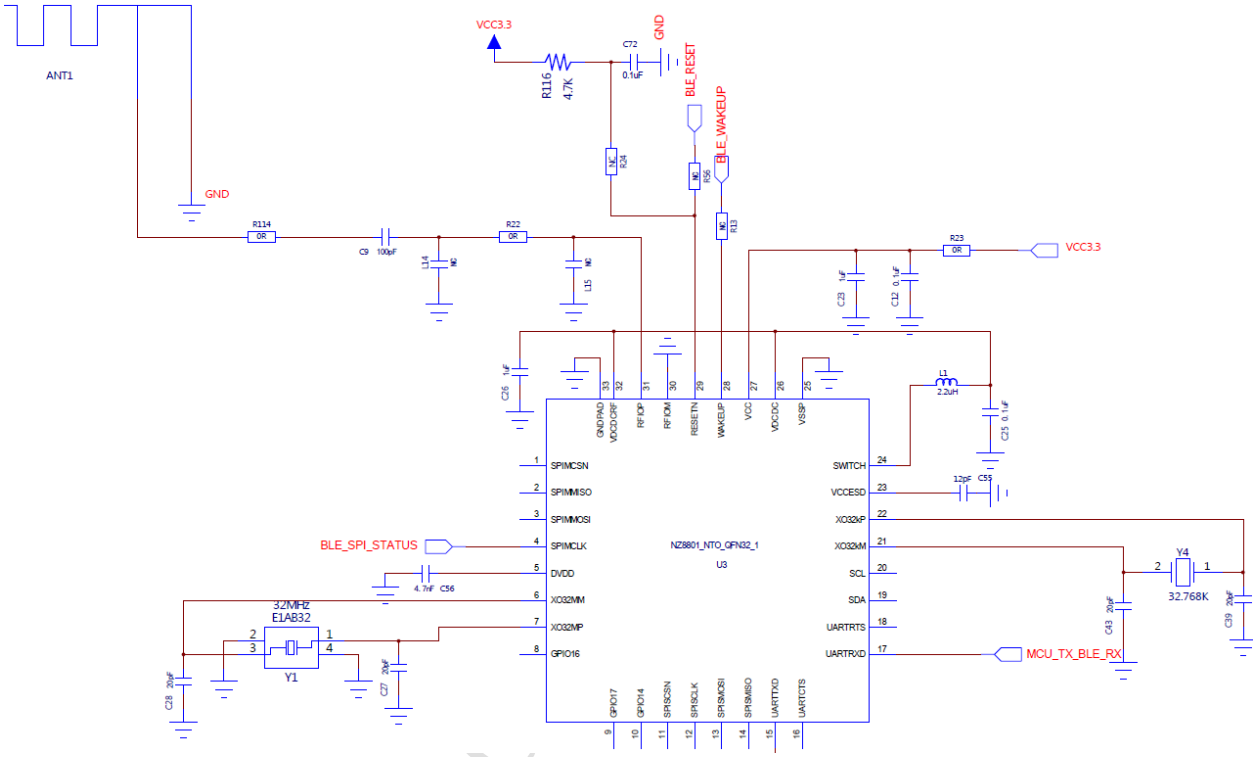


图 19 蓝牙电路

3.10 低功耗设计

本方案的重要指标是低功耗情况，硬件电路从电源网络上着重关注降低功耗设计，主要包含以下几个方面：

- 首先满足门锁方案的低功耗唤醒，包括 TSC 触控唤醒、蓝牙唤醒开锁。MCU 主控进入低功耗模式；同时，又可能的快速唤醒，STOP2 为最合适的功耗模式。
- 硬件的 3.3V 可控电源网络：语音、OLED、LED、SPI FLASH 等，在低功耗下，关闭其电源降低功耗。
- 硬件的 3.3V 常电网络：蓝牙、NFC 检卡、指纹，模块自带低功耗模式，功耗较低。
- 软件在进入低功耗时，关注各 MCU 管脚的状态配置；保持与外部期间的电压一致，防止电流消耗。

4. 软件系统

4.1 系统简介

本方案为智能门锁提供高集成度、低成本、高性价比的整体解决方案，包括主控 MCU 芯片、外围器件、软件系统实现，为门锁用户提供直接设计参考。

4.2 软件系统架构

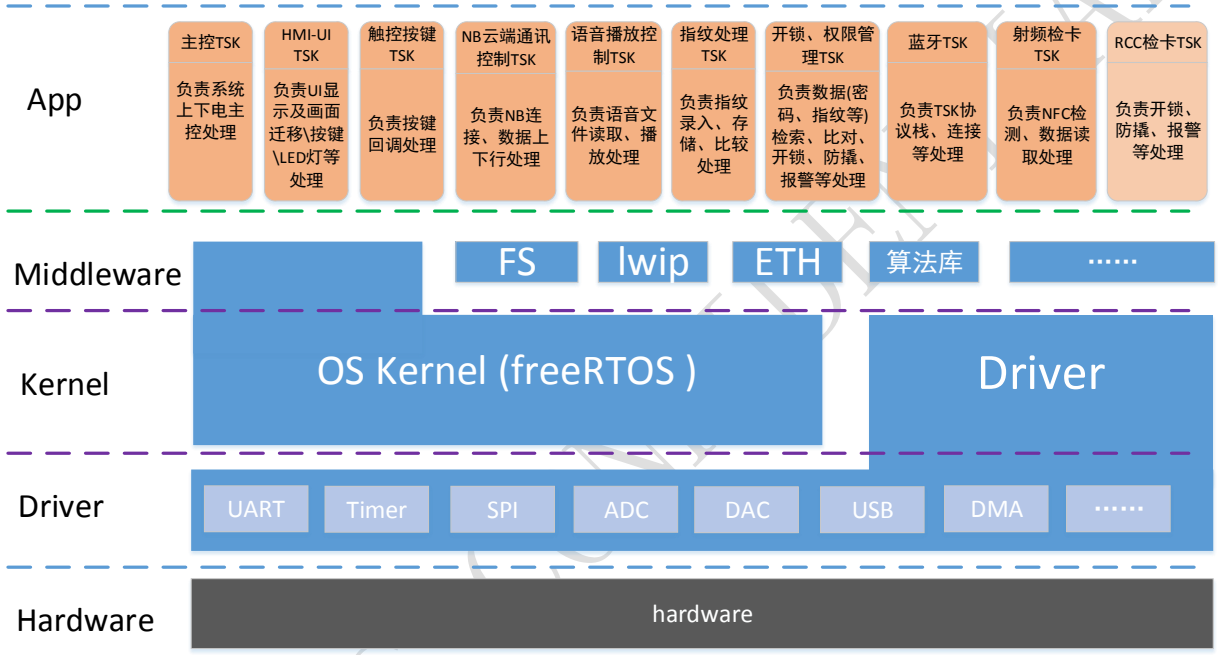


图 20 智能门锁方案软件系统架构设计

4.3 功能描述

智能门锁方案：

硬件部件：电源 LDO&晶振、主控 MCU、指纹 Sensor、SPI FLASH、语音 IC&喇叭、电机驱动&直流电机、OLED 显示屏、NZ8801 蓝牙射频芯片、NZ3802 检卡芯片、按键及 LED 灯。

软件架构：基于硬件电路上的软件架构，主要包括，MCU 片上各模块驱动、指纹算法/蓝牙协议栈/TSC 触控组件、嵌入式 freeRTOS 系统、及由各功能模块对应的 APP TASK。

功能列表参考本文档的第 2.4 功能介绍章节。

5. 快速开发指南

5.1 概述

智能门锁方案是基于 N32G4FRx 系列芯片及配套 SDK 进行用开发。

从上到下主要包括以下几个部分：

应用层：包括智能门锁的各 TASK 的业务逻辑控制。TASK 是按照外部功能模块进行划分的，便于后续的功能移植、扩展、裁剪。

系统层：本方案使用了开源免费的 FreeRTOS v10.2.1 作为软件系统，最大程度的隔离底层，方便应用层开发。

算法&驱动层：N32G45x 系列芯片配套 SDK，包含了片上各模块的驱动接口及参考范例。

硬件：基于 ARM Cortex-M4 内核、各功能模块的控制器、智能门锁电路及周边器件。

5.2 工程环境及文件说明

本方案的软件开发采用 keil5(5.26.2.0)进行编译，开发前请安装 N32G45x 系列 pack 包。

工程文件及功能说明，如下图所示：

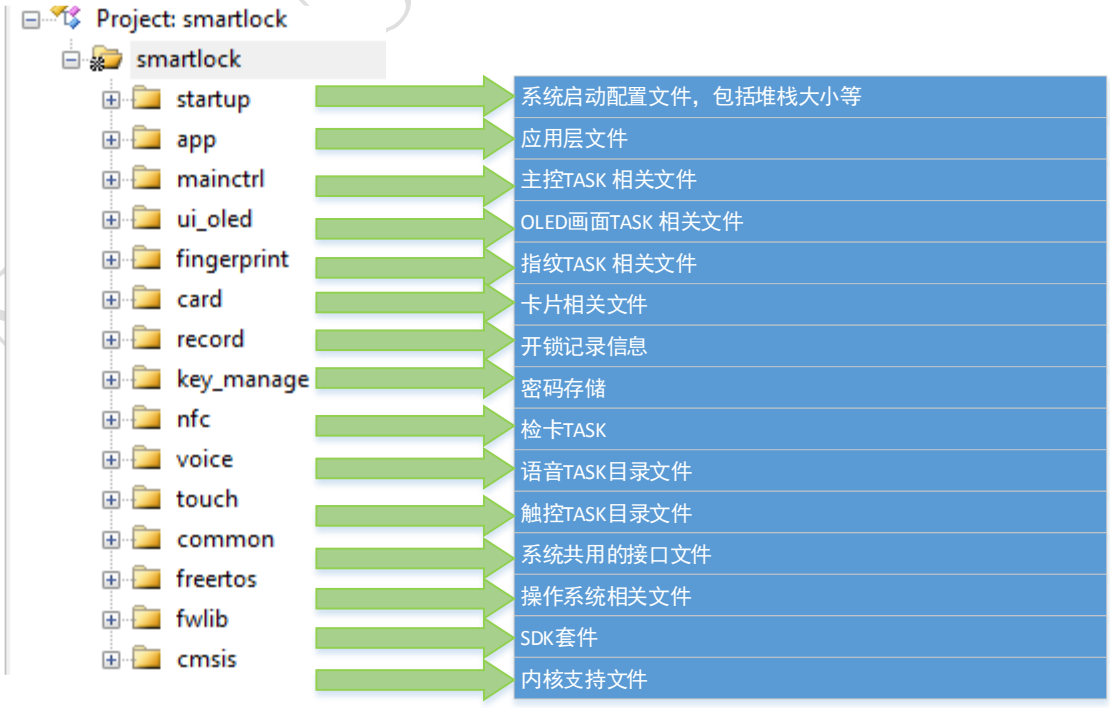


图 21 工程文件功能说明

5.3 应用主控任务实现：

如下所示：

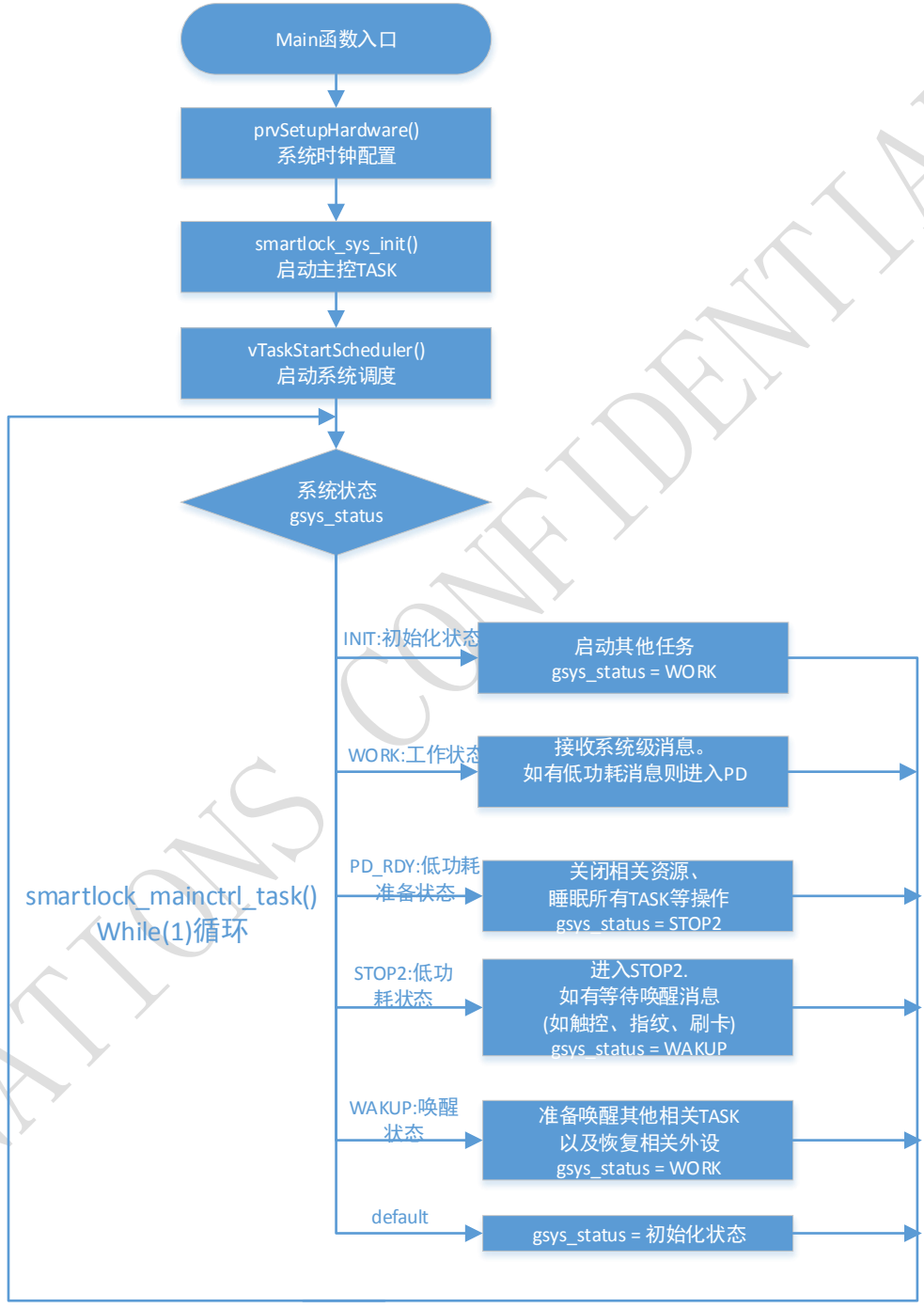


图 22 主控 TASK 控制

5.4 各功能模块实现

各功能逻辑 TASK 负责各自内部的逻辑实现，与其他 TASK 通讯尽量通过发送 OS 消息或接口，最大程度的提高功能内聚，降低各功能模块之间的耦合是一种好的设计方法。

5.5 画面 UI 显示及迁移

UI 画面显示 TASK 相对与其他 TASK 工作量相对较多，而且迁移、交互相对较复杂的功能模块。从便于修改、移植，强内聚性的思路，将所有画面按照事件驱动的方式进行显示和更新。

因 UI 画面也是最容易进行修改的模块，所以采用独立的驱动引擎+画面，由其他消息驱动画面引擎显示或更新。

设计思路是：

A) 将每一个独立的画面清晰的画出来，类似思维导图的方式。
B) 然后设计一个画面驱动引擎。系统启动后，画面引擎定位在首页画面。
C) 如有指纹或其他消息时，驱动当前画面处理消息，并迁移到下一个画面。

5.5.1 当前画面显示及处理

每一个画面有自己独立的属性和处理方法，画面数据结构如下：

```
typedef struct display_page_t
{
    uint32_t number;                          /* 画面编号-对应画面迁移图中保持一致 */
    pdisplay_init_handler pf_init_handler;    /* 初始化显示及功能处理 */
    pdisplay_rt_handler pf_rt_handler;        /* 实时处理函数 */
    uint8_t *led_index_list;                   /* 当前画面对应的LED灯显示数组 */
    display_line line_1;                      /* 第一行显示字符及语音 */
    display_line line_2;                      /* 第二行显示字符及语音 */
    display_line line_3;                      /* 第三行显示字符及语音 */
    display_line line_4;                      /* 第四行显示字符及语音 */

    line_exec_condition condition[DISPLAY_OLED_LINENUM]; /* 下一个画面的前提条件 */
} display_page;
```

画面的填充数据，如下：

```
// 输入需删除的用户ID-画面16
const display_page menu_del_input_id_user = {
    .number = DISPLAY_DEL_INPUT_ID_USER,

    .pf_init_handler = NULL, //本画面的执行函数
    .pf_rt_handler   = display_del_verify_user_handler,
    .led_index_list = "0,1,2,3,4,5,6,7,8,9,*,#",

    /* 根据消息类型，则可以迁移到下一个对应的画面 */
    .condition[4].msg_type           = MSG_T_TOUCH_KEY,
    .condition[4].pmsg_value         = "*", //触控按下*符号，迁移到下一个画面
    .condition[4].next_display_index = DISPLAY_DEL_INFO,

    .condition[5].msg_type           = MSG_T_NO_OPER_TIMEOUT,
    .condition[5].pmsg_value         = "", //超时退回到上一个画面，如无上一个画面，则进入低功耗
    .condition[5].next_display_index = DISPLAY_DEL_INFO,

};
```

说明如下：

画面的处理函数(结合上图说明)：

.pf_init_handler；为刚进入画面时的初始化处理函数。

.pf_rt_handler;为画面实时处理函数

画面属性配置(结合上图说明)：

.condition[4].msg_type；表示本画面收到触控按键消息类型

.condition[4].pmsg_value；表示本画面收到触控消息中的“*”按键值

.condition[4].next_display_index；表示本画面收到消息后，迁移到下一个画面的序号。

举例说明：

- 有些画面只是简单的显示菜单，则只配置显示字符“1.语言设置”和调用通用的画面显示函数 display_common_menu_handler()。例如：

```
// 语音&语言设置-画面33
const display_page menu_display_settign_language_voice_param = {
    .number = DISPLAY_SETTING_LANGUAGE_VOICE_PARAM,

    .pf_init_handler = display_common_menu_handler, //本画面的执行函数
    .pf_rt_handler   = NULL,
    .led_index_list = "1,2,*",

    .line_1.chindes_str            = "1.语言设置",                          //中文字符串
    .line_1.pf_ch_str_handler      = NULL,                                  //中文处理函数
    .line_1.english_str            = "1.Language",                          //英文字符串
    .line_1.pf_english_str_handler = NULL,                                  //英文处理函数
    .line_1.voice_list_addr        = (uint32_t)VOICE_LIST_CHINESE_INDEX32100, //播放的语音序号
    .line_1.attr                   = 0,                                     //其他属性
```

因在此画面下没实时处理的需求，所以设置接口 pf_rt_handler 为 NULL

有些需要实时处理，则需要实现当前画面下的处理函数。例如

```
// 动态密码模式设置-画面41
const display_page menu_display_dyn_pswd_setting = {
    .number = DISPLAY_DYN_PSWD_SETTING,

    .pf_init_handler = display_setting_dyn_pswd_handler, //本画面的执行函数
    .pf_rt_handler   = display_setting_dyn_pswd_handler,
    .led_index_list = "1,2,*",

    .line_1.chindes_str            = "1.打开动态密码",                      //中文字符串
    .line_1.pf_ch_str_handler      = NULL,                                  //中文处理函数
    .line_1.english_str            = "1.Dynamic On",                        //英文字符串
    .line_1.pf_english_str_handler = NULL,                                  //英文处理函数
    .line_1.voice_list_addr        = (uint32_t)VOICE_LIST_CHINESE_INDEX32400, //播放的语音序号
    .line_1.attr                   = 0,                                     //其他属性
```

5.5.2 画面迁移

门锁方案的信息录入、参数设置都需要结合 OLED 画面进行显示、交互。因此将所有画面按照思维导图的方式描述出来，并对照迁移图进行修改画面数据结构中的参数，很方便也高效，尽可能的少编写代码、多一些配置的方式。

以下是所有画面及迁移(注:本方案代码也严格参照如下画面进行迁移)：

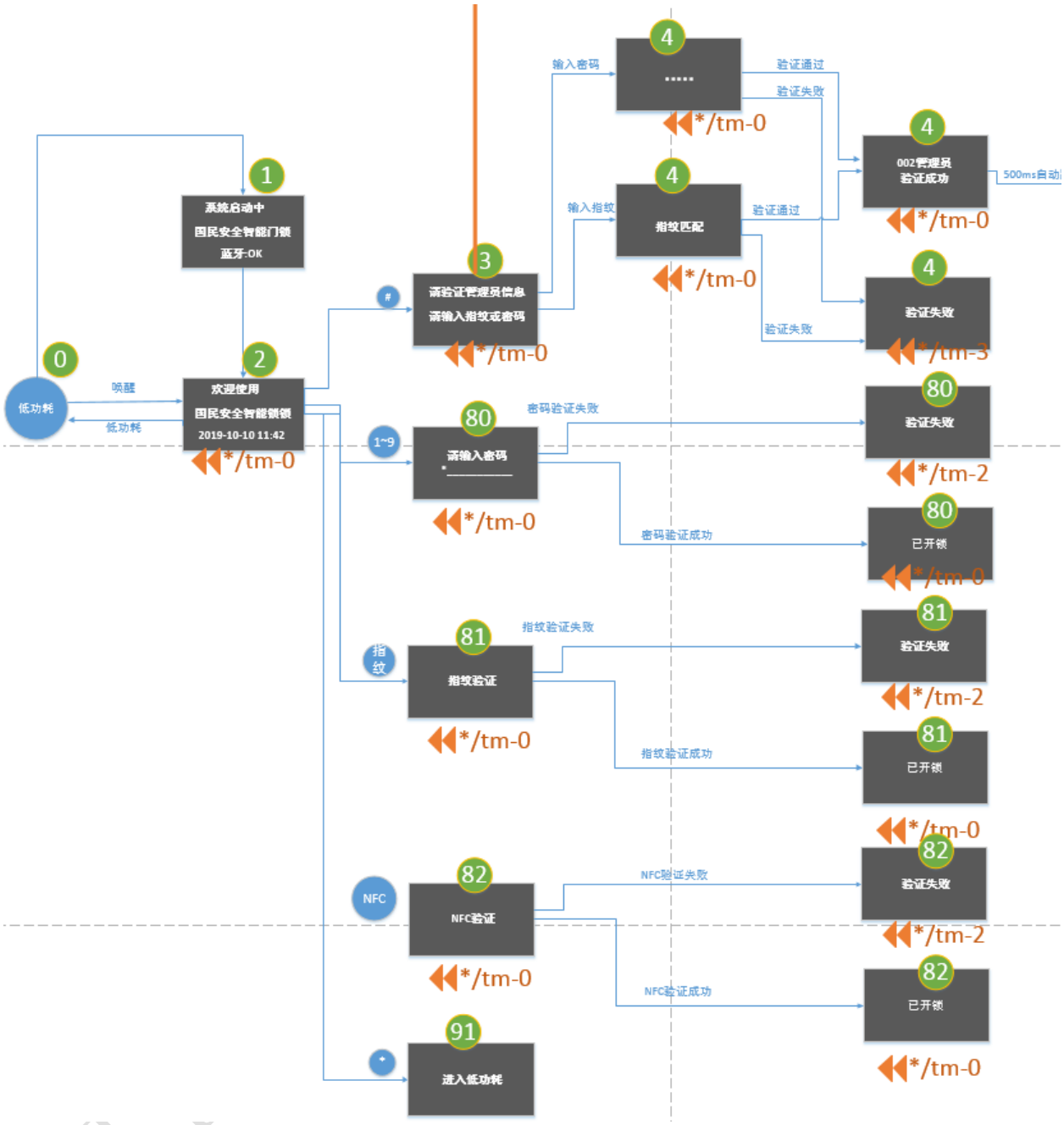


图 23 画面迁移

详细画面请参考附带的画面迁移图源文件。

从设计上，请遵守代码的画面个数、配置属性和文档画面迁移图保持一致，方便修改和移植，因此也得出先设计，后编程实现。

5.6 触控按键

本方案的触控采用 MCU 自带的硬件 TSC 及配套算法而实现。采用单独的触控 TASK 获取按键消息，并将消息发送给其他 OLED UI TASK 进行显示和处理。

TASK 交互时序如下：

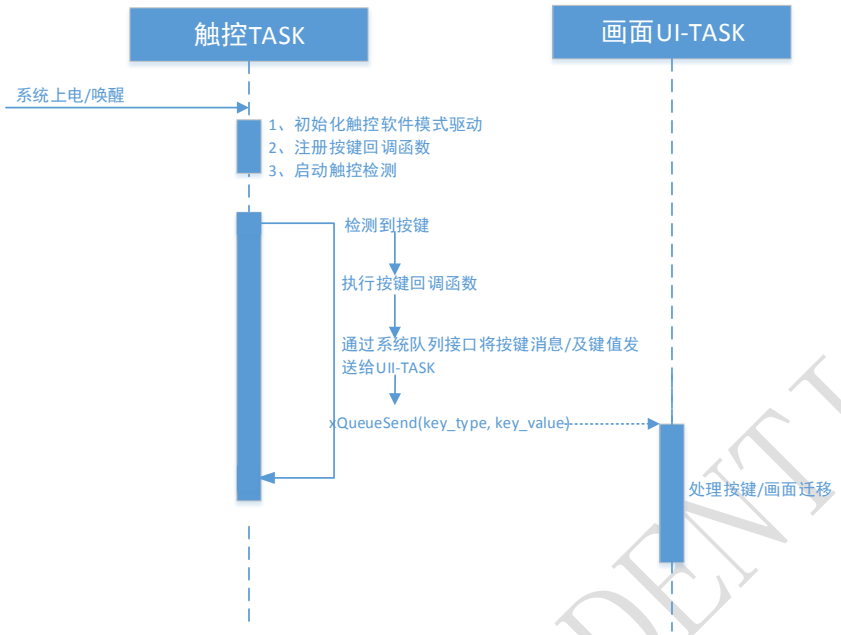


图 24 触控 TASK 触控消息接收及发送控制

注：TSC 触控组件的使用，请参考其他文档。

6. 参考与索引

6.1 FAQ

Q:【触控按下后没反应】
A:【请确认软件开启的触控通道号是否与实际硬件电路中 TSC 通道号保持对应】
详细参考《应用笔记 AN20200422-N32G45x_G4FRx_N32WBx 系列触控设计指南.pdf》。

Q:【触控按下灵敏度差】
A:【请确认 PCB 走线是否满足设计要求；适当提高软件的灵明度配置】
详细参考《应用笔记 AN20200422-N32G45x_G4FRx_N32WBx 系列触控设计指南.pdf》。

Q:【息屏后立即测量功耗偏高】
A:本方案在息屏后，不会立即进入低功耗。主要是便于，约 15s 内无操作才会进入低功耗。