项目编号: 001

**项 目 开 发 记 录**

**项目名称: 智能门锁**

**立项部门:**

**项目领域:**

**申请日期:**

**结项日期: 2023.11.04**

# 目录

[目录 2](#_Toc29806)

[一、 项目介绍 3](#_Toc26468)

[1、 项目需求 3](#_Toc27780)

[2、设计方案 3](#_Toc9869)

[3、相关技术点 3](#_Toc12772)

[4、预计效果 3](#_Toc25207)

[二、设备开发 3](#_Toc7209)

[1.语音播报模块 3](#_Toc833)

[2. Wifi上传数据功能开发 3](#_Toc28188)

[3.AT24C04简介 3](#_Toc30487)

[4. RTC实时时钟 3](#_Toc16820)

[5. LCD显示 3](#_Toc10910)

[6. lash存储 4](#_Toc26956)

[7. 字库下载 4](#_Toc1516)

[8. 指纹模块 4](#_Toc5949)

[9. freeRTOS实时操作系统 4](#_Toc26485)

[10rfid模块 4](#_Toc31875)

[三．项目结项 4](#_Toc4686)

[1.该项目能让自己有什么收获 4](#_Toc4955)

[2.总结项目中遇到的问题，及解决方法 5](#_Toc10264)

# 项目介绍

## 项目需求

## 1、开锁方式

## 指纹识别开锁:支持注册和识别多用户指纹,快速精准识别

## 密码开锁:支持自定义多组数字密码开锁

## 刷卡开锁:支持感应式IC卡,添加和删除卡片用户

## 远程App开锁:支持App链接智能门锁,提供远程开锁接口

## 用户管理

## 指纹管理:支持注册和删除多用户指纹,指纹容量不少于100个

## 刷卡用户:支持添加和删除卡片用户,卡片容量不少于100张

## 密码管理:设置6位管理员密码,支持自定义修改;设置6位开门密码,支持自定义修改

## 日志记录

## 开锁日志:记录所有开锁方式、开锁时间,支持存储至少1000条

## 报警日志:记录门锁被破坏、误操作等异常情况

## 远程控制

## 用户管理:App端可远程添加删除指纹和卡片用户

## 密码修改:App端可远程修改管理员密码和开门密码

## 远程开锁:App端提供远程开锁接口

## 扩展功能

## LCD显示:支持128\*64像素LCD,显示开锁状态、时间等信息

## 语音提示:支持语音芯片,提示开锁成功等信息

## 时间同步:支持RTC时钟,可校准时间

## 安全与稳定

## 防撬防破坏:锁体采用全钢材质,抗撬强度不小于1000kg

## 断电保护:配置电源保持电路,掉电后可保证至少24小时正常开锁

## 稳定可靠:续航时间不少于1年,开锁故障率小于千分之一

## 低功耗

## 睡眠模式:无操作10秒后进入睡眠模式

## 唤醒方式:指纹置于传感器、刷卡、密码输入可唤醒

## 设计方案

该项目旨在设计一款智能锁系统，通过使用阿里云服务器作为云端平台，通过MQTT协议实现与智能锁的通信。用户可以通过手机APP或Web应用程序远程控制智能锁，实现开锁、上锁、更改密码等功能。

硬件设计：

主控单元：选择Cortex-M3或Cortex-M4微控制器作为主控芯片，负责与各个传感器和执行器进行通信。

输入模块：包括指纹识别模块、电容按键模块和RFID射频卡模块，用于身份验证和开锁操作。

输出模块：包括LED显示屏和蜂鸣器，用于状态显示和报警。

通信模块：使用ESP32 WIFI模组实现与阿里云服务器的连接，通过TCP/IP协议进行数据传输。

软件设计：

嵌入式程序：使用C语言编写嵌入式程序，实现与各个硬件模块的通信和控制逻辑。

阿里云物联网平台：注册阿里云账号，并创建物联网平台产品设备。通过MQTT协议将设备接入阿里云服务器，并配置相应的订阅主题和发布主题。

APP或Web应用程序：开发手机APP或Web应用程序，与阿里云服务器进行通信，实现用户对智能锁的远程控制和管理功能。

## 相关技术点

1、MQTT协议：MQTT（Message Queuing Telemetry Transport）是一种轻量级的发布/订阅消息传输协议，适用于物联网设备间的通信。在该项目中，使用MQTT协议实现智能锁与阿里云服务器的通信。

2、嵌入式系统开发：通过选择适当的嵌入式处理器和编程语言（如C语言），开发与智能锁硬件模块交互的嵌入式系统。

3、主控芯片选择：使用Cortex-M系列微控制器作为主控芯片，它们具有低功耗、高性能的特点，适合用于物联网设备。

4、传感器和执行器：项目中涉及指纹识别模块、电容按键模块、RFID射频卡模块等传感器，以及LED显示屏、蜂鸣器等执行器。需要了解这些模块的工作原理和接口通信方式。

5、ESP32 WIFI模组：使用ESP32模组连接阿里云服务器，通过TCP/IP协议进行数据传输。需要熟悉ESP32模组的配置和使用方式。

6、阿里云物联网平台：注册阿里云账号并创建物联网平台产品和设备。通过阿里云物联网平台，将智能锁设备接入到阿里云服务器，并配置相应的订阅和发布主题。

7、JSON数据格式：在项目中，使用JSON（JavaScript Object Notation）格式对数据进行编码和传输。需要熟悉JSON的语法和使用方式。

8、前端开发：开发手机APP或Web应用程序，与阿里云服务器进行通信，实现用户对智能锁的远程控制和管理功能。可以使用开发框架如React Native或Vue.js。

1. FreeRTOS - 多任务系统构建、任务管理、任务通信机制

10、Flash存储器 - 工作原理、串口编程、存储管理、加密算法

## 4、预计效果

1、远程控制和管理：用户可以通过手机APP或Web应用程序远程控制智能锁，实现开锁、上锁、更改密码等功能。不再受到地理位置限制，随时随地都能控制智能锁的状态。

2、实时通知和报警：智能锁连接到阿里云服务器，可以实时将锁的状态信息传输到云端。用户可以即时收到开锁、锁定状态的通知，确保家庭、办公室或商店的安全。

3、安全性提升：通过指纹识别、电容按键和RFID射频卡等身份验证方式，智能锁可以实现更高的安全性，防止非法入侵。用户可以随时管理和更新授权人员的信息，提升安全性管理能力。

4、数据分析和统计：通过阿里云服务器提供的数据分析和统计功能，可以对智能锁的使用情况进行统计分析。例如，可以了解谁、何时使用了智能锁，并生成相应的数据报告。

5、用户体验增强：传统的钥匙锁存在操作不便、易丢失等问题，而智能锁可以通过一键开锁、密码管理、远程控制等功能，提升用户的使用体验。

6、商业应用潜力：该项目具有广阔的商业应用前景，可应用于家庭门锁、办公室门禁、酒店客房等场景。基于阿里云的智能锁系统也可作为一个创新的产品，进一步拓展市场和商机。

## 二、设备开发

## 1.语音播报模块模块介绍

“智能锁开发平台”板载一个语音芯片和一个 8 欧 1W 的喇叭，可以通过语音芯片和喇叭来播放 40 段语音 提示信息以及门铃音乐等。语音芯片内的语音数据出厂时已经固化，用户不能自行烧录改变。

**使用规则**

## 1.语音芯片通过PA1口输入8位语音编号,根据编号播放对应语音。

## 2.PB0口输出高低电平表示语音播放状态。

## 3.模块可以直接连接喇叭播放语音。使用时,MCU需要按要求的时序通过PA1口发送语音编号,然后芯片会播放相应的语音内容。

## Wifi上传数据功能开发ESP32模块介绍

ESP32是一款由乐鑫科技推出的高度集成的低功耗系统芯片(SoC)，用于物联网和嵌入式应用。它是ESP8266的升级版，提供了更强大的功能和更高的性能。ESP32集成了双核处理器，即可运行主频高达240MHz的主核（Xtensa® Dual-Core 32-bit LX6），还有一个低功耗的协处理器。这使得ESP32能够同时处理密集的计算任务和实时任务，可以说是在嵌入式领域中如日中天的一颗明星芯片。在无线通信方面，ESP32还集成了Wi-Fi和蓝牙模块，支持2.4GHz的Wi-Fi 802.11 b/g/n协议和蓝牙4.2。它可以作为无线网络的节点，与其他设备进行通信，也可以作为无线控制器与手机或其他蓝牙设备进行互动。除了强大的处理和通信能力，ESP32还提供了丰富的外设接口，包括多个通用输入输出管脚(GPIO), UART, I2C, SPI等等。这些接口使得ESP32可以连接到各种传感器、执行器和外部设备，实现与周边设备的数据交换和控制。**AT指令函数封装**

关于esp32的AT指令集封装主要可以分为STA配置和AP配置两个，主要流程包括编写一个串口发送函数，通过串口向esp8266发送AT指令，并将该模块的返回值通过串口1打印在串口助手上，并在串口发送函数的基础上再次封装，即在串口发送后来判断其返回的字符串中的固定字符，来判断AT指令是否成功，并在发送AT指令后适时延时来等待模块执行指令，并通过该函数的层层嵌套来实现对esp32的配置

## AT24C04简介

AT24C04是一种串行电路（I2C）总线EEPROM（电可擦可编程只读存储器）。它由微芯科技（Microchip）生产，具有4千字节（Kb）容量，适用于各种电子设备和嵌入式系统中存储小量数据的需求。

AT24C04的主要特点如下：

1、容量：AT24C04具有4千字节的存储空间，可以存储约 512 个8位字节数据。这使得它非常适用于存储配置信息、用户数据、传感器数据等小容量数据。

2、电子可擦除：AT24C04是一种电可擦可编程存储器，意味着它可以通过电子擦除来改变存储的数据。这相比传统的ROM（只读存储器）更加灵活，可以对存储的数据进行更新和修改。

3、I2C接口：AT24C04采用I2C串行总线接口，这是一种双线制接口协议，可以通过简单的数据交换与主控器进行通信。这种接口简单、灵活，适用于各种微控制器、传感器和其他外部设备。

4、低功耗：AT24C04具有低功耗特性，适合于电池供电的应用。它在非激活状态下消耗极低的电流，有助于延长电池使用寿命。

5、高可靠性：AT24C04具有内部校验和自动错误检测功能，可以确保数据的完整性和可靠性。此外，它还提供了写保护功能，可以防止非授权的数据修改。

总的来说，AT24C04是一种专为小容量数据存储而设计的EEPROM器件。它具有可擦写、I2C接口、低功耗和高可靠性等特点，适用于各种嵌入式系统、电子设备和传感器应用，为存储和管理小规模数据提供了便利和可靠性。

## RTC实时时钟模块

**RTC实时时钟简介**

RTC实时时钟模块是一种常见的电子设备，用于提供准确的日期和时间。它在很多应用中都有广泛的应用，比如计时器、闹钟、数据记录等

**RTC工作原理**

RTC实时时钟模块基于振荡器，该振荡器提供稳定的时钟信号。RTC是一个独立的定时器并使用这个时钟信号来更新RTC计数器，从而保持准确的日期和时间。RTC和时钟配置处于后备区域，系统复位时数据不清零

**RTC配置**

确定时钟源并启用RTC时钟，可选择3中RTC时钟源

HSE时钟除以128 通常为 8MHZ / 138

LSE振荡器时钟 通常为32.768KHZ

LSI振荡器时钟 40KHZ

配置RTC时钟预分频器和计数器精度。

配置日期和时间，并将其写入RTC寄存器

**后备寄存器**

BKP可用以存储用户应用程序数据，当VDD（2 -- 3.6）电源被切断，他们仍然由VBAT（1.8 -- 3.6）维持供电，当系统在待机模式下被唤醒，或系统复位或电源复位，他们不会被复位。向后备寄存器写入数据时需要开启电源和BKP时钟并且使能后备寄存器

**低功耗模式**

低功耗模式包括睡眠模式（sleep），停机模式（stop）和待机模式（standby），可在系统空闲时，降低STM32的功耗，延长设备待机时间

## LCD屏显示模块

**驱动器模块介绍**

ST7789VW 是用于 262K 色图形类型 TFT-LCD 的单芯片控制器/驱动器。 它由 720 条源极线和 320 条栅 极线驱动电路组成。 该芯片能够直接连接到外部微处理器，并接受 8 位/ 9 位/ 16 位/ 18 位并行接口。 显示数据可以存储在 240x320x18 位的片上显示数据 RAM 中。 它可以在没有外部操作时钟的情况下执 行显示数据 RAM 读/写操作，以最大程度地降低功耗。 另外，由于集成电源电路需要驱动液晶； 可以 制造具有最少组件的显示系统。

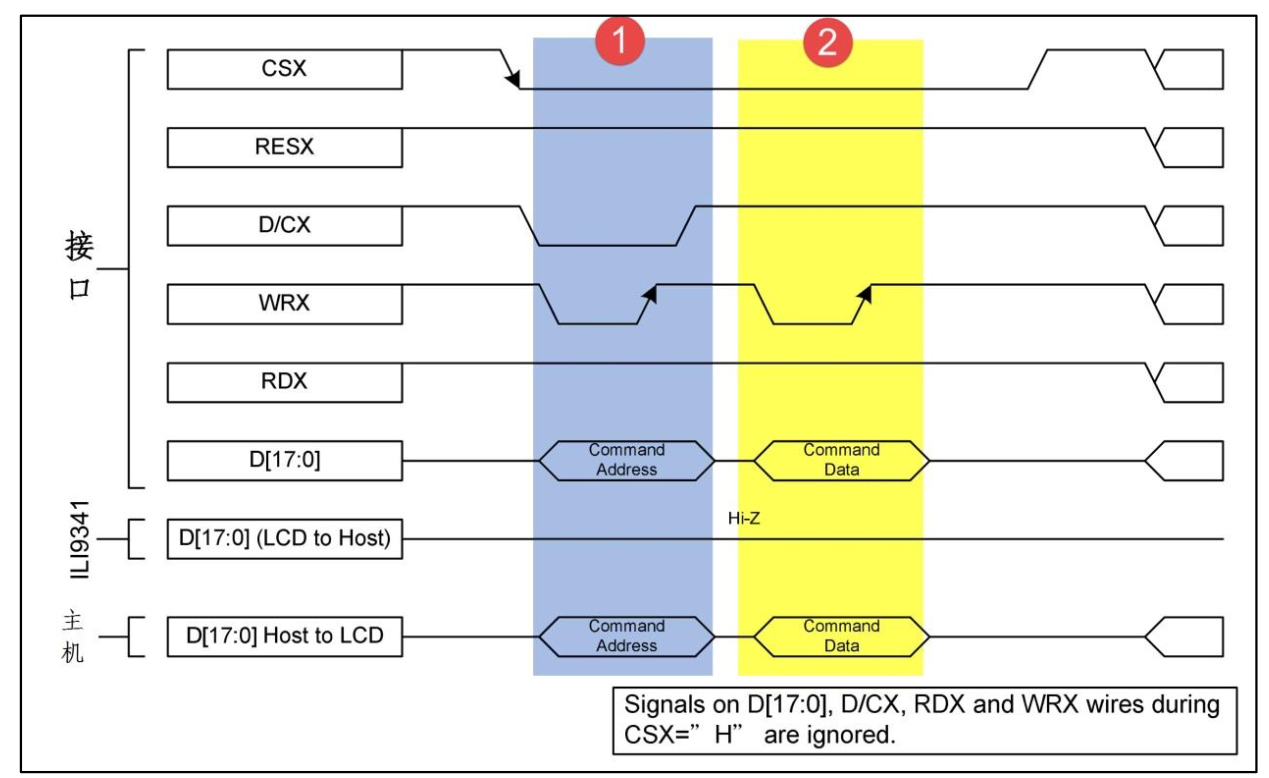
**LCD屏显示原理**

LCD屏中的液晶是一种介于固体和液体之间的特殊物质，它是一种有机化合物，常态下呈液态，但是它的分子排列却和固体晶体一样非常规则，因此取名液晶。 如果给液晶施加电场，会改变它的分子排列， 从而改变光线的传播方向， 配合偏振光片，它就具有控制光线透过率的作用，再配合彩色滤光片，改变加给液晶电压大小，就能改变某一颜色透光量的多少。 利用这种原理，做出可控红、绿、蓝光输出强度的显示结构，把三种显示结构组成一个显示单位，通过控制红绿蓝的强度，可以使该单位混合输出不同的色彩，这样的一个显示单位被称为像素。

**硬件接口时序**

液晶屏的 ILI9341 控制器在出厂前就已经按固定配置好(内部已连接硬件电路)，它被配置为通过 8080 接口通讯，使用 16 根数据线的 RGB565 格式。内部硬件电路连接完， 剩下的其它信号线被引出到 FPC 排线，最后该排线由 PCB 底板引出到排针，排针再与实验板上的 STM32 芯片连接。

|  |  |  |
| --- | --- | --- |
| 信号线 | ILI9341的引脚 | 说明 |
| DB[15:0] | D[15:0] | 数据信号 |
| RD | RDX | 读数据信号，低电平有效 |
| RS | D/CX | 数据/命令信号，高电平时， D[15:0]表示的是数据(RGB像素数据或命令数据)，低电平时 D[15:0]表示控制命令 |
| RST | RESX | 复位信号，低电平有效 |
| WR | WRX | 写数据信号，低电平有效 |
| LCD\_CS | CSX | 片选信号，低电平有效 |
| LCD\_BK | - | 背光信号，低电平点亮 |



## Flash存储

**W25q64模块介绍**

W25Q64是一款64Mb（8MB）容量的串行闪存存储器，由Winbond生产。它采用SPI（Serial Peripheral Interface，串行外设接口）协议进行通信，可以用于存储各种数据，如程序代码、配置信息和用户数据等。W25Q64模块的引脚数多达8个，并且支持低功耗休眠模式，适用于各种嵌入式应用场景。

**空间划分**

W25Q64闪存可被划分为多个扇区（Sector）、块（Block）和页（Page）。一个扇区通常包含若干个块，一个块包含若干个页。每个扇区、块和页都有唯一的地址，可以通过地址来访问存储的数据。

**接口通信**

W25Q64模块使用SPI接口来与微控制器或其他设备进行通信。SPI接口通常由四个信号线组成：片选（CS）、时钟（SCK）、数据输入/输出（MISO/MOSI）和数据/命令选择（D/C）。通过SPI接口，微控制器可以发送命令和数据，读取和写入闪存存储器中的内容。

**读写时序**

W25Q64模块的读写时序遵循SPI协议。对于读操作，首先选择W25Q64模块，然后发送读取指令和地址，接着模块会返回指定地址处的数据。对于写操作，先选择模块，发送写指令和地址，然后发送要写入的数据。

## 字库下载

**汉字编码系统GB2312**

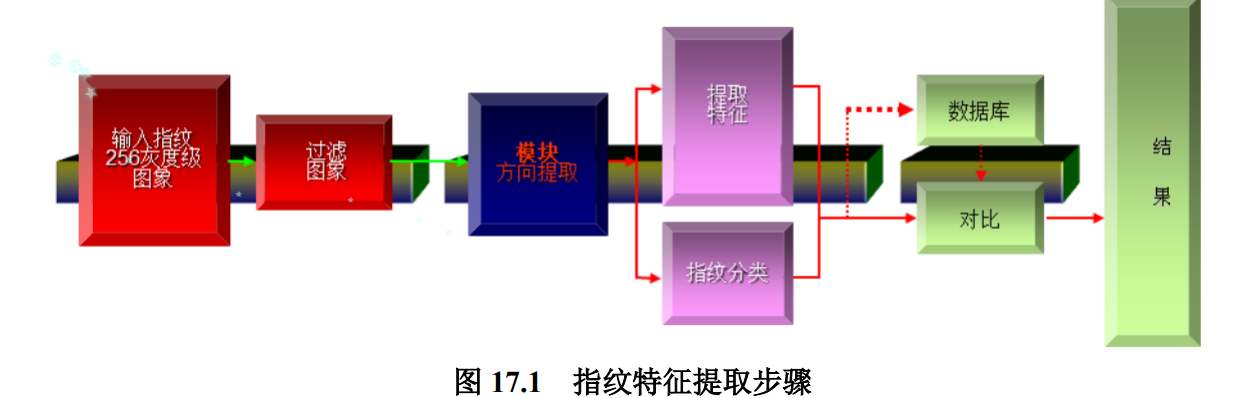
GB2312是一种汉字编码系统，它将每个汉字表示为一个双字节编码。每个汉字的数字表示由两个十六进制数字组成，表示该汉字在GB2312编码表中的位置。高字节范围为0xA1至0xF7，低字节范围为0xA1至0xFE。通过将汉字的高字节和低字节的编码值组合在一起，可以表示出GB2312编码表中的每个汉字。GB2312编码系统只包含了常用的6763个汉字，不包括繁体字等其他字符。对于非GB2312编码的汉字，可以使用更广泛的Unicode编码系统来表示，并将字库整体下载进w25q64LVGL

### BS8116 触摸按键相关简介

“智能锁开发平台”使用的是 BS8116 电容触摸传感器控制芯片来管理数字密码键盘的操作。BS81x 系列芯 片具有 2~16 个触摸按键，可用来检测外部触摸按键上人手的触摸动作。 该系列的芯片具有较高的集成度， 仅 需极少的外部组件便可实现触摸按键的检测。 BS8116 标准触控具有 I 2C 接口，允许与外部设备进行 I 2C 通信。外部设备通过 I 2C 通信可以读取键值、设置 按键感度、设置选项。相比机械按键,电容触摸按键更加可靠,没有震动和移动部件。使用简单方便,可以大大减少按键数量,降低成本。

### 指纹模块

指纹模块分为光学式和电容式，智能锁开发平台项目采用 MG200 电容式指纹采集器。MG200 电容指纹识别 模块使用电容指纹传感器，可完成指纹的采集、比对、储存以及相关的扩展功能。模块包含硬件和软件（核心算 法及管理程序）两部分。 MG200 电容指纹识别模块的管理程序，通过 TTL 电平的 RS232 串行总线接口与主控单元 MCU（或上位机） 连接，采集器接收来自主控单元 MCU（或上位机）的指令，并执行该指令对应的操作，操作完成后再将执行结 果通过 RS232 接口返回给主控单元 MCU（或上位机）；从而实现指纹处理模块的管理平台。管理程序的通信接 口由若干指令组合而成，模块的每个功能由主控单元 MCU（或上位机） 发送独立的指令来执行，执行状态通过 串口反馈给主控单元MCU进行逻辑交互。通过合理的组合使用接口指令，可以适用于指纹识别的各种应用场景， 如何实现功能逻辑则完全由主控单元 MCU（或上位机）决定。



### 10.freeRTOS实时操作系统

（freeRTOS如何进行任务管理，任务的几个模式，任务栈表示什么的空间大小，任务间通信有哪些，任务的阻塞机制有哪些）

**freeRTOS简介**

freeRTOS（Free Real-Time Operating System）是一个开源的实时操作系统，被广泛应用于嵌入式系统的开发中。它的目标是提供一个简洁、高效、可靠的操作系统，适用于各种资源受限的嵌入式设备。作为一个实时操作系统，freeRTOS被设计用于需要实时性能的应用。它采用优先级调度算法，确保高优先级任务能够及时响应，并以可预测的方式执行。freeRTOS支持多任务并发执行和管理，每个任务独立运行且具有自己的优先级和调度规则。freeRTOS还提供了多种任务间通信机制，使不同任务之间可以进行数据交换、同步和互斥访问共享资源。这些机制包括消息队列、信号量、事件标志和互斥量，可以根据应用需求选择适当的通信方式。除了实时性和任务管理，freeRTOS还注重低功耗支持。它可以与低功耗模式和硬件中断相结合，来有效管理系统的能耗。例如，它提供了睡眠模式和节能模式，以便灵活地配置和控制系统的功耗。freeRTOS具有高度的可移植性，支持多种处理器架构和编译器。无论是ARM、AVR、PIC还是MSP430等，都可以在freeRTOS上运行，并且可以方便地移植到不同的硬件平台上。

**freeRTOS任务管理**

freeRTOS使用任务（Tasks）的概念来管理代码的执行。每个任务是一个独立的函数，可以并发地运行在系统中。通过任务管理，您可以创建、启动、停止、删除和切换任务等。

**freeRTOS任务模式**

freeRTOS提供了多种任务模式，包括优先级任务模式、循环任务模式和一次性任务模式。优先级任务模式允许您为每个任务分配一个优先级，并根据优先级决定任务何时执行。循环任务模式允许您定义一个任务的函数，然后以循环方式执行该函数。一次性任务模式是在任务完成后自动删除任务。

**任务栈**

任务栈是每个任务的私有堆栈，用于保存任务执行过程中的临时变量、函数调用和返回地址等。任务栈的大小取决于任务函数的复杂性和任务所需的内存。通常，您可以根据任务的代码复杂性和内存需求来选择适当的任务栈大小。过小的任务栈可能导致堆栈溢出，而过大的任务栈则浪费内存。

**任务间通信**

freeRTOS提供了多种方式来实现任务间的通信，包括消息队列、信号量、事件组和共享资源等。消息队列允许任务通过发送和接收消息来进行通信。信号量允许任务互斥地访问共享资源，以避免竞态条件。事件组允许任务等待和触发特定的事件。共享资源允许多个任务同时访问相同的资源。

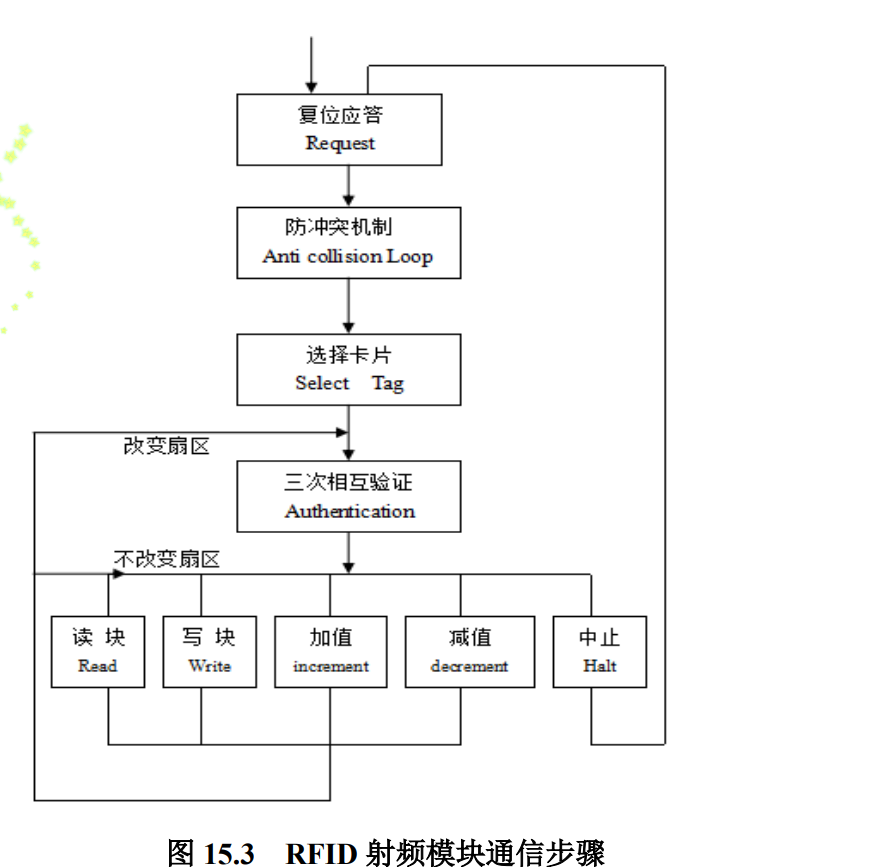
**任务的阻塞机制**

freeRTOS提供了多种任务阻塞机制，用于控制任务的执行和等待。例如，延时阻塞允许任务暂停一段时间后再继续执行。指定事件阻塞允许任务等待特定的事件发生再继续执行。互斥阻塞允许任务在访问共享资源时阻塞其他任务的访问。

**10.rfid模块**

RFID（Radio Frequency IDentification）称为射频设备计数或无线射频识别，属于通信方式的一种。可通过无 线电讯号识别特定目标并读写相关数据，而无需识别系统与特定目标之间建立机械或光学接触。常用的有低频 （125k～134.2K）、高频（13.56Mhz）、超高频，微波等技术。 一套完整的 RFID 系统， 是由阅读器（Reader）与电子标签（TAG）也就是所谓的应答器（Transponder）及 应用软件系统三个部份所组成，其工作原理是 Reader 发射一特定频率的无线电波能量给 Transponder，用以驱动 应答器（Transponder）电路将内部的数据送出，此时阅读器（Reader）便依序接收解读数据，送给应用程序做相 应的处理。

RC522 模块提供了两种通信接口，分别是 I 2C 串行通信、SPI 串行通信。数据传输顺序为先传高位，再传低 位。在 I 2C 串行通信模式下，支持快速模式（400Kbit/s）和高速模式（3400Kbit/s）。在 SPI 串行通信模式下，最 大的传输速度为 10Mbit/s，数据与时钟相位关系满足“空闲态时钟为低电平，在时钟上升沿同步接收和发送数据， 在下降沿数据转换”的约束关系。



# 三．项目结项

## 1.该项目能让自己有什么收获

首先，在这个项目中，涉及到了多种传感器，例如语音模块、指纹模块、rfid传感器等多种技术的应用和开发。通过与不同硬件模块的接入和配置，我能够更加熟练地理解和运用各种传感器以及互联网设备的开发技术。并且对I2C和SPI通信有了更深层次的理解，其次，通过这个项目，我进一步加深对单片机操作系统FreeRTOS的理解和运用，在这个项目中，我们需要进行任务优先级调度、堆栈资源的合理分配、进度控制等各种任务管理工作。通过这个项目，我对任务管理有了新的理解。并且对智能锁的实现有了初步的认识，也引起了我对智能家居的兴趣，总体来说,这个项目对提高我的嵌入式系统设计与开发能力很有帮助。

**2.项目中的问题及解决方法**

在这个智能锁项目中，其实遇到了挺多的问题，但很多自己不懂的东西和同组的兄弟们一讨论大多就迎刃而解了，当然其中也少不了老师的帮助，说一下让我影响最深的几个问题，第一个既是ucos的配置问题，ucos的移植很简单，但之后运行任务的时候，总是跳进硬件错误里面，我在这里卡了一天半，不知到时什么原因，网上查资料但就是不对，最后一步一步的往新工程里加功能才发现时串口一的配置问题，使用ucos的前提下，如果串口一使用库函数配置就很容易进硬件错误，到现在都感觉莫名奇妙，最后用寄存器问题就解决了，还有就是bs8116里面的问题，iic通信就不多说了，命名和同学时一样的时序，一样的功能，但我的跑起来就是不正确，至今无解，当然也发现了自己的短板，spi是真的不熟，还需要课下自己在努努力，最后了说一下让我影响最深刻的bug，指纹模块的指纹识别，里面有几个毫秒延时，起初不是特别在意，随便给了一下，导致最后的指纹识别一直不对，看了半天才发现延时的时间给短了，真的是马虎不得，对了，还有那个ucos运行的任务功能总是不正常，也不知道为啥，无奈，最后选择了freertos