

学 号:	0122204950903
------	---------------

# 武汉理工大学

## 课 程 设 计

题 目	测控系统微处理器应用课程设计
学 院	机电工程学院
专 业	测控技术与仪器
班 级	测控 2203
姓 名	林兆先
指导教师	牟新刚

2025 年 1 月 10 日

## 本科生课程设计成绩评定表

姓 名	林兆先	专业、班级	测控 2203
课程设计题目：测控系统微处理器应用课程设计			
<p>课程设计答辩或质疑记录（至少 3 个问题）：</p> <p>问题 1：AD0832 模数转换器是如何将输入的模拟信号转换为数字信号的？</p> <p>答：AD0832 的数模转换过程的工作过程可以具体分为四个阶段：采样、保持、量化和编码。这些阶段共同构成了 ADC 电路的基本架构，确保了模拟信号到数字信号的高效转换。</p> <p>首先，采样阶段将连续时间信号转换为离散时间信号。AD0832 的采样电路周期性地捕捉输入的模拟信号电压值。在每个采样时刻，模拟开关闭合，瞬间“采”取一个信号样本，从而将时间连续、数值连续的模拟信号变为时间离散、数值连续的采样信号。</p> <p>接下来，在保持阶段通过保持电路将采样后的信号锁定在一个稳定的电压水平。AD0832 内部集成的保持电路在采样后持续稳定输入信号，同时确保在整个转换过程中信号保持不变，从而提高转换的准确性。</p> <p>然后会进入量化阶段，AD0832 利用逐次逼近寄存器（SAR）将连续数值信号转换为离散数值信号。内部的逐次逼近寄存器通过逐位确定数字输出，从最高位开始，逐步逼近输入信号的实际值。每一位的确定都基于当前的逼近结果与输入信号的比较，逐步缩小数值差距，最终实现精确的数值表示。</p> <p>最后，编码阶段将量化后的离散数值信号转换为二进制代码。在 AD0832 中，量化电路的输出通过编码电路被转化为二进制格式的数字信号。</p> <p>问题 2：如何进一步实现步进电机在关门过程中状态的检测与监控？</p> <p>答：在此次课设中使用到的步进电机并没有在实物部分增加对于步进电机开关门的检测，而是使用了软件逻辑对其进行开关门判断，但会造成单片机断电状态丢失的情况。因此，对于这方面的改进，可以在系统中增加对于开关门检测的传感器实现。</p> <p>一种方案是使用光电传感器，将光电传感器的一端作为光源发射器，另一端作为光信号接收器，分别固定在门框和门的对应位置上。当门关闭时，光源与接收器之间的光路被完整遮断，此时可以单片机检测到对应的 I/O 口信号发生跳变，从而判断门的状态。</p> <p>另一种方案是使用磁簧开关，将磁簧开关固定在门框的一侧，磁铁则安装在门的对应位置。当门关闭时，磁铁靠近磁簧开关，产生磁场使簧片闭合，从而完成电路连接。此时，单片机可以检测到对应的 I/O 口信号发生跳变，判断门的状态为“关闭”。当门打开时，磁铁远离磁簧开关，簧片断开，电路中断，单片机同样通过检测 I/O 口信号的变化来判断门的状态为“打开”。这种安装方式简便且响应迅速，能够有效实现门的开关状态检测。</p> <p>问题 3：步进电机的正反转和驱动的控制原理是如何实现的，如何实现对门的不同开关角度进行控制？</p> <p>答：</p> <p>步进电机的工作基于电磁感应原理。当定子绕组通以脉冲电流时，定子产生的磁场与转子磁场相互作用，产生转矩，使转子按一定步距角旋转。每接收一个脉冲，</p>			

转子就旋转一个固定的步距角，从而实现精确的角度控制。

(1) 步进电机的控制原理：

本次课程设计采用 42 步进电机，需要将其三个控制端口分别连接至单片机的任一 I/O 口。其中，三个 I/O 口的作用分别为 En（使能端）、Stp（步进脉冲端）、Dir（方向端），这三个端口的作用分别为：

**En：**该端口用于启用或禁用步进电机。当 En 端口被激活时，电机开始工作；当该端口处于禁用状态时，步进电机停止工作。

**Stp：**该端口用于生成步进脉冲信号。每接收到一个脉冲，步进电机转动一个固定的步距角。通过控制脉冲的频率，单片机可以调节电机的旋转速度。

**Dir：**该端口用于设置步进电机的旋转方向。通过控制该端口的高低电平，单片机可以选择步进电机是顺时针还是逆时针旋转。

本次使用的 42 步进电机通过 STM32 驱动，使用脉冲段模拟高低电平进行脉冲的发送进行控制。即在软件段，通过控制单片机的一个 GPIO 口，使用 for 循环不断地发送指定数量的高低电平脉冲，即可实现步进电机的驱动。

而对于步进电机的正反转控制，只需要对步进电机 Dir 端所连接单片机的 I/O 口进行高或低电平的控制即可实现对于方向的改变。

(2) 对于不同开关角度的控制：

每发送一个脉冲，步进电机便会旋转  $1.8^{\circ}$ 。在本次课程设计中，门的开关控制要求步进电机进行  $\pm 90^{\circ}$  的旋转，因此需要在软件端发送 800 个脉冲，以实现  $\pm 90^{\circ}$  的旋转。为了实现更灵活的角度控制，并根据门的实际角度状态进行调整，可以通过将陀螺仪与门的运动结合起来，实时检测门所处的角度。通过陀螺仪的角度数据，系统可以精确判断门的位置，并根据实际需要调节电机的旋转角度，从而确保门的开关角度得到准确控制。

## 成绩评定依据

序号	评 价 内 容	分值	得分
1	进行文献资料检索与综述，并运用相关理论知识，分析并提出满足特定需求的系统总体方案。	20	
2	进行系统设计时，能够结合相关专业理论知识，选择与使用恰当的工程工具、专业软件对关键部件和系统整体进行设计与模拟，并分析其局限性，且结论正确。	20	
3	能运用现代工具完成电路图的绘制，并完成电路板的焊接，以模拟装置的方式呈现设计成果。	20	
4	编制相应的程序并完成调试，结果正确。	20	
5	能准确、清晰地表达、交流设计内容（设计过程、答辩）	10	
6	能按规范撰写设计说明书、图表。	10	
合 计		100	

## 其他事项说明

指导教师签字：\_\_\_\_\_

年      月      日



# 课程设计任务书

学生姓名: 林兆先      专业班级: 测控 2203

指导教师： 牟新刚      工作单位： 机电工程学院测控系

题 目： 测控系统微处理器应用课程设计

初始条件:

- 1、相关电路的原理图、PCB 板和元器件
- 2、PC 机、keil c51 编程软件

### 要求完成的主要任务:

1. 基本系统：设计一个单片机系统，完成原理图设计、PCB 设计、PCB 加工、电子元器件焊接、调试、程序下载，并实现数码管显示、矩阵键盘扫描、中断程序、定时器程序、串口通讯等基本功能；
2. 显示功能：焊接电路并实现对 1602 液晶屏的显示功能，要求能滚动显示字符；
3. 数据采集：焊接电路并实现 ADC0832 的数据采集功能；
4. 从选做项目中任选一项，完成相关软硬件设计；
5. 参考武汉理工大学课设设计规范，完成设计说明书的撰写；
6. 课程设计要求参考文献不少于 5 篇，其中不能全部是书，要有 2-3 篇论文。应该在说明书中引用。

### 时间安排:

序	设计内容	时间 (天)
1	导师提前布置任务、学生选题	1
2	查阅资料，分析关键问题，确定总体方案	2
3	硬件电路设计及选型	4
4	软件设计及仿真联调	3
5	系统焊接与调试	3
6	个人总结、成果评价与交流	2
合 计		15 天

指导教师签名： 年 月 日  
系主任（或责任教师）签名： 年 月 日

# 目录

第一章 绪论.....	7
1.1 设计任务.....	7
1.2 设计内容及要求.....	7
第二章 51 单片机系统总体设计方案.....	9
2.1 51 单片机设计方案及对比分析.....	9
2.2 系统各模块原理图设计.....	13
2.3 系统各模块代码设计.....	19
第三章 51 单片机系统效果演示.....	24
3.1 中断程序演示.....	24
3.2 定时器程序演示.....	26
3.3 LCD 程序演示.....	26
3.4 ADC 采样程序演示.....	27
3.5 串口通信程序演示.....	28
第四章 智能门禁系统总体设计方案.....	30
4.1 各模块设计方案及对比分析.....	30
4.2 各模块原理图绘制.....	34
4.3 门禁菜单交互设计.....	42
4.4 门禁系统开关方式设计.....	43
4.5 门禁系统开关装置设计.....	50
4.6 门禁系统拓展功能设计.....	51
第五章 智能门禁系统效果演示.....	54
5.1 交互功能演示.....	54
5.2 门禁驱动实物.....	65
5.3 电源模块实物.....	67
第六章 课程设计总结.....	68
参考文献.....	70

# 第一章 绪论

## 1.1 设计任务

随着现代工业与科技的飞速发展，单片机作为嵌入式系统的核心技术，已在多个领域中得到了广泛应用。作为一种高度集成的微控制器，单片机不仅具备强大的数据处理能力，还能执行复杂的控制任务、进行数据采集、信息处理及系统调度等多项功能。因此，它在自动化控制、智能化设备、家电系统、仪器仪表等领域中，发挥着越来越重要的作用。单片机系统的广泛应用，推动了现代电子技术的迅速发展，也为各种嵌入式系统的设计与创新提供了无限可能。

## 1.2 设计内容及要求

本次课程设计的任务分为两个主要部分：其一，基于单片机系统的设计与实现，涵盖原理图设计、PCB 设计与加工、电子元器件焊接与调试、程序下载与调试等多个环节。通过这一部分的工作，旨在实现数码管显示、矩阵键盘扫描、定时器控制、串口通讯等基本功能，并最终完成 1602 液晶屏的字符滚动显示与 AD0832 数据采集模块的设计与实现。其二，围绕自主单片机选型与课题选择，设计一个完整的单片机系统，通过合理的硬件选择和软件开发，实现系统功能的全面验证与优化。

通过本项目的开展，我们不仅将深入理解单片机系统的工作原理，还能全面掌握从硬件设计到软件实现的全过程，提升在嵌入式系统开发中的实践能力。此外，项目的完成将为后续更加复杂的嵌入式系统设计与应用奠定坚实的基础。

本次课程设计的任务书分为两个部分，旨在通过实践深入理解单片机系统的设计与实现。第一部分采用 51 单片机为主控，完成对数码管显示、矩阵键盘扫描、定时器控制、串口通讯等基本功能的实现。该部分工作将包括自主设计原理图、PCB 绘制、焊接及调试等环节，以确保硬件部分能够顺利运行，并完成所需功能。数码管显示模块将用于信息展示，矩阵键盘用于输入，定时器控制实现周期性任务的调度，串口通讯模块则实现与外部设备的数据交换。通过这些功能模块的集成，形成一个基础的单片机系统。

第二部分则采用 STM32 系列单片机，结合按键、蓝牙、NFC 等模块，设计并实现一个智能门禁系统。该系统通过按键输入、蓝牙通信以及 NFC 识别技术，提供便捷的

身份验证与门禁控制。**STM32** 单片机在性能、资源和外设支持上相较于 **51** 单片机具有更高的处理能力，能够满足智能门禁系统对于高速数据处理和多种通信协议的需求。通过该部分的设计与实现，进一步拓宽了单片机在智能硬件领域的应用。

通过本次课程设计，既能深化对单片机硬件设计、软件开发与系统调试的理解，又能够提升项目实际开发能力，为未来复杂嵌入式系统的设计与实现提供宝贵经验。

## 第二章 51 单片机系统总体设计方案

根据任务书要求，本次单片机课程设计要求设计一个单片机系统，其中要求完成原理图设计、PCB 设计、PCB 加工、电子元器件焊接、调试、程序下载，并实现数码管显示、矩阵键盘扫描、中断程序、定时器程序、串口通讯等基本功能。总体设计分为原理图和 PCB 设计、PCB 焊接、程序烧录验证这三个过程。

### 2.1 51 单片机设计方案及对比分析

#### 2.1.1 MCU 部分选型

根据任务书要求，对于必做部分的 MCU 选择了 STC89C52 芯片。STC89C52 是一款基于 8051 架构的单片机，该芯片内部集成了丰富的外设资源，包括 GPIO、定时器、串口通信、PWM 输出等，同时也符合任务书需求。

此外，STC89C52 的编程简便，支持丰富的开发工具和编程环境，便于快速开发和调试。这款芯片的稳定性和可靠性经过广泛验证，已经在多种工业和民用产品中得到应用。结合任务书中对系统稳定性、性价比和性能的要求，STC89C52 无疑是一个理想的选择<sup>[1]</sup>。



图 1 STC89C52 单片机

#### 2.1.2 中断程序设计

根据任务书要求，本次设计采用了按键中断的方式来实现中断程序。具体要求是实现按键触发中断的功能，目的是通过按键触发中断后，使得数码管显示的数字加 1。具体实现过程中，当按下 INT2 按键时，单片机触发外部中断信号，程序进入中断处理程序，在该程序中实现对显示内容的更新，即使数码管显示的数字加 1。

首先，需要进行对于外部中断的 IO 口配置，根据表 1 和表 2 的内容，外部中断 0 的配置需要通过设置相关寄存器来实现。首先，在中断允许控制寄存器（IE）中，需要将 EA（全局中断使能位）设置为 1，以使能全局中断功能。此外，EX0（外部中断 0 使能位）也需要设置为 1，允许外部中断 0 触发中断请求。

接下来，在 TCON 中，需要配置 IT0 来指定中断 0 的触发方式。若 IT0 设置为 1，则外部中断 0 为边缘触发；若设置为 0，则为电平触发。为了确保外部中断能够正常响应，还需要确保 IE0 在中断触发时被正确清除。

通过上述配置，外部中断 0 即可在按键按下时触发中断，进而实现数码管显示数字加 1 的功能。

表 1 中断允许控制寄存器 IE

7	6	5	4	3	2	1	0
EA			ES	ET1	EX1	ET0	EX0

表 2 中断请求标志位 TCON

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

### 2.1.3 定时器程序设计

本次定时器程序设计结合数码管显示，实现时钟效果。采用两个四位数码管，利用定时器时钟和定时器中断来实现时钟功能。具体来说，八个显示位的安排如下：前两位显示日期，接下来的两位显示小时，随后两位显示分钟，最后两位显示秒钟。例如，8 号早上 8 点 21 分 56 秒会显示为“08082156”。

对于定时器的设置，根据表 3，定时器 0 采用 16 位定时器模式。为了设置定时器的初值，程序通过计算定时器的初值来保证每 50 毫秒中断一次。定时器的计数周期由定时器的初值 TH0 和 TL0 决定，其中对于 TH0 和 TL0 有计算：

$$\text{计时器} = 50\text{ms} \times \frac{f_{osc}}{12} = 50000$$

式中， $f_{osc}$  为系统时钟频率，通常为 12 MHz。定时器的计时周期为 12 MHz 时钟频率的  $\frac{1}{12}$ ，因此每个定时器时钟周期的时间为  $\frac{1}{12} \times \frac{1}{f_{osc}}$ 。根据这一关系，计算得出定时器初始计时值为 5000。

然后，使用 16 位定时器的最大计数值 65536 减去 50000，得到定时器的初始值为 15536。将其转换成 16 进制：

$$15536 = 0x3CB0$$

因此，定时器的初值设置为：TH0 = 0x3C, TL0 = 0xB0

表 3 定时器工作方式表

M1	M2	位数
0	0	两个 13 位定时器/计数器
0	1	两个 16 位定时器/计数器
1	0	两个 8 位定时器/计数器
1	1	一个 8 位定时器/计数器

程序设计部分通过在定时器中断部分对时钟进行逻辑处理，再将处理后的字符以每秒钟的变化显示在数码管上，以此实现时钟的效果。

#### 2.1.4 LCD 程序设计

根据任务书的要求，系统需实现 LCD 的正常显示以及 LCD 滚动显示字符的功能。在本项目中，LCD 在第一行进行静态显示日期，在第二行进行字符滚动显示。

LCD 滚动显示字符的基本原理是通过不断地更新显示的内容，使得屏幕上的字符位置发生变化，从而给用户呈现一种字符滚动的效果。具体来说，根据其原理，LCD 滚动显示算法为：

（1）初始化：将要滚动的字符串存储到一个缓冲区，并将 LCD 初始化完成。假设缓冲区长度为 N，LCD 显示行数为 2，行宽为 16 字符。

（2）逐字符移动：

将第一个字符显示到 LCD 屏的第一列。

每次更新时，将缓冲区中的字符向左移动一位，新的字符从缓冲区中取出并显示在 LCD 屏的右侧。

当所有字符都被显示完毕时，再次从缓冲区的第一个字符开始，形成循环。

（3）控制显示更新的时间间隔：

可以利用定时器中断或主循环中的延时函数来控制每次字符更新的时间间隔，通常会设定为几十毫秒，以便产生平滑的滚动效果。

（4）显示更新的顺序：

通过对显示内容的控制，逐行或逐字符更新 LCD 显示的内容。在每一帧更新时，确保 LCD 的显示区被正确刷新。

#### 2.1.5 串口通信程序设计

据任务书的要求，您需要实现 51 单片机的串口通信功能，其中单片机在接收到通过串口发送过来的字符串后，会将接收到的字符串原封不动地返回给串口端。实现“回显”功能。

为了实现双机通信，根据表 3 和表 4，口需要设置为工作方式 1。在工作方式 1 下，串口的数据传输包括 8 位数据位、1 位停止位，且不使用校验位。波特率通过定时器进行设置，目标波特率为 9600。具体来说，波特率的设置是通过定时器初值的计算来实现的。

其中通过定时器设置波特率为 9600 的计算过程为：

$$9600 = \frac{f_{osc}}{32 \times (65536 - \text{定时器初值})}$$

计算得到，TH0 = 0xFD, TL0 = 0xFD。

表 4 串口寄存器 SCON

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

表 5 串口工作方式

SM0	SM1	工作方式	功能说明	波特率
0	0	0	同步移位寄存器	$F_{osc}/12$
0	1	1	10 位异步接收/发送	可变，与定时器 1 溢出率有关
1	0	2	11 位异步接收/发送	$F_{osc}/32$ 或 $F_{osc}/64$
1	1	3	11 位异步接收/发送	可变，与定时器 1 溢出率有关

## 2.1.6 ADC 转换程序设计

根据任务书要求，需完成焊接电路并实现对 AD0832 的数据采集功能。根据前文描述，本次任务实现了通过 AD0832 对光敏电阻的数据采集。光敏电阻的工作原理是：光照越强，其电阻值越小；光照越弱，电阻值越大。因此，随着光照强度的变化，ADC 所采集到的电压也会发生变化：光照强时，ADC 采集的电压较小；光照弱时，ADC 采集的电压较大。本次任务实现了对光敏电阻电压的采集，并将采集的数据在两个四位数码管上显示。

### 2.1.6.1 AD0832 工作原理

AD0832 的 A/D 转换过程开始时，CS 输入端需要被拉低，表示芯片进入工作模式。此时，CLK 时钟信号开始驱动芯片进行转换，同时 DI 端用于输入控制信号。第一个时钟脉冲的下降沿前，DI 端需要置为高电平，作为转换的起始信号。接下来的两个时钟脉冲下降沿前，DI 端输入两位数据，用于选择 A/D 转换的模式和通道。通过这些信号，芯片确定是选择单端输入还是差分输入，并且选择具体的输入通道<sup>[2]</sup>。

从第四个时钟脉冲下降沿开始，芯片会输出转换结果，数据从 DO 端按位输出，MSB 先输出，LSB 后输出。转换结果为 8 位数据，代表模拟信号的数字化值。整个过程中，CS 需要保持低电平，直到转换完成后才能拉高，结束本次转换。通过这种时



序控制，AD0832 完成了从模拟信号到数字信号的转换，并提供了相应的数字输出。

ADC0832 MUX ADDRESS CONTROL LOGIC TABLE

MUX ADDRESS		CHANNEL NUMBER	
SGL/ $\overline{\text{DIF}}$	ODD/ $\overline{\text{EVEN}}$	0	1
L	L	+	-
L	H	-	+
H	L	+	+
H	H		

H = high level, L = low level,  
- or + = polarity of selected input pin

图 2 ADC 通道地址选定图

## 2.2 系统各模块原理图设计

本次 51 单片机系统的原理图设计采用了嘉立创软件。嘉立创软件具有丰富的功能模块库和高效的设计工具，能够帮助设计者快速完成电路图的绘制，并提供了方便的元器件选型与自动布局功能。通过使用该软件，能够确保原理图设计的准确性与规范性，同时提高设计效率。

### 2.2.1 MCU 原理图绘制

对于其原理图绘制，除了少数不需要用到的引脚外，对该芯片的所有引脚进行了引出，原理图如下：

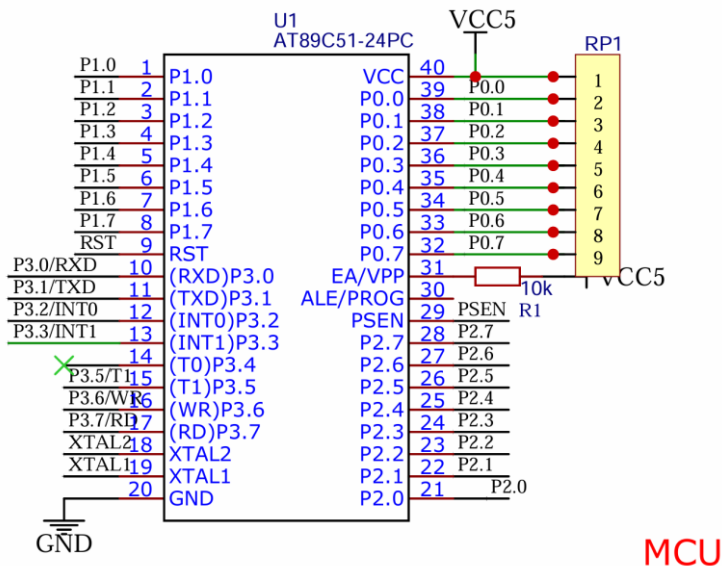


图 3 STC89C52 最小系统原理图

### 2.2.2 LCD 原理图绘制

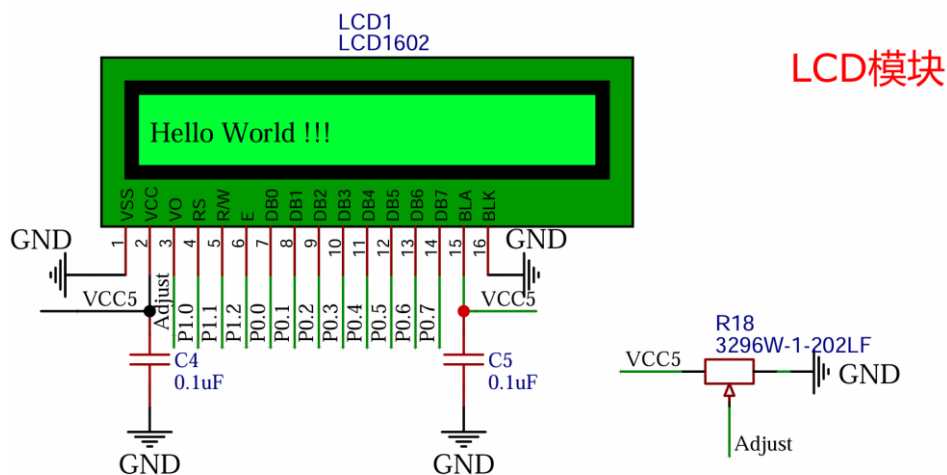


图 4 LCD 原理图

### 2.2.3 矩阵键盘原理图绘制

矩阵键盘部分采用了 3x3 矩阵键盘，并将其行列引脚接入单片机的 P1 端口。除此之外，系统还引出了四个独立按键，其中两个按键连接到 P1 端口的其他引脚，另外两个按键则接入外部中断口，用于实现特定的功能操作。

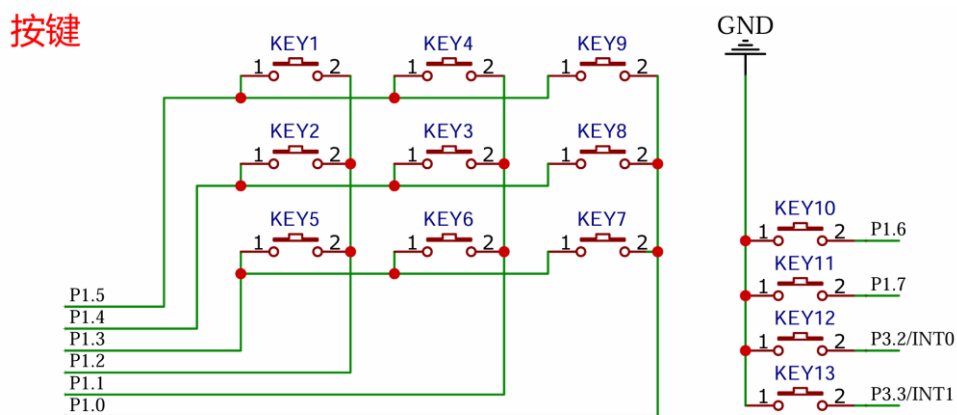


图 5 矩阵键盘原理图

### 2.2.4 数码管原理图绘制

数码管显示部分采用了两位四位数码管，并通过 74HC573 芯片实现数码管的驱动。该芯片的控制信号通过 P2 端口与单片机连接，以便完成显示内容的控制与切换。

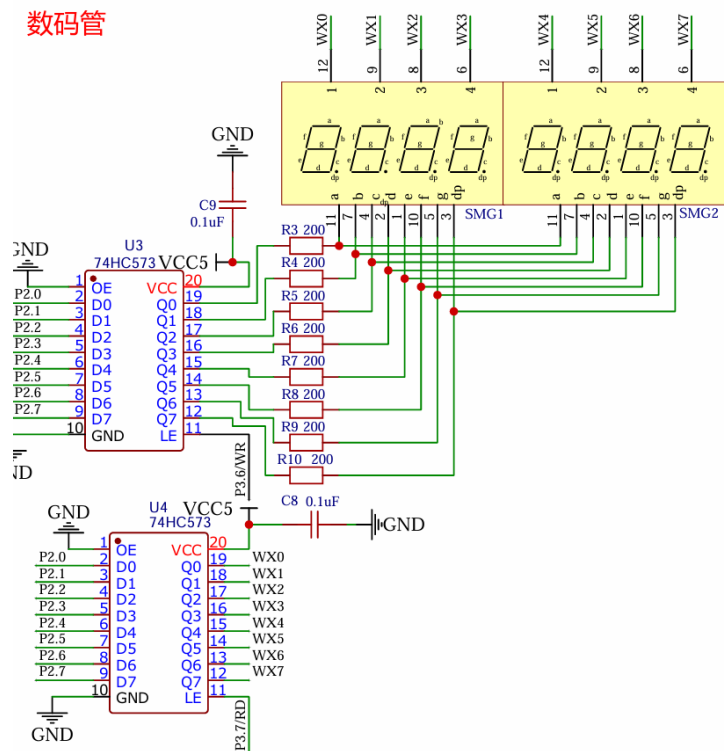


图 6 数码管原理图

### 2.2.5 ADC 转换原理图设计

根据任务书要求，要求使用 AD0832 实现 ADC 转换，根据前文分析，将 ADC0832 的两个采样通道分别接入电位器和光敏电阻，实现对电位器和光敏电阻电压的采集。

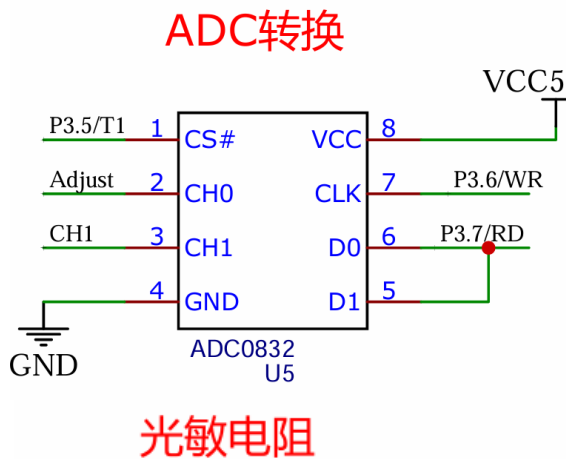


图 7 ADC0832 部分原理图

### 2.2.6 其它模块原理图绘制

### 2.2.6.1 LED 原理图

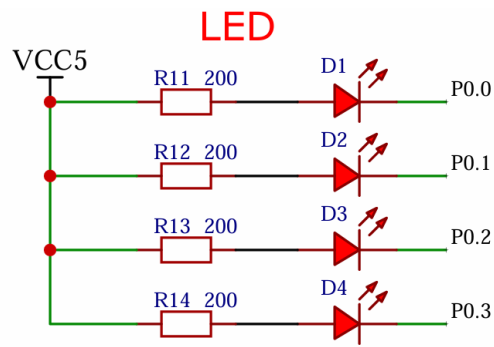


图 8 LED 阵列

### 2.2.6.2 晶振原理图

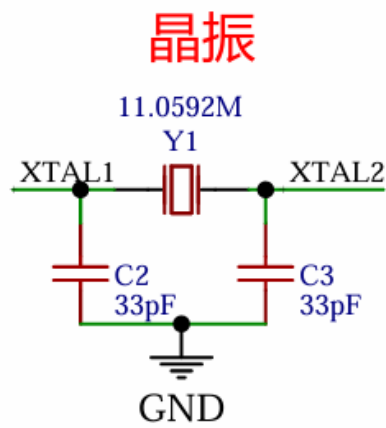


图 9 晶振原理图

### 2.2.6.3 复位按键原理图

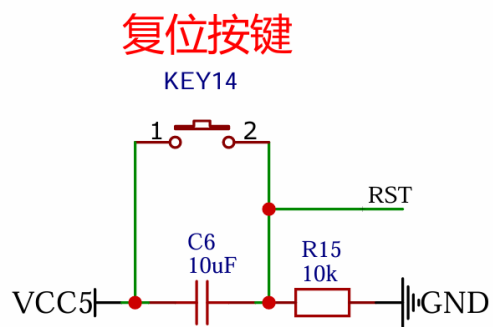


图 10 复位按键原理图

### 2.2.6.4 自锁开关原理图

## 自锁开关

U6

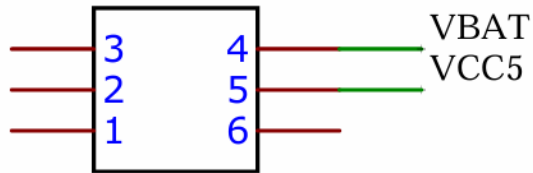


图 11 自锁开关原理图

### 2.2.6.5 排针原理图

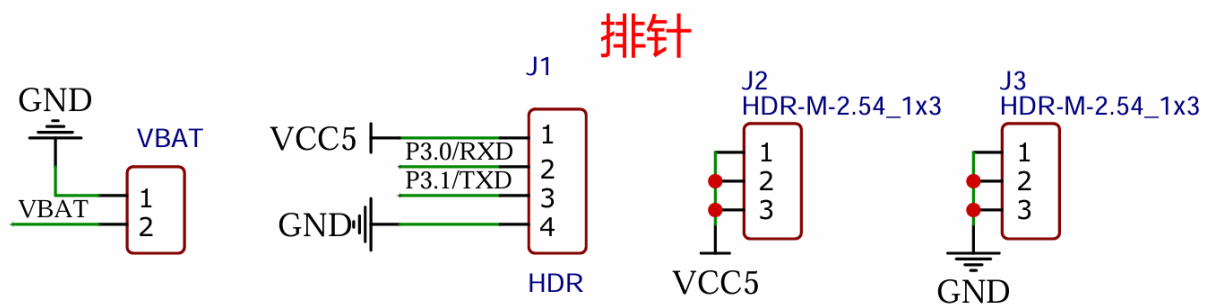


图 12 排针原理图

### 2.2.7 原理图和 PCB

下图为 51 单片机系统的原理图设计：

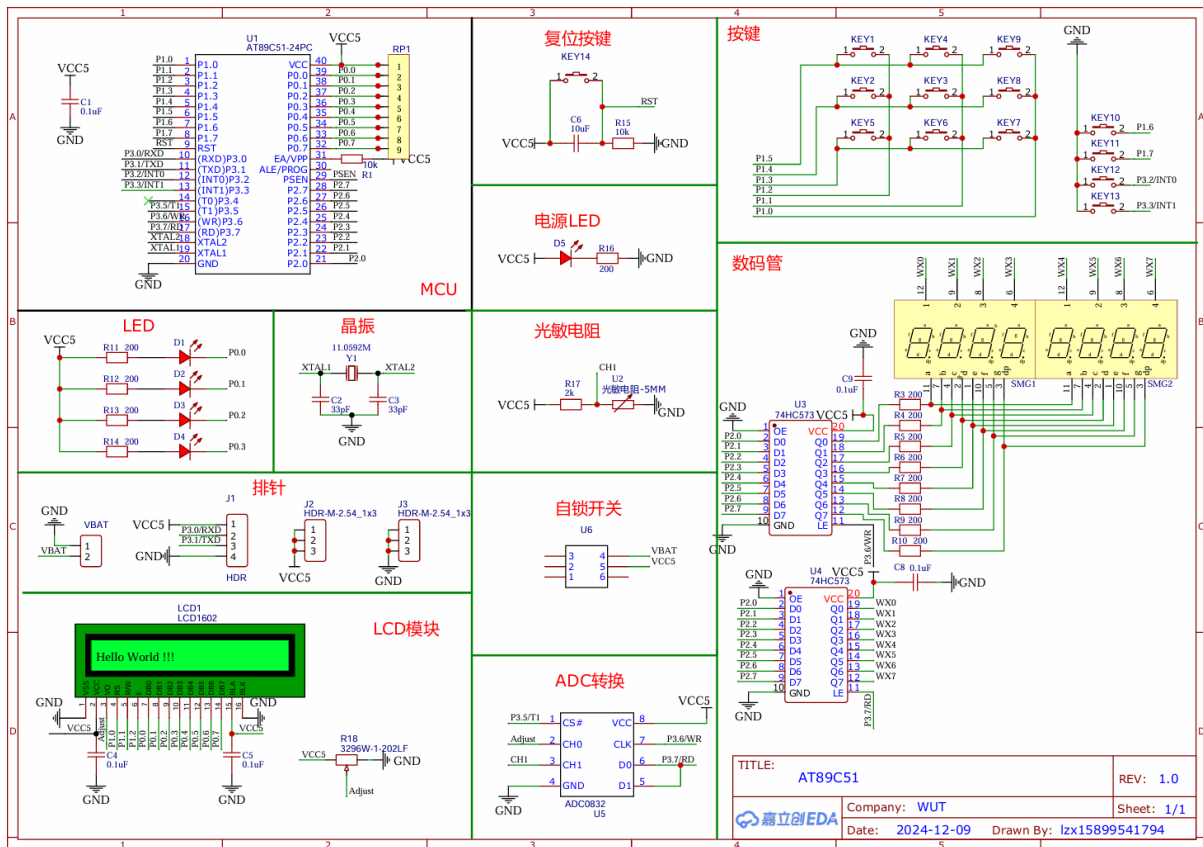


图 13 整体原理图

下图为 51 单片机的 PCB 和实物部分：

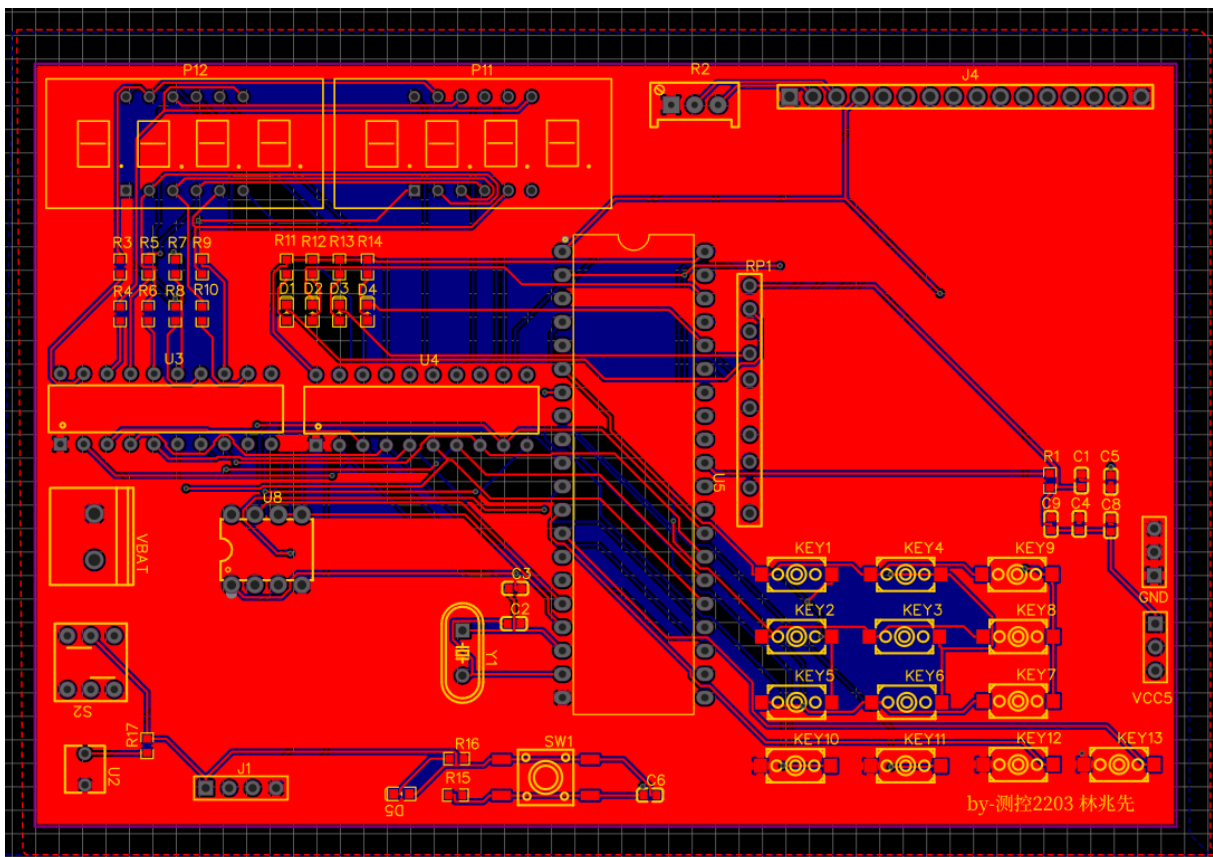


图 14 51 单片机 PCB 图

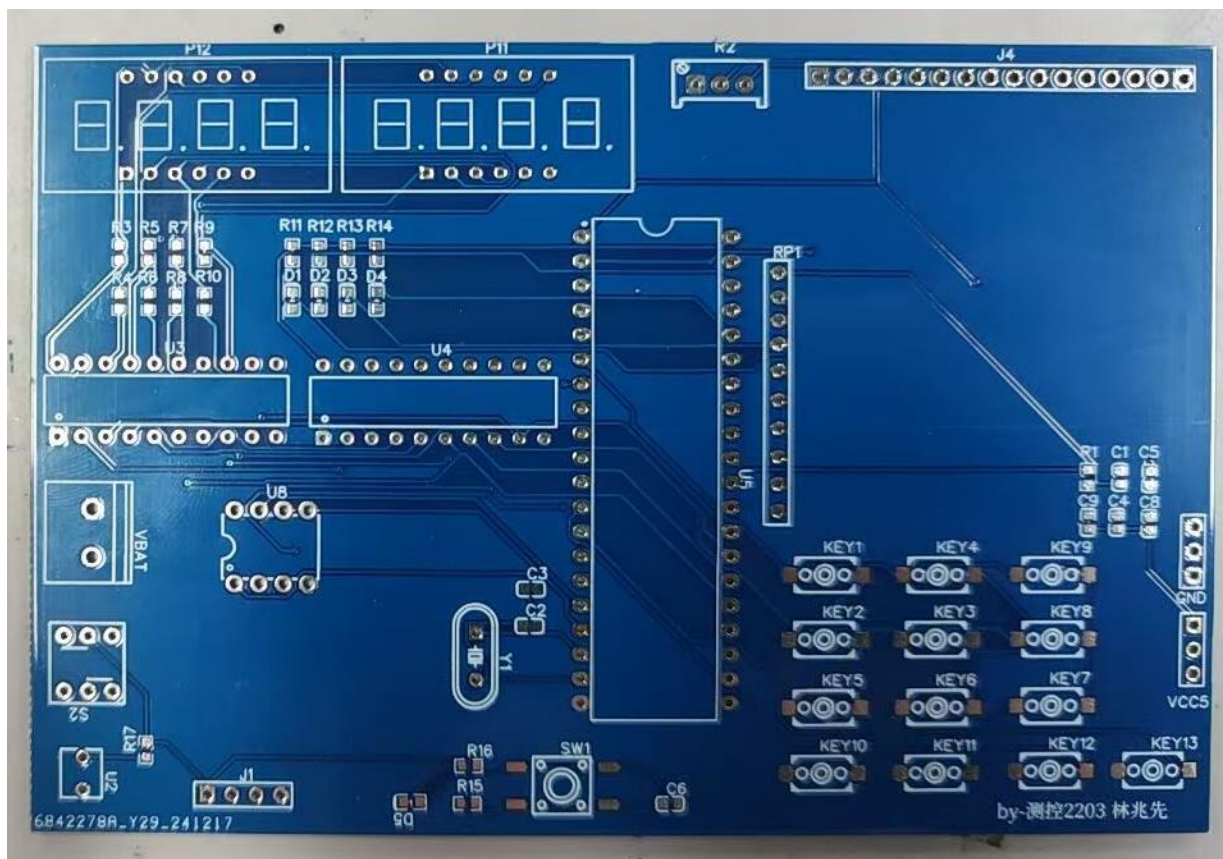


图 15 PCB 实物图

## 2.3 系统各模块代码设计

### 2.3.1 中断程序代码

如下图所示为中断程序的主函数代码部分。

程序首先启用了全局中断( $EA = 1$ )，设置外部中断 0 为边缘触发( $IT0 = 1$ )并使能外部中断 0( $EX0 = 1$ )，以响应按键输入。当外部中断触发时，进入中断服务程序。

在 `while(1)` 循环中，程序通过 `for` 循环控制 8 个数码管的显示，使用 `_crol_` 宏左移 P2 口，逐一选通各个数码管。每次显示时，P3.7 产生锁存信号，确保数据稳定显示，然后通过 `ShowNum[button_cnt]` 将数字编码输出到 P2 口，P3.6 生成显示信号。为了避免闪烁，使用 `delay_ms(1)` 实现短延时，确保视觉暂留效果。



```

void main()
{
    unsigned int j;
    P2 = 0x01; // 初始化P2口

    EA = 1;
    IT0 = 1;
    EX0 = 1;

    while(1)
    {
        for(j = 0; j < 8; j++)
        {
            P2 = 0x01; // 重新初始化P2, 开始选通第一个数码管
            P2 = _crol_(P2, j); // 循环左移, 选通相应的数码管
            P3_7 = 1; // 产生锁存信号, 高电平
            P3_7 = 0; // 锁存信号恢复为低电平

            P2 = ShowNum[button_cnt]; // 显示编码, 输出到P2口
            P3_6 = 1; // 产生显示信号, 高电平
            P3_6 = 0; // 显示信号恢复为低电平
            delay_ms(1); // 短延时实现视觉暂留效果
        }
    }
}

```

图 16 中断程序主函数

如下所示为外部中断 0 的中断服务程序代码。当按键被按下时，触发外部中断 0 并进入此中断服务程序。

```

void int0() interrupt 0 {
    if (!key) {
        button_cnt += 1;
        button_cnt = button_cnt & 0x0F;
        delay_ms(100);
    }
}

```

图 17 外部中断代码

### 2.3.2 定时器程序代码

如下图为定时器初始化程序：

```

void shumaguan_init(){
    TMOD = 0x01;
    TH0 = 0x3C; // 初值高8位就是65536-计时值50000 (50毫秒)
                // = 15536转换成16进制为 3cb0
    TL0 = 0xB0; // 高8位为3c, 低8位为b0
    ET0 = 1;    // 开启定时器0
    EA = 1;    // 开启总中断
    TR0 = 1;    // 开启定时器
}

```

图 18 数码管初始化

如图所示，这是一个定时器中断程序，旨在每 50ms 触发一次中断。当中断触发 20 次时，计时器将累积为 1 秒，依此类推实现时钟功能。



```

// 定时器0中断处理函数
void Timer0_ISR(void) interrupt 1 {
    uint8_t count = 0;

    // 重载定时器初值（每50ms触发一次中断）
    TH0 = 0x3C; // 定时器高8位（3CB0，表示50000微秒）
    TL0 = 0xB0; // 定时器低8位

    count_timer0++; // 每50ms中断，count加1

    // 1秒钟计时：如果count达到20，则秒钟加1
    if (count_timer0 == 20) {
        second++; // 秒钟加1
        count_timer0 = 0; // 重置计数器
    }

    // 处理秒钟溢出
    if (second == 60) {
        minute++; // 分钟加1
        second = 0; // 秒钟重置
    }

    // 处理分钟溢出
    if (minute == 60) {
        hour++; // 小时加1
        minute = 0; // 分钟重置
    }

    // 处理小时溢出
    if (hour == 24) {
        hour = 0; // 小时重置
        // 可扩展：添加天、月、年等更长时间单位
    }
}

```

图 19 定时器中断

### 2.3.3 LCD 显示程序

对于静态显示部分，本程序通过 for 循环遍历将日期 2025-1-13 的每个字符依次显示在 LCD 的第一行。具体来说，首先通过命令 `xml(0x80)` 设置光标到第一行的最左侧，然后逐个显示日期字符串中的字符。每个字符通过 `xdat()` 函数被写入 LCD，并且每写入一个字符后，程序延时 50 毫秒，以确保字符稳定显示。

对于滚动显示部分，本程序将 `str2` 字符串通过循环实现滚动效果。在 LCD 的第二行，字符从左到右滚动显示，使用 `xml(0xC0 + j)` 动态改变光标位置，逐个字符写入并实现滚动显示。每滚动一次后，程序延时 500 毫秒，从而让显示效果更加平滑。

```

unsigned char str1[]={"2025-1-13"};
unsigned char str2[]={"HelloWorld!imlx"};
void delay_lcd(){
    set_guangbiao(0x80);
    for(i=0;i<=8;i++)
    {
        xdat(str1[i]);
        delay_ms(50);
    }
    for(j=15;j>=0;j--)
    {
        set_guangbiao(0xC0+j);
        for(i=0;i<=14;i++)
        {
            xdat(str2[i]);
            delay_ms(50);
        }
        delay_ms(500);
    }
}

```

图 20 LCD 显示函数

### 2.3.4 ADC 数据采集代码

为了实现实时采集并更新显示，本系统使用了定时器中断机制来触发 ADC 数据

采集。具体的定时器配置如下：

```
// 初始化ADC和定时器相关设置的函数
void Init_ADC_and_Timer() {
    ET0 = 1; // 使能定时器0中断
    TR0 = 1; // 启动定时器0
    ET1 = 1; // 使能定时器1中断
    TR1 = 1; // 启动定时器1
    EA = 1; // 使能总中断

    CLK = 0;
    CS = 1;
}
```

图 21 定时器及中断初始化

在每次定时器中断发生时，程序会自动调用 Get\_AD\_Result() 函数进行 ADC 数据采集。该函数通过配置时钟信号和控制信号来读取 ADC 的数据转换结果。采集到的结果会被转化为电压值，并显示在数码管上。

其中对于 ADC 采集部分的时序代码如下：

```
unsigned char Get_AD_Result()
{
    uchar i;
    uchar data1=0,data2=0;
    CS=0;

    CLK=0;DIO=1;
    CLK=1;

    CLK=0;DIO=1;
    CLK=1;

    CLK=0;DIO=1;
    CLK=1;DIO=0;

    CLK=0;DIO=1;

    for(i=0;i<8;i++)
    {
        CLK=1;
        CLK=0;
        data1=(data1<<1)|(uchar)DIO;
    }
    CS=1;
    return data1;
}
```

图 22 ADC 转化代码

主程序如下图所示：

```
void main(void)
{
    uint value;
    Init_ADC_and_Timer();
    while(1)
    {
        AD1=Get_AD_Result();
        value= (5000/256.0)*AD1;
        date[0] =dis_adc[value/1000];
        date[1] =dis_adc[value/100%10];
        date[2] =dis_adc[value%100/10];
        date[3] =dis_adc[value%10];
        DigDisplay();
    }
}
```

图 23 AD 转换主程序

### 2.3.5 串口通信部分

串口和定时器初始化部分代码如下图：

```
void InitUART(void) {
    PCON &= 0x7F;      //波特率不倍速
    SCON = 0x50;        //8位数据,可变波特率
    TMOD &= 0x0F;        //清除定时器1模式位
    TMOD |= 0x20;        //设定定时器1为8位自动重装方式
    TL1 = 0xFD;         //设定定时初值
    TH1 = 0xFD;         //设定定时器重装值
    TR1 = 1;            //启动定时器1
    ET1 = 0;            //禁止定时器1中断
    EA=1;               //开放总中断
    ES=1;               //开放串口中断
}
```

图 24 串口初始化代码

要实现单片机发送字符串给串口助手，只需要编写好 UART\_SendByte 和 UART\_SendStr 函数即可，并修改串口中断函数，代码如下：

```
void UART_ISR() interrupt 4//串口中断函数
{
    if(RI==1)//接收中断
    {
        SendByte(SBUF); //把接收的数据发送到计算机
        RI=0; //软件清零
    }
}
```

图 25 串口中断

```
void SendByte(unsigned char dat) {
    SBUF = dat;
    while (!TI);
    TI = 0;
}

void SendStr(unsigned char *s) {
    while (*s != '\0') { // 发送字符串
        SendByte(*s);
        s++;
    }
}
```

图 26 串口收发函数

## 第三章 51 单片机系统效果演示

由于前文涉及的程序较多且内容复杂，为了提高代码的集成性并便于演示，演示过程将所有功能程序集成到一个统一的代码文件中。通过矩阵键盘扫描的形式，从按键 1 至 5，当按下相应的按键时，程序会初始化对应的寄存器，执行对应的功能。这种方式简化了操作并提高了演示的便捷性。

### 3.1 中断程序演示

当按下按键 1 时，两位四显数码管会显示 "0"。随后，当按下 INT3.2 按键触发外部中断时，数码管显示的数值会逐步递增，每次加 1，从 "0" 开始一直递增到 "F"。依此类推，每次按下 INT3.2，数码管显示的数值都会按顺序递增。

下图分别为没按下按键时，按下两次时，按下五次时的数码管显示情况。

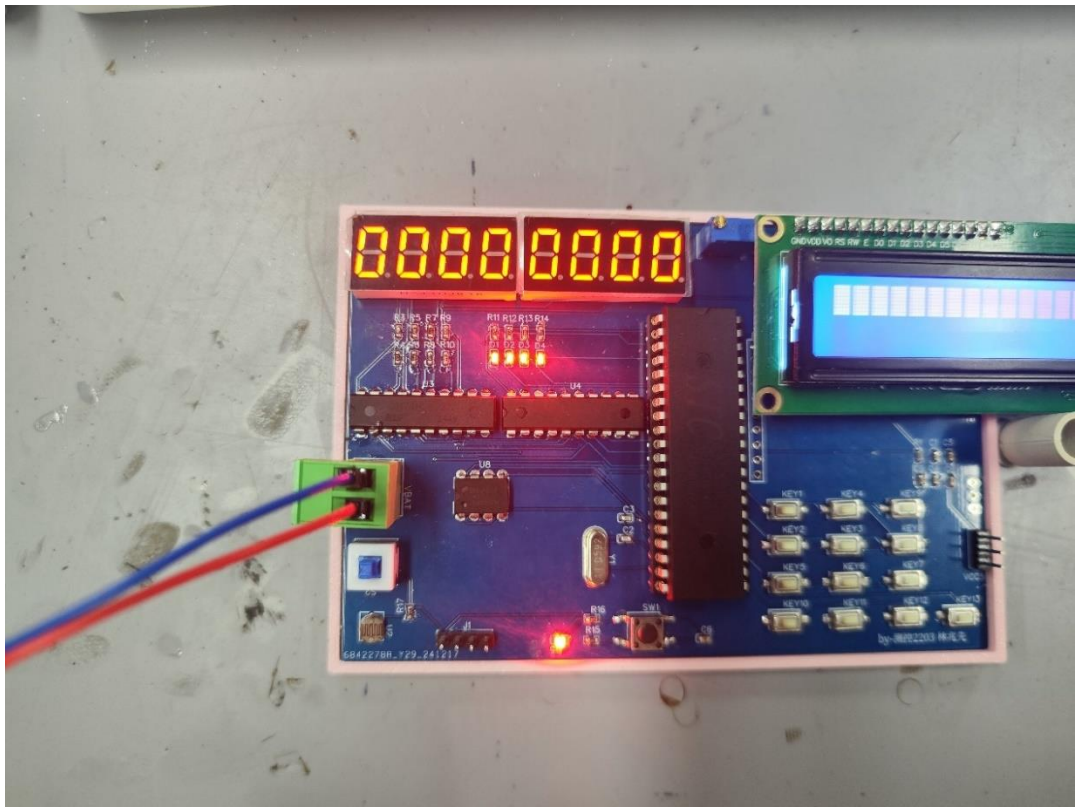


图 27 数码管 1



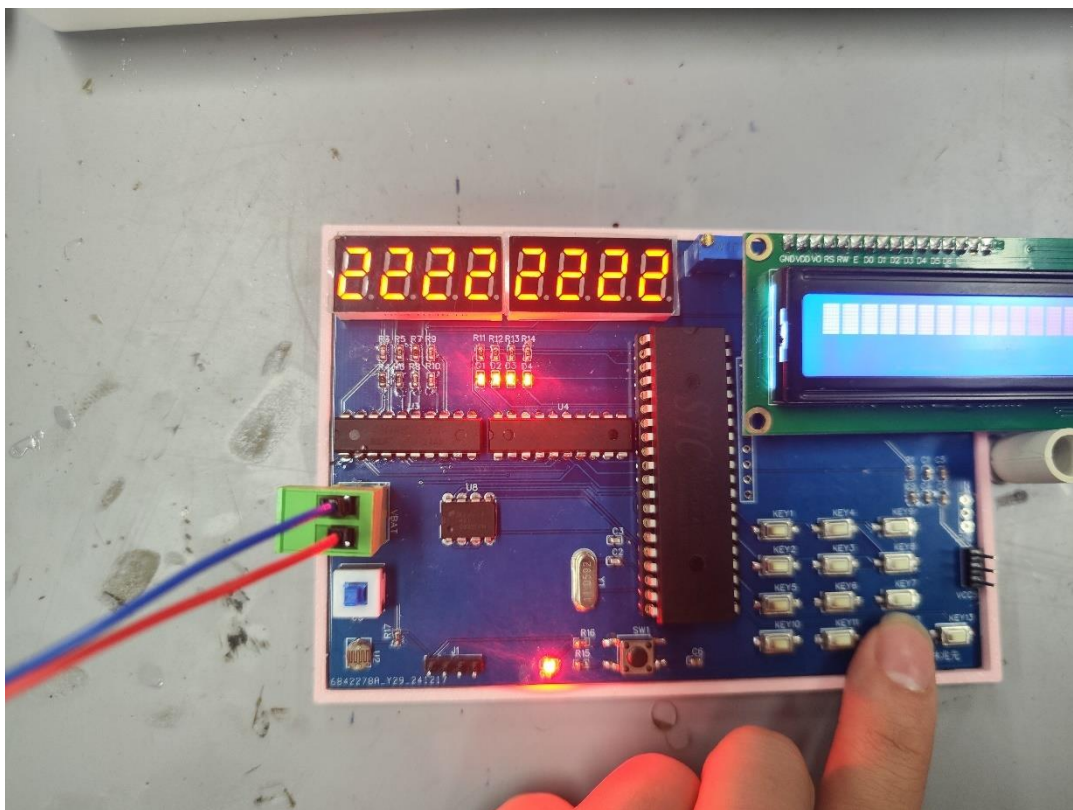


图 28 数码管 2

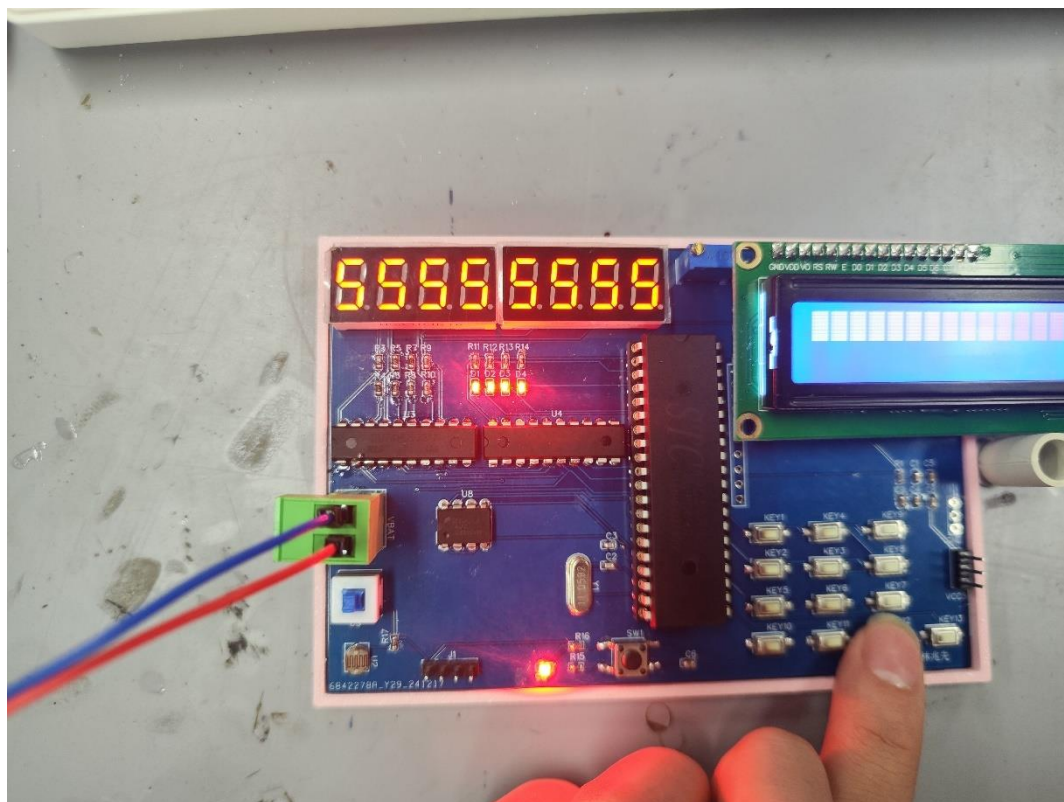


图 29 数码管 3

## 3.2 定时器程序演示

当按下按键 2 时，两位四显数码管将显示 "10095950"，表示当前时间为 10 号 9 点 59 分 50 秒，并实现数码管的时钟动态显示功能，时间会随着秒钟的变化而实时更新。

图 和图 分别为显示为过了 3 秒和 11 秒后的演示效果，

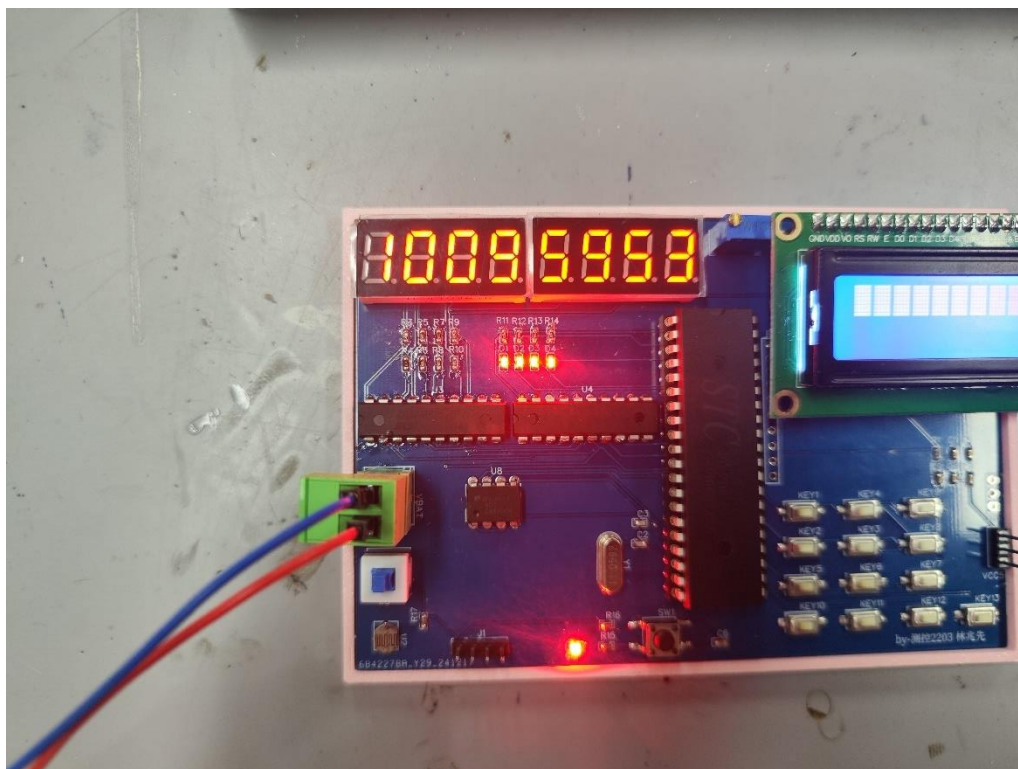


图 30 定时器演示

## 3.3 LCD 程序演示

当按下按键 3 时，LCD 第一行显示当前日期为 “2025-1-13”，第二行滚动显示 “HelloWorldimlzx”。LCD 程序演示效果如下：



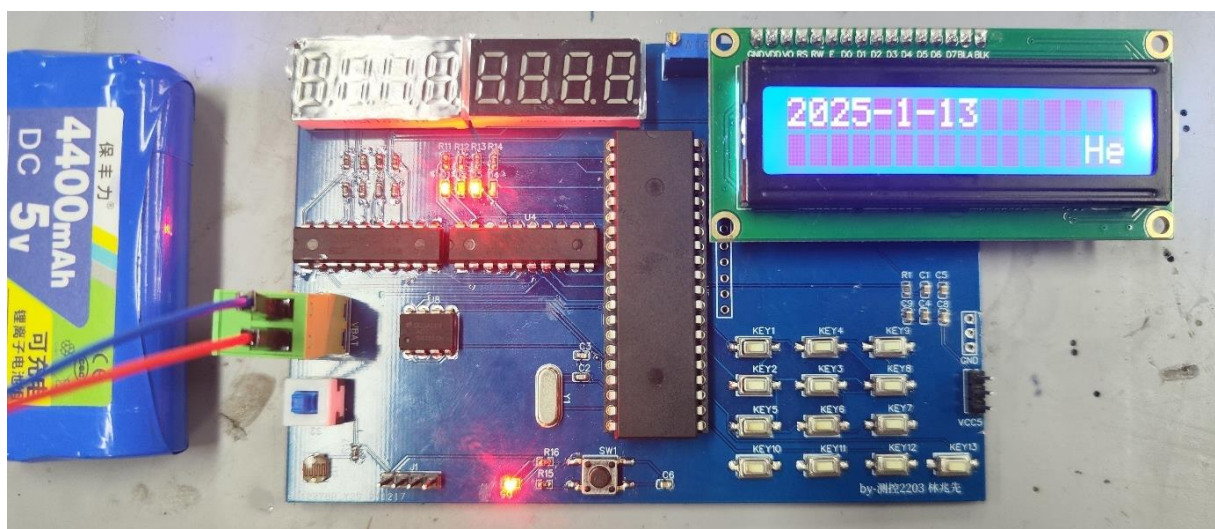


图 31 LCD 演示

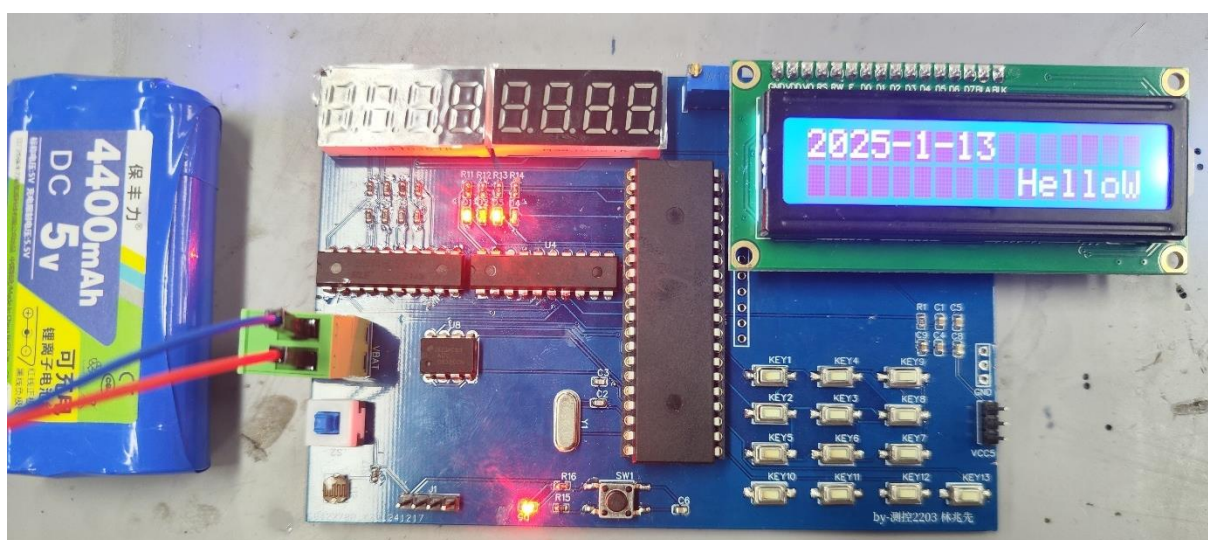


图 32 LCD 演示

### 3.4 ADC 采样程序演示

ADC 采集演示效果如下图，当对光敏电阻无遮挡时，电压采样值如数码管显示，约为 2109mv；当使用手对光敏电阻进行部分遮挡，电压采样值变大，约为 4199mV。

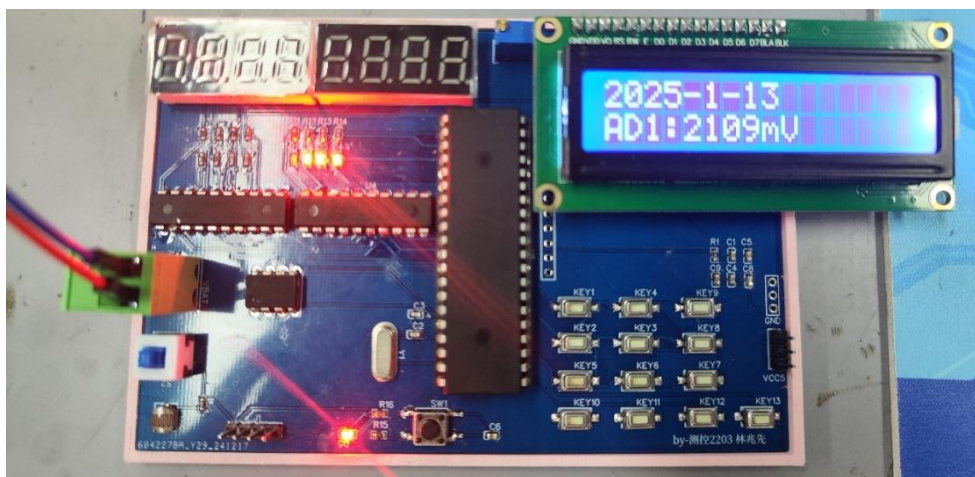


图 33 ADC 采样

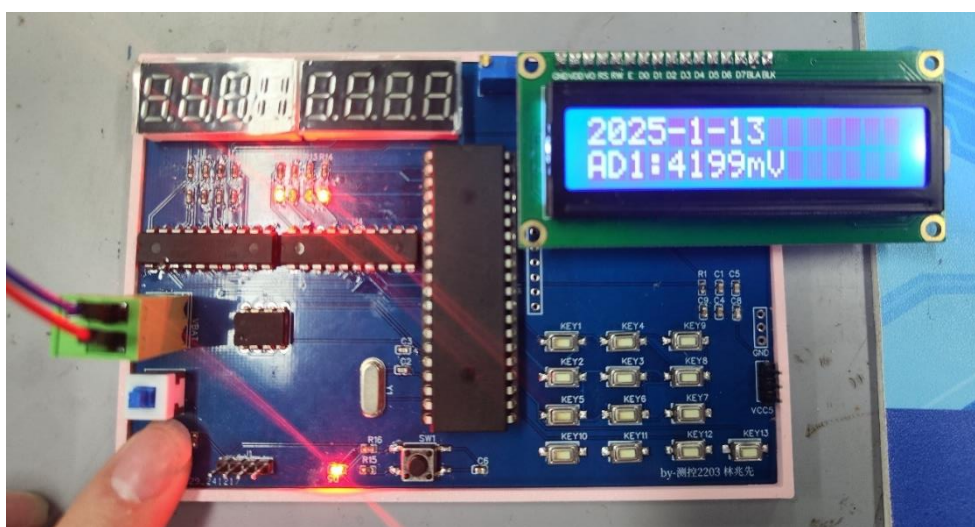


图 34 ADC 采样

### 3.5 串口通信程序演示

按下按键 5 时，单片机会完成串口通信所需寄存器的初始化，并通过串口发送“Hello”给上位机作为初始化完成的标志。当上位机接收到“Hello”时，即表示串口通信初始化成功。通信效果如下图所示：





图 35 串口通讯

当向下位机发送一串字符，例如“Hello,imlzx”时，下位机会将上位机发送的字符串逐字符存储到 SBUF 寄存器中，并通过串口回发给上位机串口助手进行显示。显示效果如下：



图 36 串口通讯

# 第四章 智能门禁系统总体设计方案

根据任务书要求，本次单片机课程设计以智能门禁系统为拓展内容，旨在通过软硬件结合的方式实现多功能的门禁管理。本系统设计了便捷的人机交互界面，支持多种验证方式，包括按键密码输入、RFID 卡验证和蓝牙通信控制，满足不同场景的门禁需求。通过嵌入式程序控制和硬件电路的配合，系统能够完成身份验证、门锁控制、数据记录等功能，同时结合 OLED 显示和按键交互，实现门禁的智能化设计。

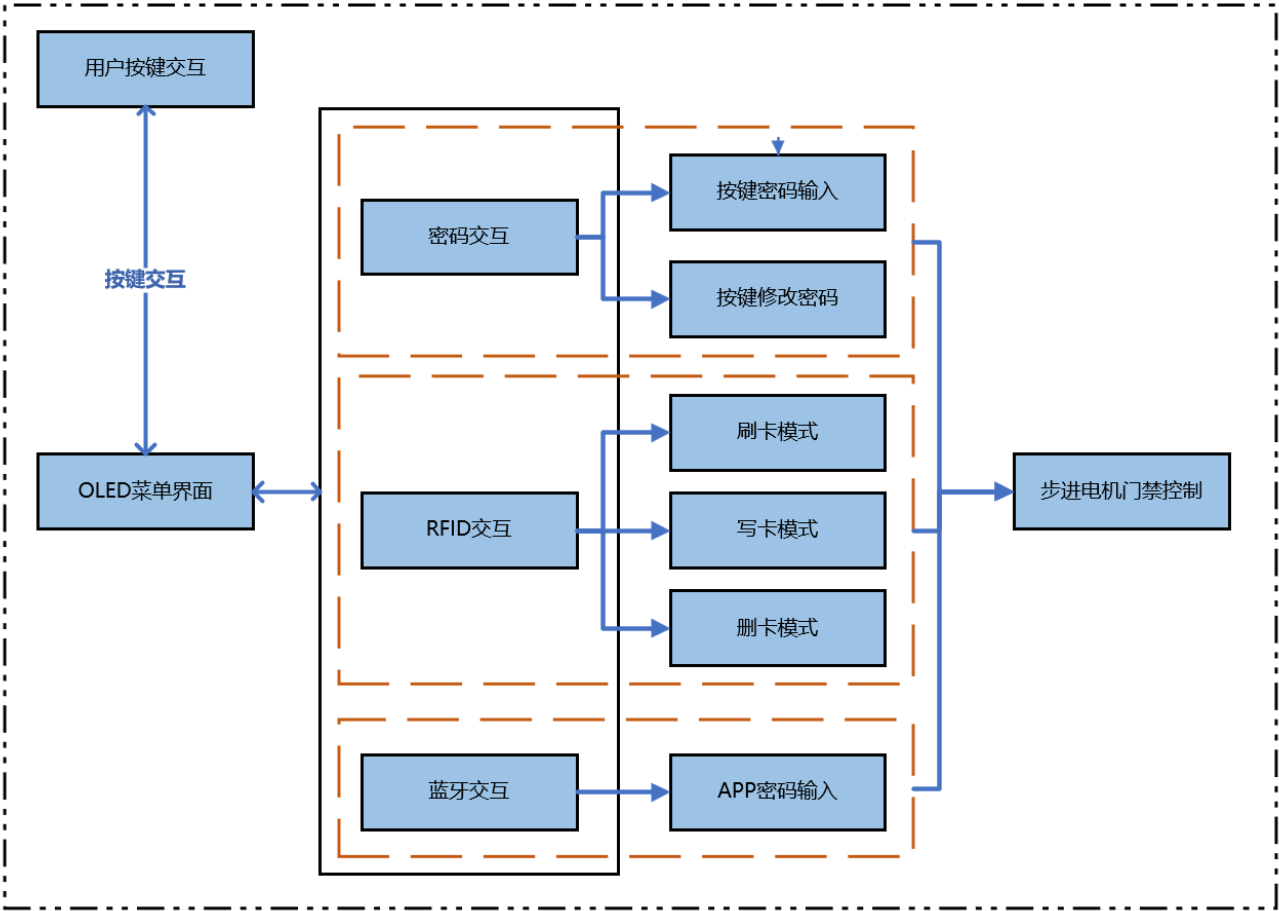


图 37 智能门禁系统框架图

## 4.1 各模块设计方案及对比分析

### 4.1.1 单片机主控板方案

在智能门禁系统的设计中，选择合适的主控板至关重要。主控板需要具备足够的性能、合理的功耗以及良好的开发支持。为此，我们对比了市面上三款常见的单片机主控板：STM32F103C8T6、C51（AT89C51）和 Arduino Uno，并结合项目需求进行了分析。

表 6 主控板对比方案

对比项	STM32F103C8T6	C51（AT89C51）	Arduino Uno
适用场景	复杂控制任务，高性能需求	简单应用，低成本设计	教育、快速开发，小型项目
功耗	支持低功耗模式，适合长时间工作	功耗较高，不适合长时间工作	功耗介于 STM32 和 C51 之间，适合一般应用
成本	约 10-15 元人民币（取决于购买渠道）	非常低，约 3-5 元人民币	约 30-50 元人民币

根据设计需求，本次项目涉及按键输入、蓝牙通信、RFID，并且具备继续扩展的能力，例如未来可能加入人脸识别功能。考虑到项目的中等难度，主控板需要提供强大的算力以及便捷的开发方式。因此，综合考虑性能、功耗和扩展性，选择了 STM32F103C8T6 作为主控板，它能够满足项目目前的需求，并具有较强的开发潜力。

4.1.2 门禁开关方案

在智能门禁系统的设计中，选择合适的门禁开关方案对系统的可靠性和实用性至关重要。门禁开关方案通常包括舵机开关和步进电机开关两种选择。我们对比了这两种方案，并根据项目的需求进行了分析。

表 7 门禁开关方案

对比项	舵机开关	步进电机开关
精度	精度较低，通常适用于小范围角度控制	高精度，能够实现较大范围的精准控制
使用场景	适用于简单的门禁开关、低精度控制场景	适用于高精度控制、大范围门禁开关
可靠性	机械结构简单，长期使用可能会受磨损影响	步进电机控制稳定，可靠性较高，适合长期使用

根据项目需求，智能门禁系统需要实现较为精确的开关控制，同时保证系统的可靠性和扩展性。不同于普通的舵机，步进电机采用脉冲的方式驱动，其基本步距角为 1.8°/步，配上半步驱动器后，步距角减少为 0.9°，配上细分驱动器后其步距角可细分达 256 倍（0.007°/微步），因此精度极高。

对于步进电机的方案，由于传统的步进电机需要外接驱动，接线和布置较为复杂，因此我们选择了将驱动和电机集成一体的 ZDT 步进电机。该电机自带闭环控制系统，能够实现精准的角度闭环调节，极大地简化了接线和系统布置。

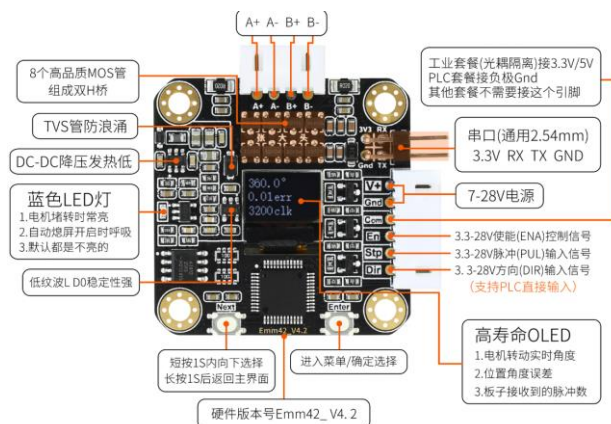


图 38 ZDT 步进电机

#### 4.1.3 开门方式设计

本次开门方式设计采用三种方式：密码输入、RFID 刷卡、蓝牙通讯开启。这是本次设计的基本功能。

在完成这些部分逻辑的搭建后，我还考虑到对于 STM32 单片机，由于单片机会存在断电数据不保存的问题，会导致写入的 UID 在单片机重启后出现掉电丢失的现象。为了解决这个问题，根据数据手册，我们利用 STM32 单片机 Flash 存储器掉电不丢失数据的特点，使用软件将写入的 UDI 保存到 Flash 中，实现了密码以及 RFID 的 UID 长期保存的效果。

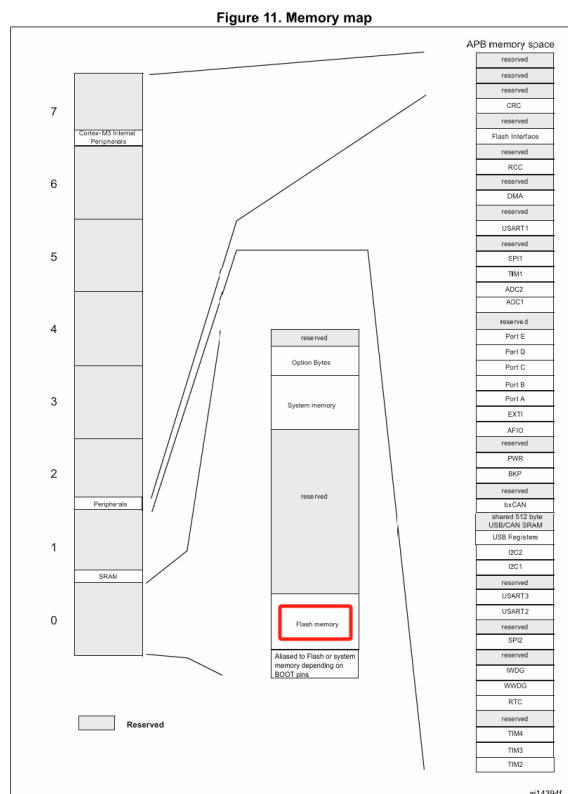


图 39 STM32Flash 寄存器

#### 4.1.3.1 按键输入设计

对于开门方式，第一种采用按键输入的方式。方式较简单，本次设计了四个按键，分别代表 1~4，能够兼容与 OLED 交互的多级菜单及案件密码输入的功能。

当用户输入四位密码时，系统读取 Flash 中的密码单元，并判断其是否和用户输入的密码一致。

#### 4.1.3.2 RFID 刷卡设计

本次 RFID 设计了三种方式，刷卡开门、写卡模式、闪卡模式。硬件部分使用市面上应用广泛的 RFID 模块。

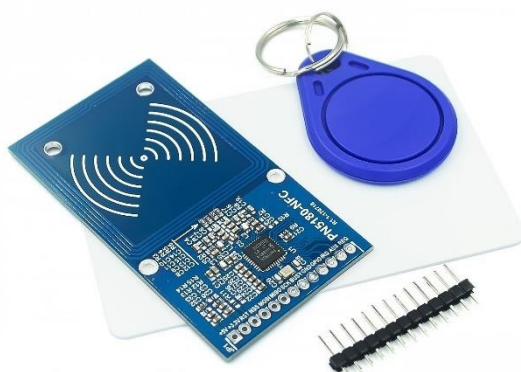


图 40 RFID 模块

刷卡开门模式是 RFID 门禁系统中最常见的功能之一。在该模式下，当用户持有授权卡片时，只需将卡片靠近 RFID 读卡器，系统便会通过读取 Flash 中的唯一识别信息（UID）进行身份验证。如果验证成功，控制器将发出开门信号，驱动电锁开门<sup>[3]</sup>。

写卡模式允许管理员将特定的数据写入空白或可写的 RFID 卡片，用于增加新的授权用户或修改卡片的存储信息。本次课程设计为了方便演示，将 RFID 的写卡功能设置为用户可访问功能。

同时，对于写卡模式，由于单片机会存在断电数据不保存的问题，会导致写入的 UID 在单片机重启后出现掉电丢失的现象。为了解决这个问题，使用软件将写入的 UID 保存到 Flash 中，实现了 UID 长期保存的效果。

删卡模式，也是结合 STM32 微控制器的 Flash 存储特性。删卡模式相当于使用软件擦除的方式，将 STM32 的 Flash 寄存器内指定的 UID 数据指定擦除，即可实现删卡的效果。

#### 4.1.3.3 蓝牙通信方案比较及设计

对于蓝牙通讯方式，系统提供了两种可选方案。第一种方案是使用市面上常见的 HC05/HC06 蓝牙模块，通过串口通讯方式与 STM32 进行连接。这种方案的优点是模块成本低、应用广泛，适合一般的蓝牙数据传输需求<sup>[4]</sup>。第二种方案是选择具备蓝牙功能的开发板，它集成了蓝牙模块，可以直接处理蓝牙端的接收和发送逻辑，简化了硬件连接和配置过程，同时也提高了系统的稳定性和可扩展性。这种方案适用于需要更高集成度和灵活性的场景。

本次实现使用自带开发板，选用市面上成本较低的 ESP32 开发板，使用 Arduino IDE 开发，利用 Arduino 强大的集成库，相比于使用 STM32 进行蓝牙模块的配置，其具有极大的开发便利性。

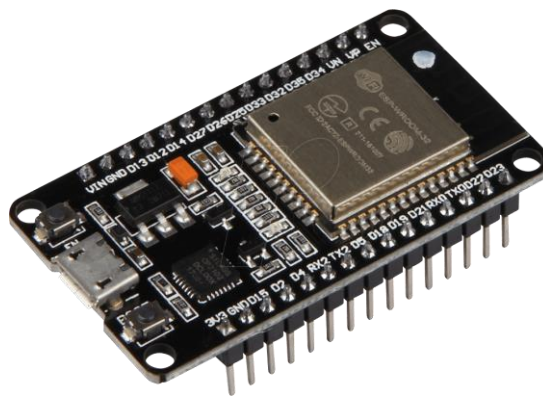


图 41 ESP32

对于 ESP32 开发板，系统需要在开发板上处理来自用户的输入逻辑，并通过发送标志位的方式与 STM32 系统进行交互，从而实现对开关锁的控制。这种方式简便高效，使得操作逻辑能够在 ESP32 上完成并传递给 STM32。因此，其中关键部分是实现 ESP32 与用户蓝牙端之间的交互逻辑<sup>[5]</sup>。

当上述逻辑建立后，对于蓝牙的设计逻辑大致为，当用户接入蓝牙，可以在 APP 上实现交互过程，比如密码输入、密码逻辑的判断等。ESP32 对其进行逻辑处理，进而将处理的结果通过 UART 通信发送给 STM32 系统，进而进行开关门的逻辑判断。

## 4.2 各模块原理图绘制

### 4.2.1 MCU 部分

MCU 部分选用了 STM32F103C8T6 芯片，并通过自绘原理图和 PCB 设计实现了自主焊接。与市面上一些常见的开发板相比，本次自主设计的开发板具有更高的通用性，特别是在接口设计方面引入了 Type-C 接口，使其在现代设备中更具兼容性。此

外，开发板所有元器件均采用贴片式小型封装，进一步提高了板载元件的密度和稳定性。开发板还集成了 I2C 通信接口，用于直接插接 OLED 显示模块，便于实现高效且稳定的 OLED 显示功能<sup>[6]</sup>。

其原理图如下图所示：

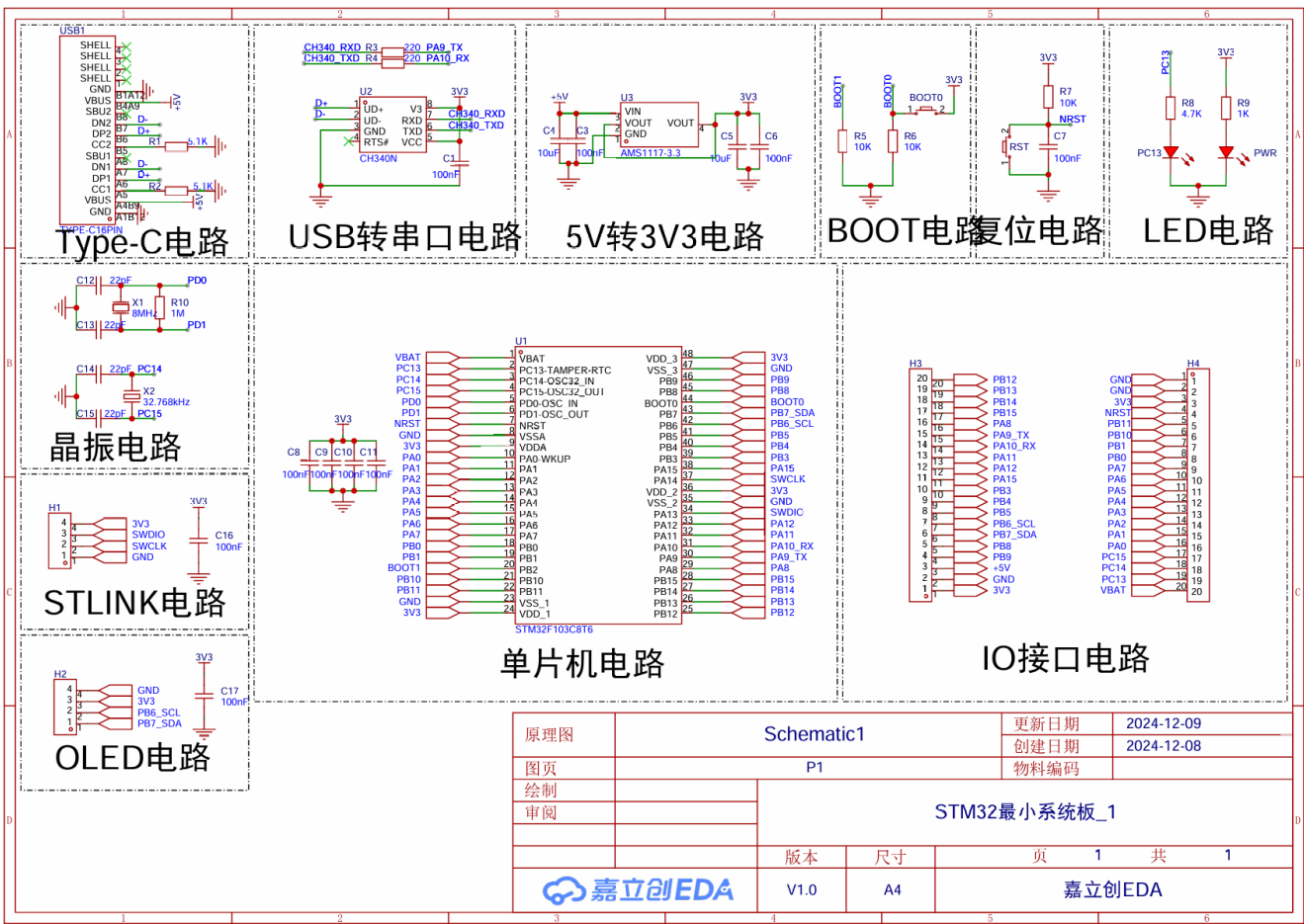


图 42 STM32F103C8T6 原理图

其 PCB 和实物如下图所示：



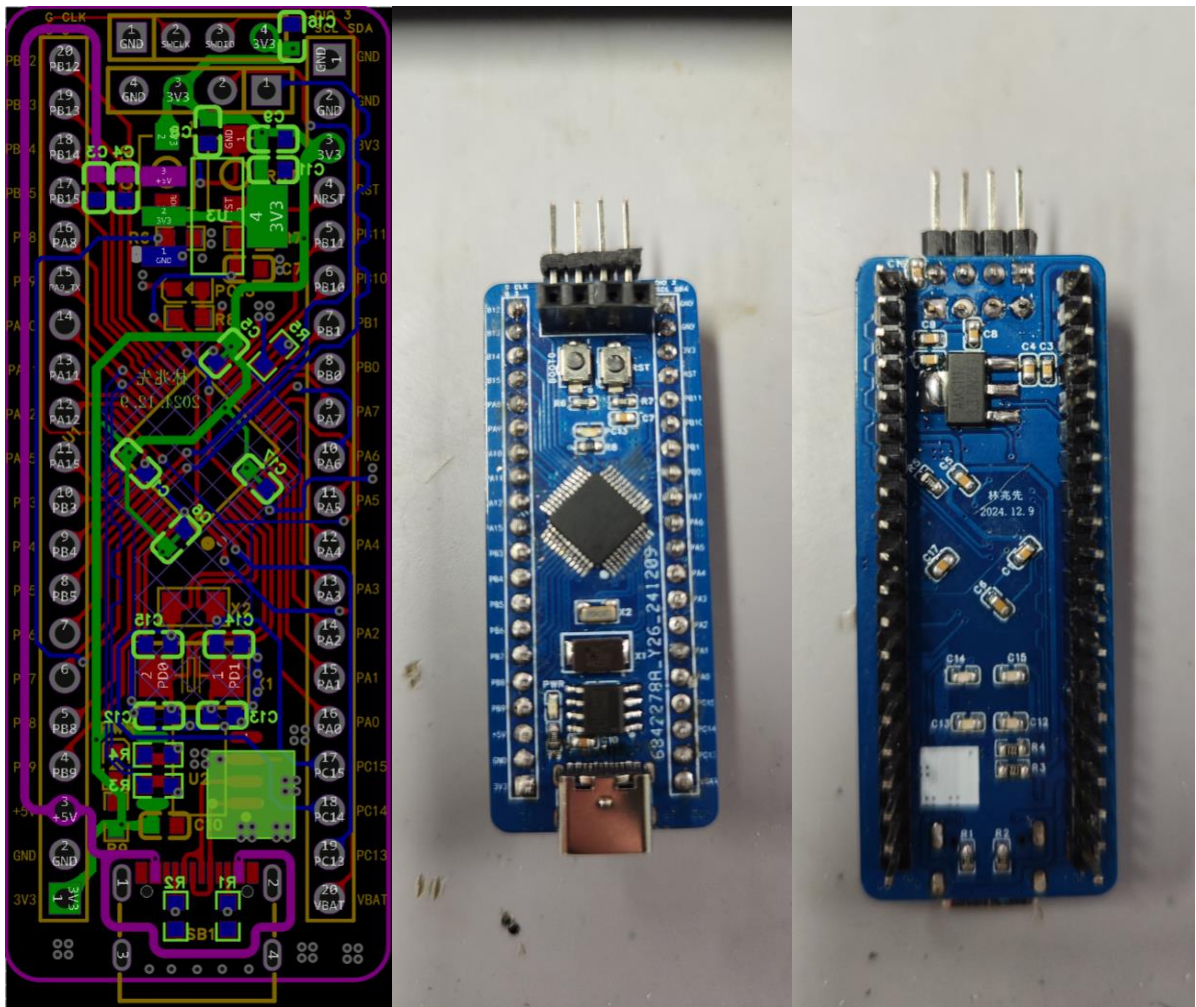


图 43 原理图及实物图

其中，在 STM32 系统的开发板中，STM32F103C8T6 采用直插式设计，只需将开发板引脚连接至对应的外设接口即可。

STM32 系统原理图如下：





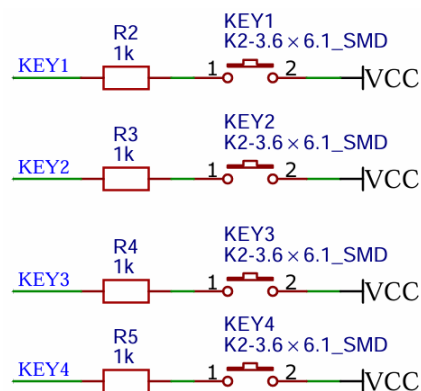


图 46 按键原理图

#### 4.2.4 RFID 原理图设计

RFID 采用 SPI 通信，将 RFID 的 SPI 引脚连接至 STM32 最小系统板对应的 SPI 引脚即可。

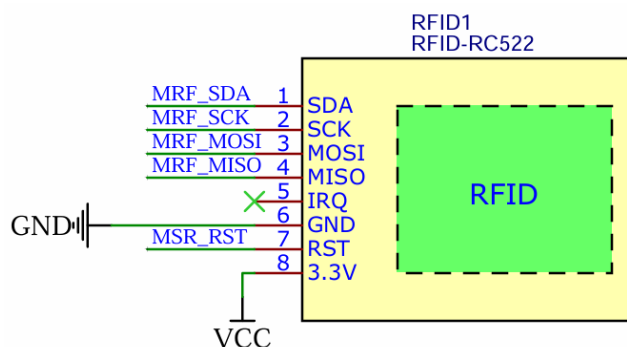


图 47 RFID 原理图

#### 4.2.5 蓝牙部分原理图设计

蓝牙通讯采用直插式设计，将串口通信引脚 RX/TX 与单片机引脚相连接即可。

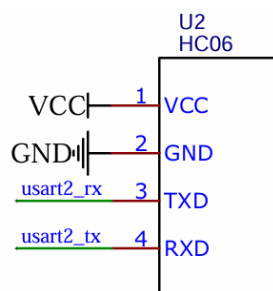


图 48 HC06 原理图

#### 4.2.6 步进电机原理图

引出步进电机所需的 3 个普通 I/O 口和供电口即可。

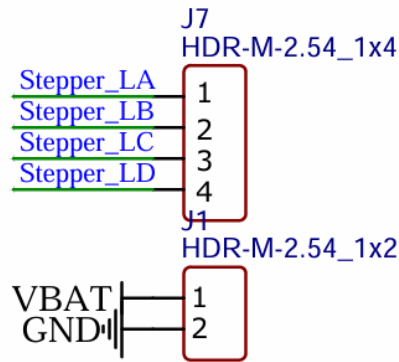


图 49 步进电机原理图

#### 4.2.7 其它直插式原理图

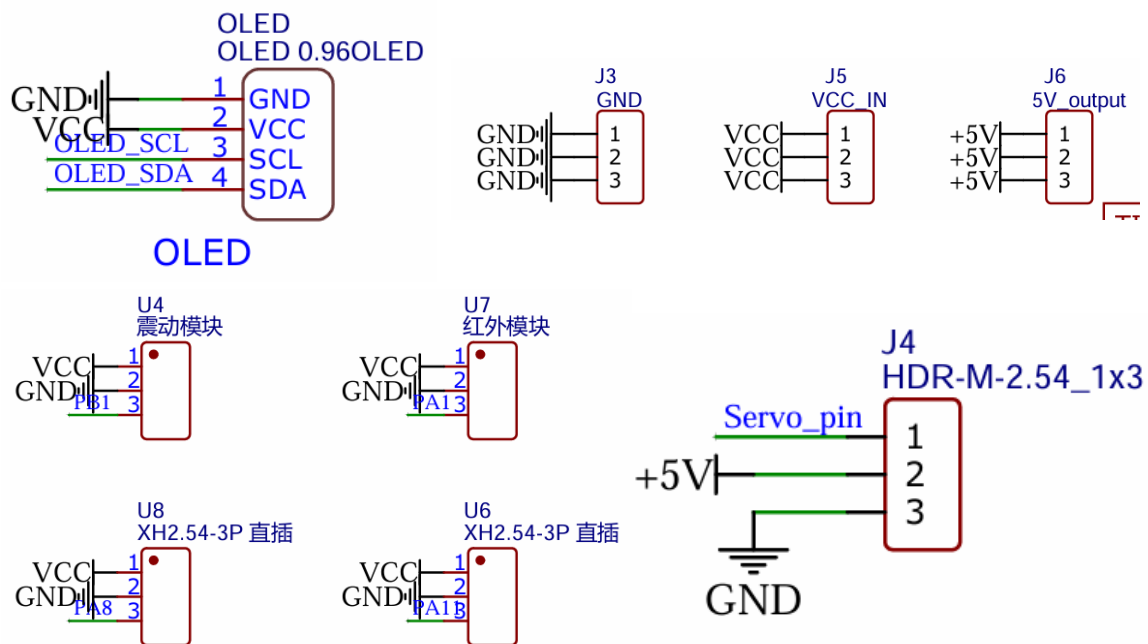


图 50 直插式部分原理图

#### 4.2.8 整体原理图和 PCB 设计

下图为 STM32 系统板原理图：

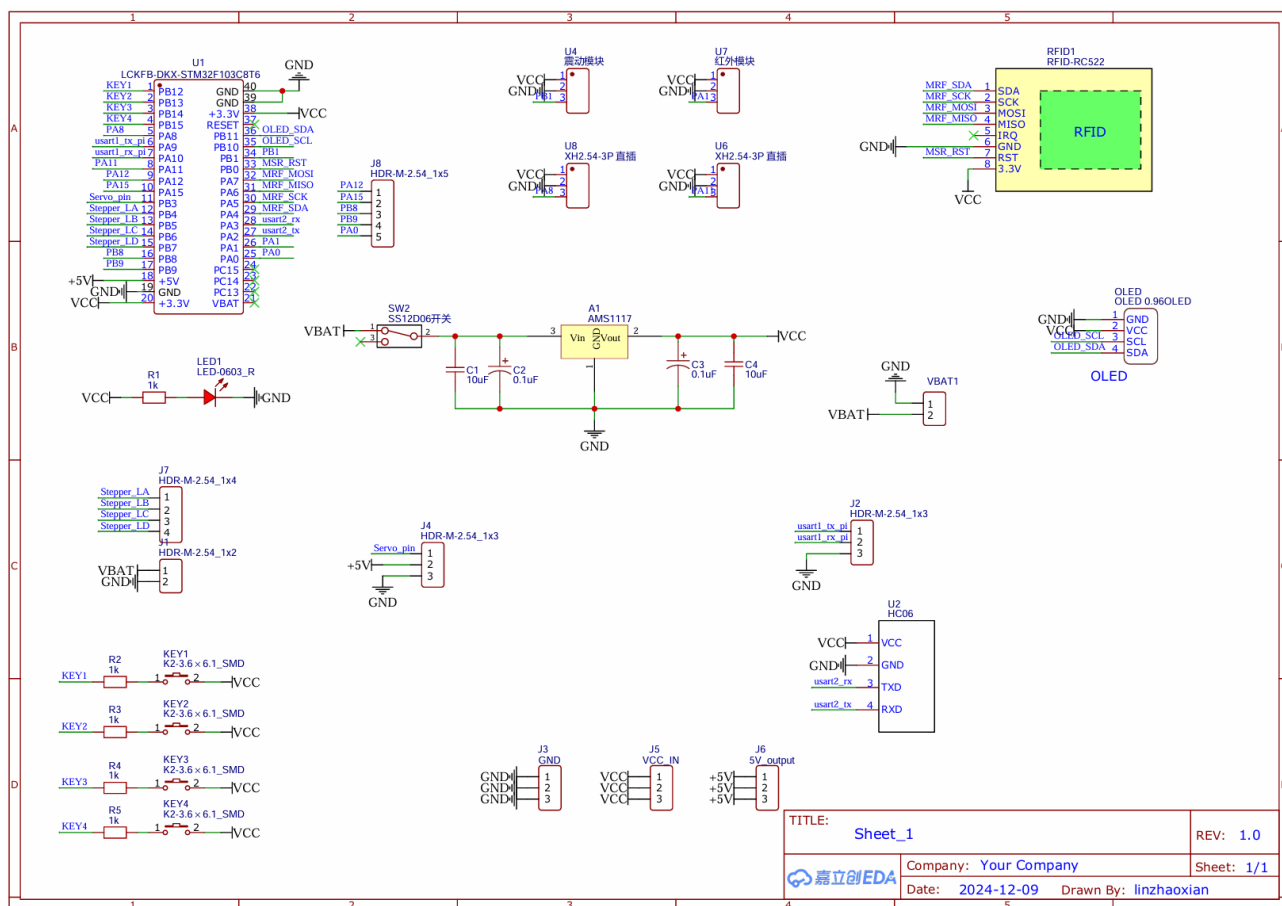


图 51 STM32 系统板原理图

下图为 STM32 系统 PCB 和实物部分：

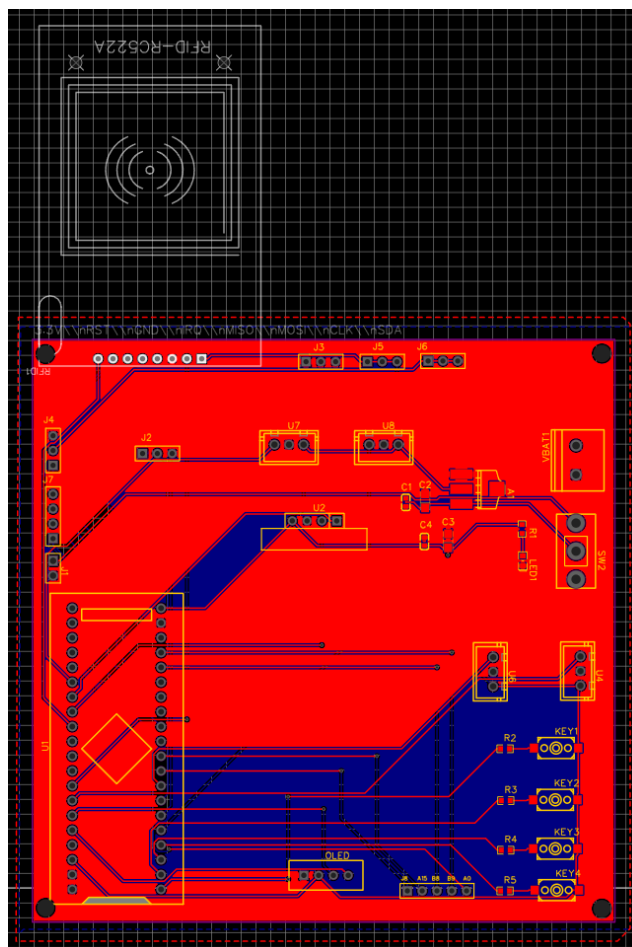


图 52 STM32 系统板 PCB

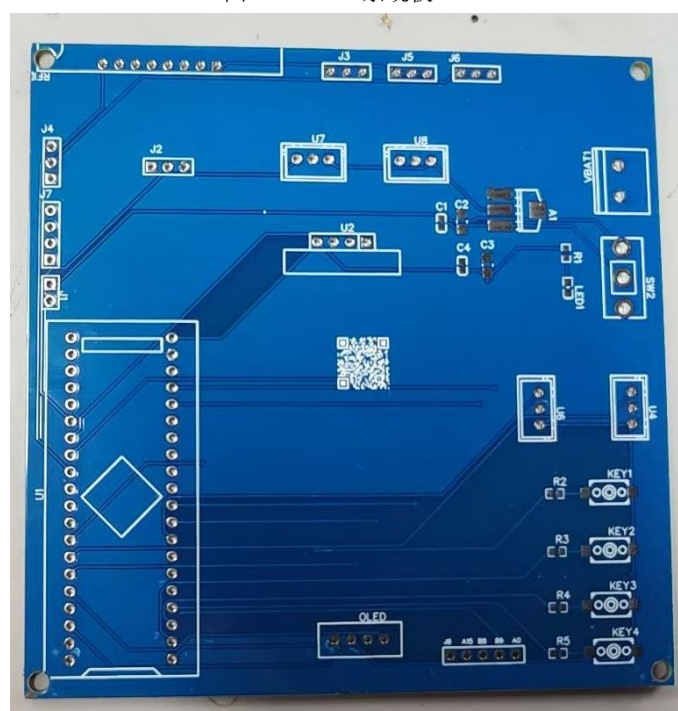


图 53 STM32 系统板实物图

### 4.3 门禁菜单交互设计

本系统的核心功能模块包括交互界面、按键控制、蓝牙通讯以及 RFID 控制四个关键部分。下文将主要介绍这四个模块以及一些拓展部分的设计方案及实现细节，阐述其如何协同工作以实现系统的整体功能。

交互界面设计通过将交互界面呈现再 0.96 寸的 OLED 上，并结合按键的方式去实现。为此给系统编写了适配的菜单选项，使得用户能够通过键操作在不同的菜单选项之间进行选择。为了实现这一功能，我们为系统编写了一个二级菜单，使得用户可以通过按下不同的按键来浏览和选择界面中的选项。

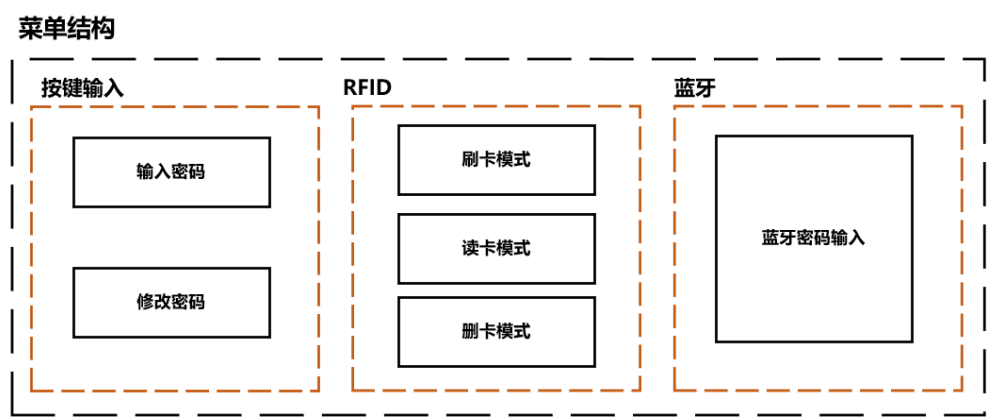


图 54 门禁菜单系统图

#### 4.3.1 一级菜单界面

在智能门禁系统的一级菜单界面中，用户可以选择三种不同的身份验证方式，具体包括：1. Password（密码输入）；2. RFID（射频识别）；3. BlueTooth（蓝牙输入）。

代码如下所示。通过在 while 循环中持续检测用户按下的按键，并利用 switch 语句来控制 OLED 显示的内容，从而实现用户通过按键在不同菜单之间的导航。这样，系统能够根据用户的选择动态更新显示界面，允许用户顺利进入各个菜单层级。

```

void displayMainMenu(void)
{
    OLED_ShowString(1, 3, "Door Locked");
    OLED_ShowString(2, 1, "1.Password");
    OLED_ShowString(3, 1, "2.RFID");
    OLED_ShowString(4, 1, "3.BlueTooth");
}

if (!inPasswordMode) {
    switch (KeyNum) {
        case 1:
            OLED_Clear();
            inPasswordMode = 1; // 按键密码进入
            TIM_ITConfig(TIM2, TIM_IT_Update, DISABLE); //关闭时钟
            break;
        case 2:
            OLED_Clear();
            OLED_ShowString(1, 1, "RFID");
            inPasswordMode = 2; // RFID进入
            TIM_ITConfig(TIM2, TIM_IT_Update, DISABLE);
            break;
        case 3:
            OLED_Clear();
            inPasswordMode = 3; // 蓝牙进入
            TIM_ITConfig(TIM2, TIM_IT_Update, DISABLE);
            break;
    }
}

```

图 55 一级菜单代码

### 4.3.2 二级菜单界面

首先，按键输入密码的界面是在一级菜单中实现的。当用户按下按键 1 时，系统会跳转至密码输入界面，OLED 屏幕会显示“Pls enter pd:”，此时用户通过按键输入 4 位密码进行身份验证。

对于 RFID 菜单部分，系统提供了多项功能，包括刷卡开门、添加卡片 ID 和删除卡片 ID。用户可以通过按键 1 至 3 来选择不同操作，既可以执行开门操作，也可以对已有的 RFID 卡片进行管理（如添加或删除卡片 ID）。

在蓝牙部分，当用户选择蓝牙菜单时，系统会自动判断是否通过手机 APP 的蓝牙串口成功连接到 STM32 的蓝牙模块。如果已连接，系统将自动开启蓝牙设备的检索模式，准备与其他蓝牙设备进行通信。

## 4.4 门禁系统开关方式设计

当用户按下按键 1 时，系统会根据用户所按下的 1 至 4 按键，系统会实时更新屏幕，在 OLED 屏幕上显示输入的密码。密码输入完成后，系统会根据预设的验证逻辑判断密码是否正确。

## 按键输入

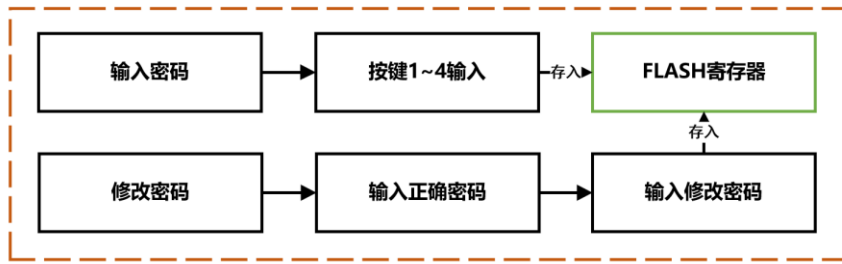


图 56 按键输入设计

具体的代码验证逻辑如下：

```

if (KeyNum >= 1 && KeyNum <= 4 && passwordPos < PASSWORD_LENGTH && !board_move_mode) {
    password_client[passwordPos] = KeyNum;
    passwordPos++;
    OLED_ShowNum(2, passwordPos + 1, KeyNum, 1);

    if (passwordPos == PASSWORD_LENGTH) {
        uint8_t isCorrect = 1; // 检查密码是否正确
        for (uint8_t i = 0; i < PASSWORD_LENGTH; i++) {
            if (password_client[i] != correct_password[i]) {
                isCorrect = 0; // 密码错误
                break;
            }
        }
        Delay_ms(1000);
        OLED_Clear();
        OLED_ShowString(1, 1, "Checking...");
        Delay_ms(1000);
        OLED_Clear();
        if (isCorrect) {
            OLED_ShowString(1, 1, "Yes");
            Delay_ms(500);
            // 密码正确，执行相关操作
            board_move_mode = 1; // 进入逻辑判断阶段
        } else {
            OLED_ShowString(1, 1, "No");
            error_count++; // 记录错误次数
            Delay_ms(500);
            // 密码错误，执行相应提示
            board_move_mode = 0;
        }
    }
}

} else if (board_move_mode == 1) {
    rotate90_degrees(); // 开门
}

```

图 57 门禁开关逻辑

密码修改逻辑代码如下：

```

if (!password_display_flag) {
    password_display_flag = 1;
    OLED_ShowString(1, 1, "Change Password");
    OLED_ShowString(2, 1, "Pls enter cur pd:"); // 提示用户输入当前密码
}

// 输入当前密码
if (KeyNum >= 1 && KeyNum <= 4 && passwordPos < PASSWORD_LENGTH && !isCorrect_flag) {
    password_client[passwordPos] = KeyNum; // 存储用户输入的密码位
    passwordPos++;
    OLED_ShowNum(2, passwordPos + 1, KeyNum, 1); // 显示输入的密码位

    // 如果已经输入了完整的当前密码，进行密码校验
    if (passwordPos == PASSWORD_LENGTH && !password_change_flag) {
        isCorrect_change_pd = 1; // 假设密码正确，进行比对
        uint8_t stored_password[PASSWORD_LENGTH];
        LoadPasswordFromFlash(stored_password); // 从Flash读取密码

        for (uint8_t i = 0; i < PASSWORD_LENGTH; i++) {
            if (password_client[i] != stored_password[i]) {
                isCorrect_change_pd = 0; // 密码错误
                break;
            }
        }

        Delay_ms(1000); // 显示延时，给用户一些缓冲时间
        OLED_Clear();
        OLED_ShowString(1, 1, "Checking..."); // 提示正在验证密码
        Delay_ms(1000);
        OLED_Clear();

        if (isCorrect_change_pd) {
            // 当前密码正确，允许用户输入新密码
            passwordPos = 0; // 清空输入位置，准备输入新密码
            password_change_flag = 1;
            isCorrect_flag = 1;
            OLED_Clear();
            OLED_ShowString(1, 1, "Pls enter new pd:"); // 提示输入新密码
            first_change_key = 1;
        } else {
            // 当前密码错误，重新开始输入
            OLED_ShowString(1, 1, "Incorrect password");
            Delay_ms(500);
            passwordPos = 0; // 清空输入密码
            OLED_Clear();
            OLED_ShowString(1, 1, "Pls enter pd:"); // 提示重新输入当前密码
        }
    }
}

// 如果输入正确的当前密码，继续输入新密码
if (isCorrect_change_pd && passwordPos < PASSWORD_LENGTH && password_change_flag == 1) {
    if (first_change_key == 1) {
        first_change_key = 0;
        KeyNum = 0;
        passwordPos = 0;
    }

    if (KeyNum >= 1 && KeyNum <= 4) { // 确保按键未被处理
        new_password[passwordPos] = KeyNum; // 存储新密码的当前位
        passwordPos++;
        OLED_ShowNum(2, passwordPos + 1, KeyNum, 1); // 显示新密码位
    }

    // 当前密码输入完成时
    if (passwordPos == PASSWORD_LENGTH && password_change_flag == 1) {
        // 完成新密码的输入，进行确认
        password_change_flag = 2; // 标记新密码输入完成，准备进行确认
    }

    // 当前密码输入完成时
    if (isCorrect_change_pd && passwordPos == PASSWORD_LENGTH && password_change_flag == 2) {
        // 完成新密码的输入，进行确认
        OLED_Clear();
        OLED_ShowString(1, 1, "New password set:"); // 显示设置成功的信息

        // 保存新密码
        for (uint8_t i = 0; i < PASSWORD_LENGTH; i++) {
            correct_password[i] = new_password[i]; // 更新密码
        }

        // 将新密码写入Flash
        SavePasswordToFlash(correct_password);

        Delay_ms(1000); // 显示修改成功，延时1秒
        OLED_Clear();
        displayMainMenu(); // 返回主菜单
        password_display_flag = 0; // 清除密码显示标志
        passwordPos = 0; // 清空密码输入位置
        inPasswordMode_PD_Mode = 0; // 退出密码修改模式
        inPasswordMode = 0;
        password_change_flag = 0;
        isCorrect_change_pd = 0;
    }
}

```

图 58 修改密码逻辑代码



### 4.4.1 RFID 设计

本系统的 RFID 模块设计作为门禁系统的核心组成部分，负责通过射频识别技术实现身份验证和门禁管理。用户可以通过刷卡来验证身份，并执行如开锁、添加卡片以及删除卡片等操作。以下部分详细描述了 RFID 模块的实现方案和操作流程，以及如何通过按键控制不同的功能<sup>[7]</sup>。

当用户选择进入 RFID 模式时，OLED 屏幕会显示 RFID 模式的主界面，提示用户选择以下操作：

1. Scan：进入刷卡验证模式，系统会通过 RFID 模块扫描用户的卡片并进行身份验证。
2. Add card：进入添加卡片模式，用户可以将新的 RFID 卡片添加到系统中。
3. Delete：进入删除卡片模式，用户可以删除已添加的卡片信息。

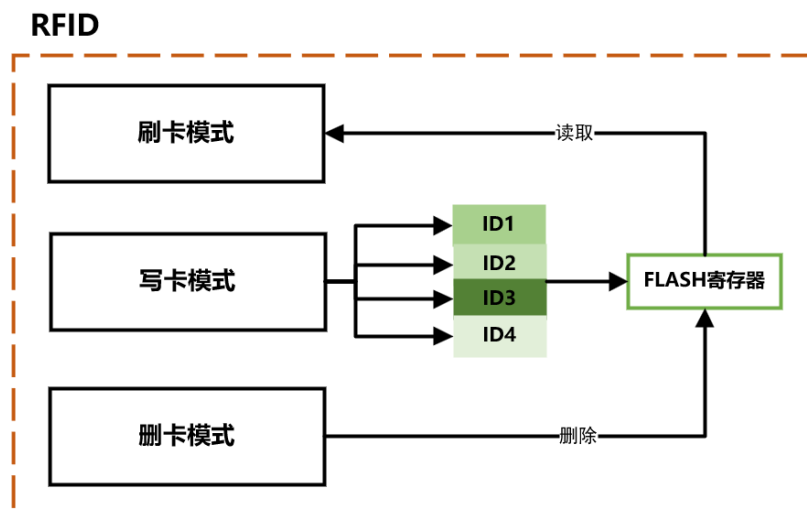


图 59 RFID 设计系统图

在该界面中，系统会实时监听按键输入，根据用户的选择来切换操作模式。用户可以通过按键 1、2 和 3 来选择不同的功能，而按键 4 可以随时退出当前操作并返回到主菜单。

```

// 按键处理
if (KeyNum == 1) { // 进入刷卡模式
    flag_scan = 1;
    flag_addcard = 0;
    flag_deletcard = 0;
}
if (KeyNum == 2) { // 进入写卡模式
    flag_scan = 0;
    flag_addcard = 1;
    flag_deletcard = 0;
}
if (KeyNum == 3) { // 进入删卡模式
    flag_scan = 0;
    flag_addcard = 0;
    flag_deletcard = 1;
}
if (KeyNum == 4) // 按下4键退出删卡模式，进入寻卡模式
{
    OLED_Clear();

    flag_scan = 0;
    flag_addcard = 0;
    flag_deletcard = 0;

    inPasswordMode = 0;
    OLED_Clear();
    displayMainMenu();
    isFirstEnter = 1;
    rfid_display_flag = 0;
}
}

```

图 60 RFID 逻辑代码

#### 4.4.1.1 刷卡设计

在选择“Scan”模式后，系统会等待用户刷卡并读取 RFID 卡片的 ID。通过比对卡片的 ID，系统判断是否允许开锁。

```

void RFID_Check()
{
    if (!rfid_mode) {
        cardnumber = Rc522Test(); // 获取卡片编号
        if (isFirstEnter) {
            OLED_Clear(); // 清屏
            isFirstEnter = 0;
        }
        if (cardnumber == 0) // 如果卡片无效（返回0），则显示错误信息
        {
            OLED_ShowString(1, 1, "Error card ");
            Buzzer_Alarm(); // 发出警报音
            WaitCardOff(); // 等待卡片移开
        }
        // 如果卡片编号是1到4，表示有效卡片
        else if ((cardnumber == 1 || cardnumber == 2 || cardnumber == 3 || cardnumber == 4))
        {
            OLED_Clear();
            OLED_ShowString(1, 1, "The CardID is: ");
            OLED_ShowNum(1, 15, cardnumber, 2);
            OLED_ShowString(3, 1, "opening...");
            Delay_ms(500);
            rfid_mode = 1; // 开门标志位
        }
        else
            OLED_ShowString(3, 3, "Scanning...");
    }
}

```

图 61 RFID 读卡代码

#### 4.4.1.2 写卡设计

用户进入“Add card”模式时，系统将扫描并验证 RFID 卡片 ID。若卡片未添加过，系统会将卡片 ID 存储在 Flash 中，提示用户“Add Card X OK”；若卡片已存在，则提示“Card exists”。

```

while (flag_addcard == 1) { // 添加卡片
    if (isFirstEnter) {
        OLED_Clear(); // 清屏
        isFirstEnter = 0;
    }

    OLED_ShowString(1, 1, " Add Card: ");

    KeyNum = Key_Scan();
    if (KeyNum == 1) { // 返回到扫描模式
        flag_scan = 1;
        flag_addcard = 0;
        flag_deletecard = 0;
    }
    if (KeyNum == 3) { // 返回到删除卡片模式
        flag_scan = 0;
        flag_addcard = 0;
        flag_deletecard = 1;
    }
    if (KeyNum == 4) // 按下4键退出删卡模式, 进入寻卡模式
    {
        flag_scan = 0;
        flag_addcard = 0;
        flag_deletecard = 0;

        OLED_Clear();

        inPasswordMode = 2;
        isFirstEnter = 1;
        rfid_display_flag = 0;
    }

    case 2:
        UI2[0] = UID[0];
        UI2[1] = UID[1];
        UI2[2] = UID[2];
        UI2[3] = UID[3];
        FLASH_W(FLASH_ADDR3, UI2[0], UI2[1], UI2[2], UI2[3]);
        OLED_ShowString(1, 1, " Add Card 3 OK ");
        Buzzer1();
        WaitCardOff();
        OLED_Clear();
        break;
    case 3:
        UI3[0] = UID[0];
        UI3[1] = UID[1];
        UI3[2] = UID[2];
        UI3[3] = UID[3];
        FLASH_W(FLASH_ADDR4, UI3[0], UI3[1], UI3[2], UI3[3]);
        OLED_ShowString(1, 1, " Add Card 4 OK ");
        Buzzer1();
        WaitCardOff();
        OLED_Clear();
        break;
    case 4:
        OLED_ShowString(1, 1, "Card max!!");
        Buzzer1();
        Delay_ms(500);
        OLED_Clear();
        break;
    case 5:
        OLED_ShowString(1, 1, "Card exists!");
        Buzzer1();
        Delay_ms(500);
        OLED_Clear();
        break;
}

if (PcdRequest(REQ_ALL, Temp) == MI_OK) {
    if (PcdAnticoll(UID) == MI_OK) {
        // 检查卡片ID是否为已有卡片
        if (UID[0] == 0xFF && UID[1] == 0xFF && UID[2] == 0xFF && UID[3] == 0xFF)
            tempcard = 0;
        else if (UI1[0] == 0xFF && UI1[1] == 0xFF && UI1[2] == 0xFF && UI1[3] == 0xFF)
            tempcard = 1;
        else if (UI2[0] == 0xFF && UI2[1] == 0xFF && UI2[2] == 0xFF && UI2[3] == 0xFF)
            tempcard = 2;
        else if (UI3[0] == 0xFF && UI3[1] == 0xFF && UI3[2] == 0xFF && UI3[3] == 0xFF)
            tempcard = 3;
        else
            tempcard = 4;

        // 判断卡片是否已存在
        if (UID[0] == UI0[0] && UID[1] == UI0[1] && UID[2] == UI0[2] && UID[3] == UI0[3])
            tempcard = 5;
        if (UID[0] == UI1[0] && UID[1] == UI1[1] && UID[2] == UI1[2] && UID[3] == UI1[3])
            tempcard = 5;
        if (UID[0] == UI2[0] && UID[1] == UI2[1] && UID[2] == UI2[2] && UID[3] == UI2[3])
            tempcard = 5;
        if (UID[0] == UI3[0] && UID[1] == UI3[1] && UID[2] == UI3[2] && UID[3] == UI3[3])
            tempcard = 5;

        switch (tempcard) {
            case 0:
                UI0[0] = UID[0];
                UI0[1] = UID[1];
                UI0[2] = UID[2];
                UI0[3] = UID[3];
                FLASH_W(FLASH_ADDR1, UI0[0], UI0[1], UI0[2], UI0[3]); // 写入闪存
                OLED_ShowString(1, 1, " Add Card 1 OK ");
                Buzzer1();
                WaitCardOff(); // 等待卡片移开
                OLED_Clear();
                break;
            case 1:
                UI1[0] = UID[0];
                UI1[1] = UID[1];
                UI1[2] = UID[2];
                UI1[3] = UID[3];
                FLASH_W(FLASH_ADDR2, UI1[0], UI1[1], UI1[2], UI1[3]);
                OLED_ShowString(1, 1, " Add Card 2 OK ");
                Buzzer1();
                WaitCardOff();
                OLED_Clear();
                break;
        }
    }
}

```

图 62 RFID 写卡代码

#### 4.4.1.3 删卡设计

用户选择“Delete”模式后，系统会提供删除指定卡片的选项。用户通过按键选择要删除的卡片，系统会清除对应的卡片 ID 数据，并提示“Clear Card X OK”。

```

while(flag_deletecard==1) //删卡模式
{
    if (isFirstEnter) {
        OLED_Clear(); // 清屏
        isFirstEnter = 0;
    }
    OLED_ShowString(1,1," Delete Card: ");
    KeyNum = Key_Scan();

    if(KeyNum==4) //按下4键退出删卡模式，进入寻卡模式
    {
        flag_scan=0;
        flag_addcard=0;
        flag_deletecard=0;

        OLED_Clear();

        inPasswordMode = 2;
        isFirstEnter = 1;
        rfid_display_flag = 0;
    }

    if(KeyNum==1) //在删卡模式下按下1键，选择要删除的卡片，卡片序列增加
    {
        select++;
        if(select>=4 || select<1)select=0;
    }
    if(KeyNum==2) //在删卡模式下按下2键，选择要删除的卡片，卡片序列减少
    {
        select--;
        if(select>=4 || select<1)select=0;
    }

    case 2:
    {
        OLED_ShowString(2,1," Delete Card 3 ?");
        if(KeyNum==3)
        {
            FLASH_Clear(FLASH_ADDR3);
            UI2[0]=0xFF;
            UI2[1]=0xFF;
            UI2[2]=0xFF;
            UI2[3]=0xFF;
            OLED_ShowString(2,1,"Clear Card 3 OK ");
            Buzzer1();
            Delay_ms(1500);
        }
        else if(KeyNum==4){
            OLED_Clear();
            break;
        }
    }
    }break;
    case 3:
    {
        OLED_ShowString(2,1," Delete Card 4 ?");
        if(KeyNum==3)
        {
            FLASH_Clear(FLASH_ADDR4);
            UI3[0]=0xFF;
            UI3[1]=0xFF;
            UI3[2]=0xFF;
            UI3[3]=0xFF;
            OLED_ShowString(2,1,"Clear Card 4 OK ");
            Buzzer1();
            Delay_ms(1500);
        }
        else if(KeyNum==4){
            OLED_Clear();
            break;
        }
    }
    }break;
    default:break;
}

switch (select)
{
    case 0:
    {
        OLED_ShowString(2,1," Delete Card 1 ?");
        if(KeyNum==3) //在删卡模式下按下3键，删除对应的卡片
        {
            FLASH_Clear(FLASH_ADDR1);
            UI0[0]=0xFF;
            UI0[1]=0xFF;
            UI0[2]=0xFF;
            UI0[3]=0xFF;
            OLED_ShowString(2,1,"Clear Card 1 OK ");
            Buzzer1(); //删除成功后蜂鸣器响一声
            Delay_ms(1500);
        }
        else if(KeyNum==4){
            OLED_Clear();
            break;
        }
    }
    }break;
    case 1:
    {
        OLED_ShowString(2,1," Delete Card 2 ?");
        if(KeyNum==3)
        {
            FLASH_Clear(FLASH_ADDR2);
            UI1[0]=0xFF;
            UI1[1]=0xFF;
            UI1[2]=0xFF;
            UI1[3]=0xFF;
            OLED_ShowString(2,1,"Clear Card 2 OK ");
            Buzzer1();
            Delay_ms(1500);
        }
        else if(KeyNum==4){
            OLED_Clear();
            break;
        }
    }
    }break;
}

```

图 63 RFID 删卡代码

## 4.4.2 蓝牙通讯设计

蓝牙通讯设计分为两部分，一部分为 STM32 和 ESP32 进行串口通讯，另一部分为 ESP32 和蓝牙串口助手 APP 通讯。实现流程当用户点击“BlueTooth”菜单选项时，STM32 会开始接收从 ESP32 传输的信息，以判断手机用户是否连接至蓝牙。当用户连接 ESP32 的蓝牙模块后，用户可以通过 APP 输入密码，从而实现门的开关控制。

### 4.5.2.1 STM32 与 ESP32 串口通讯

首先，需要先配置好 STM32 和 ESP32 的串口通讯寄存器。随后，需要书写双机通讯的逻辑部分。

对于 STM32 的串口中断部分，实现了对 ESP32 所传输的字符储存，代码如下图所

示:

```
void USART2_IRQHandler(void) {
    if (USART_GetITStatus(USART2, USART_IT_RXNE) == SET && inPasswordMode==3) {
        char receivedChar = USART_ReceiveData(USART2); // 读取接收到的数据

        // 检查是否接收到结束符
        if (receivedChar == PASSWORD_END_CHAR) {
            receivedPassword[passwordIndex] = '\0'; // 添加字符串结束符
            passwordReceived = true; // 设置标志
            passwordIndex = 0; // 重置索引
        } else {
            if (passwordIndex < BUFFER_SIZE - 1) { // 检查缓冲区是否有足够空间
                receivedPassword[passwordIndex++] = receivedChar; // 存储接收到的字符
            } else {
                // 缓冲区已满, 清空缓冲区
                ClearPasswordBuffer();
                // 在OLED上显示错误信息
                OLED_ShowString(4, 1, "Password Too Long!");
            }
        }
    }
}
```

图 64 UART 通讯代码

对于 ESP32 的串口部分, 实现了向 STM32 字符串的发送, Arduino 端的代码如下图所示:

```
Serial.begin(115200); // 初始化串口, 用于调试输出
Serial1.begin(9600, SERIAL_8N1, 16, 17); // 初始化串口1 (D16为RX, D17为TX)
// 将密码发送给单片机
Serial1.print(receivedPassword); // 使用 Serial1.print 确保逐字发送
Serial1.write(' '); // 追加发送空格作为结束符

// 清空缓冲区
memset(receivedPassword, 0, sizeof(receivedPassword));
passwordIndex = 0;
```

图 65 ESP32 蓝牙初始化代码

#### 4.4.2.2 ESP32 与手机蓝牙串口通讯

由于 Arduino IDE 中已经有兼容 ESP32 的蓝牙库函数, 因此直接进行调用即可。

部分通讯交互代码如下图所示:

```
void checkBluetoothStatus() {
    // 检查蓝牙连接状态
    if (SerialBT.connected()) {
        if (!bluetoothConnected) {
            bluetoothConnected = true; // 蓝牙刚刚连接成功
            Serial.println("蓝牙连接成功!");
            SerialBT.println("蓝牙已连接, 请输入密码: ");
            // 不再发送 'A'
        }
    } else {
        if (bluetoothConnected) {
            bluetoothConnected = false; // 蓝牙断开连接
            Serial.println("蓝牙连接断开!");
        }
        // 不再发送 'B'
    }
}

// 检查蓝牙串口是否有数据可读
while (SerialBT.available()) {
    char receivedChar = SerialBT.read();

    // 检测输入结束符 (空格)
    if (receivedChar == ' ') {
        receivedPassword[passwordIndex] = '\0'; // 添加字符串结束符

        // 显示接收到的密码
        Serial.print("接收到的密码: ");
        Serial.println(receivedPassword);

        // 密码验证
        if (strcmp(receivedPassword, correctPassword) == 0) {
            Serial.println("密码正确");
            digitalWrite(led, HIGH);
            SerialBT.println("密码正确, 欢迎!");
            attemptCount = 0; // 重置错误计数
        } else {
            Serial.println("密码错误");
            digitalWrite(led, LOW);
            attemptCount++;

            if (attemptCount >= maxAttempts) {
                lastAttemptTime = millis();
                SerialBT.println("错误次数过多, 已锁定输入30秒!");
            } else {
                SerialBT.print("密码错误, 剩余尝试次数: ");
                SerialBT.println(maxAttempts - attemptCount);
            }
        }
    }
}
```

图 66 ESP32 蓝牙通讯代码

### 4.4.2.3 蓝牙控制菜单代码

如下图为蓝牙控制的菜单逻辑代码，结合交互界面实现门禁的控制。该段代码实现了蓝牙模式下的密码验证和门禁操作。当用户通过蓝牙输入密码时，系统验证密码是否正确。

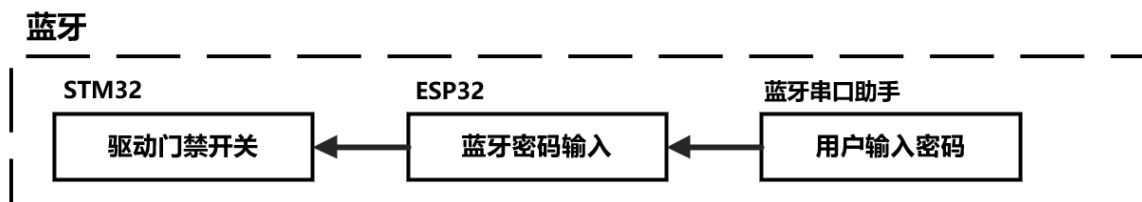


图 67 蓝牙控制流程图

若正确，显示“Opening door...”并执行开门操作；

若错误，显示提示信息要求重新输入。

密码验证后，系统根据不同状态执行步进电机转动、显示门已打开信息，并在操作完成后返回主菜单。通过这种方式，系统实现了蓝牙控制的门禁功能，并确保在各个阶段的状态切换和信息提示。

```
else if (inPasswordMode == 3) { // 蓝牙模式
    if (isFirstEnter) {
        OLED_Clear(); // 清屏
        isFirstEnter = 0;
        OLED_ShowString(1, 1, "Connecting...");
    }
    if (passwordReceived && !blue_tooth_mode) {
        // 比较接收到的密码是否正确
        if (strcmp(receivedPassword, "1234") == 0) {
            // 密码正确
            OLED_Clear();
            OLED_ShowString(1, 1, "Yes");
            // 显示 Opening door
            OLED_ShowString(2, 1, "Opening door...");
            // 关闭USART2接收中断
            USART_ITConfig(USART2, USART_IT_RXNE, DISABLE);
        }
        // 密码错误
        else {
            // 密码错误
            OLED_Clear();
            OLED_ShowString(1, 1, "No");
            // 保持 TIM2 中断关闭
            Delay_ms(2000);
            OLED_Clear();
            OLED_ShowString(1, 1, "Waiting for Password:");
            OLED_ShowString(2, 1, "Password: ");
        }
    }
    // 更新last_receivedPassword (可选, 根据需求)
    strcpy(last_receivedPassword, receivedPassword, BUFFER_SIZE);
    last_receivedPassword[BUFFER_SIZE - 1] = '\0'; // 确保字符串结束
    // 重置标志
    passwordReceived = false;
    // 清空密码缓冲区, 准备下次接收
    ClearPasswordBuffer();
}

else if (blue_tooth_mode == 1) {
    rotate90_degrees();
}
else if (blue_tooth_mode == 2) {
    Delay_ms(1500);
    // 可选: 显示开锁完成
    OLED_ShowString(3, 1, "Door Opened");
    Delay_ms(2000);
    OLED_Clear();
    blue_tooth_mode = 3;
}
else if (blue_tooth_mode == 3) {
    inPasswordMode = 0;
    passwordReceived = false;
    blue_tooth_mode = 0;
    OLED_Clear();
    displayMainMenu();
    // 重新开启 TIM2 中断
    TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
    USART_ITConfig(USART2, USART_IT_RXNE, ENABLE); // 使能USART2接收中断
    isFirstEnter=1;
}
```

图 68 STM32 串口通讯代码

## 4.5 门禁系统开关装置设计

步进电机的控制是通过 GPIO 引脚输出脉冲信号来实现的。步进电机在接收到每一个脉冲时都会按照预定的步距角旋转一定的角度，这使得步进电机具有精确的定位能力。步进电机的转动通常是通过控制电机的正反转以及脉冲的频率来完成的。通过设置 GPIO 口的输出电平，可以控制电机的转动方向和每次转动的角度。



步进电机逆时针转动程序示例如下：

```
//开门
void rotate90_degrees(void) {
    static uint16_t pulse_count = 0;

    // 如果已经顺时针转过90度，就不再顺时针转
    if (board_turn_flag_cw == 1) {
        return; // 已经执行过顺时针转动，直接返回
    }

    // 设置电机方向为正转（顺时针）
    GPIO_ResetBits(GPIOB, GPIO_Pin_7); // 方向信号输出低电平，电机正转

    // 输出一个脉冲
    GPIO_SetBits(GPIOB, GPIO_Pin_6); // 高电平
    delay_cnt(100); // 高电平持续时间
    GPIO_ResetBits(GPIOB, GPIO_Pin_6); // 低电平
    delay_cnt(100); // 低电平持续时间

    // 计数加1
    pulse_count++;

    // 每10个脉冲更新一次OLED显示
    if (pulse_count % 10 == 0) {
        OLED_ShowNum(3, 10, pulse_count, 3);
    }

    // 检查是否完成800个脉冲(90度)
    if (pulse_count >= 800) {
        pulse_count = 0; // 重置计数器
        Delay_ms(1000); // 延时1秒
        OLED_ShowString(3, 1, "Done");
        OLED_Clear();
        board_move_mode = 2; // 切换到完成状态
        rfid_mode = 2;
        blue_tooth_mode = 2;

        // 设置顺时针标志位为1，表示顺时针转动已完成
        board_turn_flag_cw = 1;
    }
}
```

图 69 步进电机控制代码

4.6 门禁系统拓展功能设计

为了实现门禁系统的智能化设计，保证在系统在只有控制基础功能的基础上，加入部分拓展功能，使得门禁系统功能更加完善。其功能设计如下：

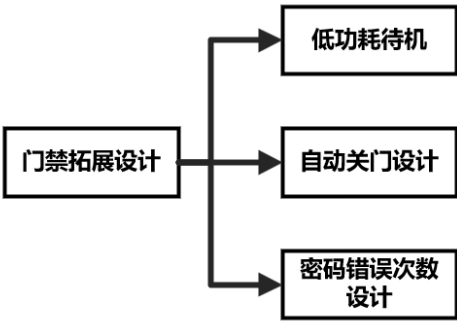


图 70 门禁拓展设计流程图

4.6.1 低功耗待机和自动关门设计

为了实现系统的低功耗设计，我们在系统中引入了智能的低功耗管理逻辑。通过单片机定时器的方式进行计时，系统能够在一定时间内监测到交互部分处于待机状态，并自动启用待机逻辑，如关闭 OLED 显示屏等，达到节能效果。

此外，结合系统的定时机制，我们实现了门禁的智能化管理。在系统监测到门未关闭的情况下，软件逻辑会自动控制关门装置，确保在设定的时间内完成自动关门操作，提升系统的智能化与节能性能。



```

void TIM2_IRQHandler(void) //
{
    if (TIM_GetITStatus(TIM2, TIM_IT_Update) == SET )
    {
        if(key_err==0&&stop_flag==0) //如果没有检测到按键按下
        count++; //定时计数

        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
    }
}

```

图 71 定时器代码

#### 4.6.2 密码错误次数限制设计

除了这部分功能外，为了增强系统的智能化，我们还在系统中加入了密码错误次数限制的机制。当用户输入密码错误超过一定次数时，系统会暂时禁止用户通过交互界面进行操作。此时，系统会通过定时器进行控制，同时在 OLED 上进行计时，直到设定的时间到达后，用户才能重新尝试输入密码或进行其他交互。

```

if(count>=5 && error_count <5){ //超过时间点 或者 按下按键
    count =0;
    clear_flag = 1;
    if(board_turn_flag_cw == 1 && !board_turn_flag_ccw && count>=5){ //如果超时发现门没关
        TIM_ITConfig(TIM2, TIM_IT_Update, DISABLE);
        USART_ITConfig(USART2, USART_IT_RXNE, DISABLE); // 使能USART2接收中断
        board_turn_flag_ccw = 1; //标志位置1 代表已经关门
        rotate90_degrees_reverse(); //顺时针
    }
}
else if(count >=3 && error_count>=5){
    error_count = 0;
    count = 0;
    displayMainMenu();
}
else {
    if(KeyNum) count = 0; //如果按下按键清零
}
if (error_count>=5) { // 当错误次数超过五次时
    // 根据按键执行不同操作
    if (isFirstEnter) {
        OLED_Clear(); // 清屏
        isFirstEnter = 0;
        inPasswordMode = 0;
        rfid_display_flag = 0;
        flag_scan = 0;
        flag_addcard = 0;
        flag_deletecard = 0;
        //标志位清空
        board_move_mode = 0; // 切换到完成状态
        rfid_mode = 0;
        blue_tooth_mode = 0;

        // 设置逆时针标志位为1，表示逆时针转动已完成
        TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
        USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);
    }
    OLED_ShowString(1, 1, "error times");
    OLED_ShowString(2, 1, "try again ");
    error_count_last_time = 60 - count;
    OLED_ShowNum(2, 1, error_count_last_time, 2);
    OLED_ShowString(2, 13, "s ");

    if(count >=3) OLED_Clear();
}
}

```

图 72 密码错误逻辑代码

#### 4.6.3 STM32 Flash 寄存器代码

对 Flash 寄存器配置，实现密码和 RFID 掉电不丢失，代码如下：

```

/**
 * @brief flash写入数据
 * @param add 32位flash地址
 * @param dat 16位数据
 * @retval 无
 */
void FLASH_W(u32 add,u8 dat1,u8 dat2,u8 dat3,u8 dat4)
{
    FLASH_Unlock(); //解锁FLASH编程擦除控制器
    FLASH_ClearFlag(FLASH_FLAG_BSY|FLASH_FLAG_EOP|FLASH_FLAG_PGERR|FLASH_FLAG_WRPERR); //清除标志位
    FLASH_ErasePage(add); //擦除指定地址页
    FLASH_ProgramHalfWord(add,dat1); //从指定页的addr地址开始写
    FLASH_ProgramHalfWord(add+2,dat2);
    FLASH_ProgramHalfWord(add+4,dat3);
    FLASH_ProgramHalfWord(add+6,dat4);
    FLASH_ClearFlag(FLASH_FLAG_BSY|FLASH_FLAG_EOP|FLASH_FLAG_PGERR|FLASH_FLAG_WRPERR); //清除标志位
    FLASH_Lock(); //锁定FLASH编程擦除控制器
}

/**
 * @brief FLASH读出数据
 * @param add 32位读出FLASH地址
 * @retval 16位数据
 */
u16 FLASH_R(u32 add)
{
    u16 a;
    a = *(u16*)add;
    return a;
}

/**
 * @brief 擦除指定FLASH地址页内的内容
 * @param add 32位FLASH地址
 * @retval 无
 */
void FLASH_Clear(u32 add)
{
    FLASH_Unlock(); //解锁FLASH编程擦除控制器
    FLASH_ClearFlag(FLASH_FLAG_BSY|FLASH_FLAG_EOP|FLASH_FLAG_PGERR|FLASH_FLAG_WRPERR); //清除标志位
    FLASH_ErasePage(add); //擦除指定地址页
    FLASH_ClearFlag(FLASH_FLAG_BSY|FLASH_FLAG_EOP|FLASH_FLAG_PGERR|FLASH_FLAG_WRPERR); //清除标志位
    FLASH_Lock();
}

// 写入密码到Flash
void SavePasswordToFlash(uint8_t* password) {
    // 假设密码为4位
    FLASH_W(FLASH_ADDR5, password[0], password[1], password[2], password[3]);
}

// 从Flash读取密码
void LoadPasswordFromFlash(uint8_t* password) {
    password[0] = FLASH_R(FLASH_ADDR5);
    password[1] = FLASH_R(FLASH_ADDR5 + 2);
    password[2] = FLASH_R(FLASH_ADDR5 + 4);
    password[3] = FLASH_R(FLASH_ADDR5 + 6);
}

// 在修改密码后，调用保存新密码到Flash
void UpdatePasswordInFlash(uint8_t* new_password) {
    SavePasswordToFlash(new_password);
}

```

图 73 Falsh 寄存器配置

## 第五章 智能门禁系统效果演示

### 5.1 交互功能演示

#### 5.1.1 按键输入密码

当单片机系统上电时，OLED 一级菜单如下图所示：

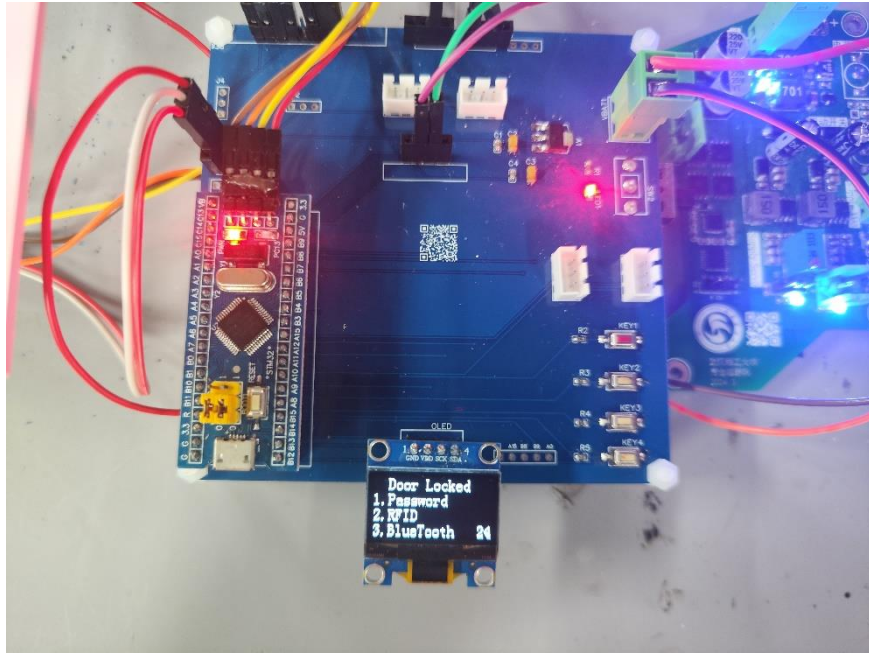


图 74 按键交互

当按下按 Key1 时，进入按键输入密码的二级菜单，如下图所示：

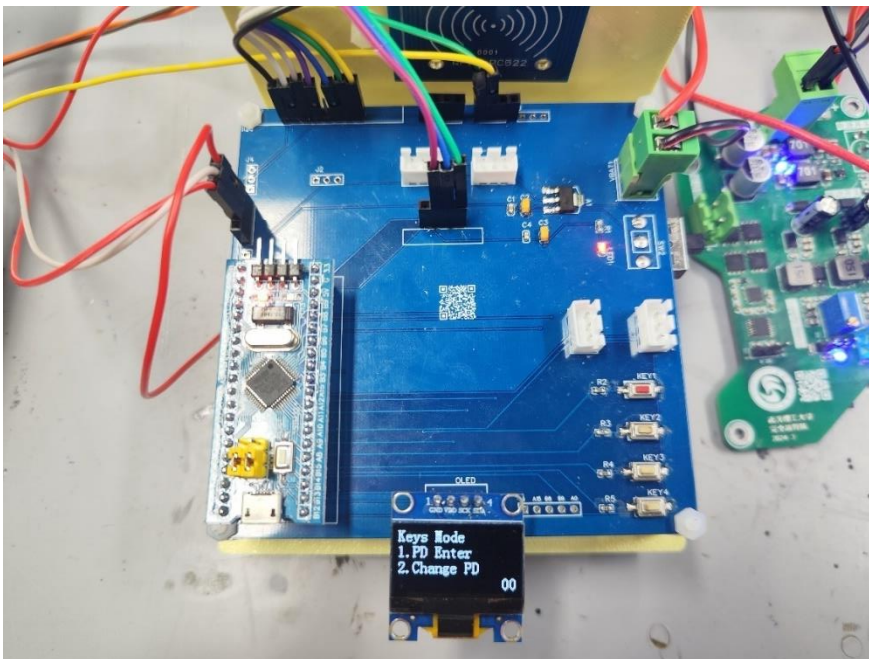


图 75 菜单选项

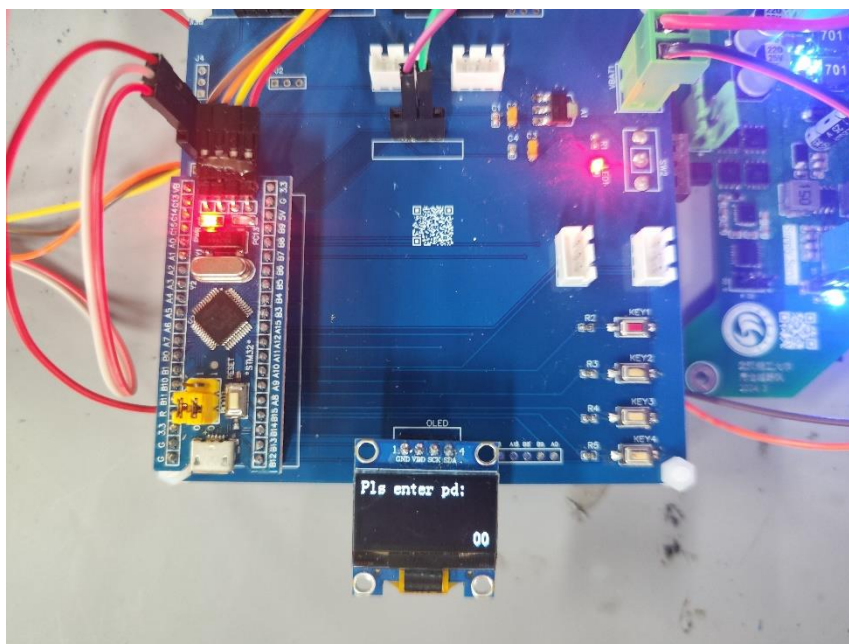


图 76 按键交互

当用户依次按下“1234”时，OLED 显示如下：

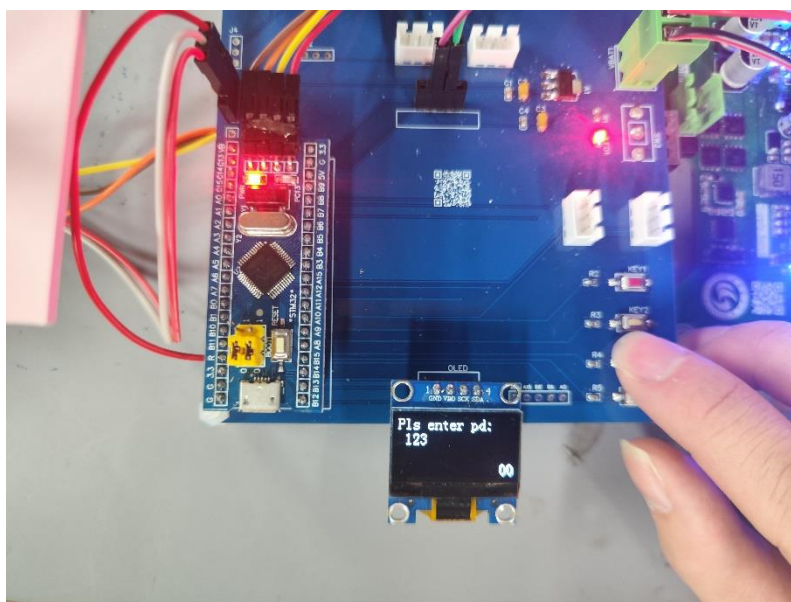


图 77 按键交互

当用户密码输入正确时，OLED 显示 Yes，其中数字部分为步进电机发送的脉冲数：



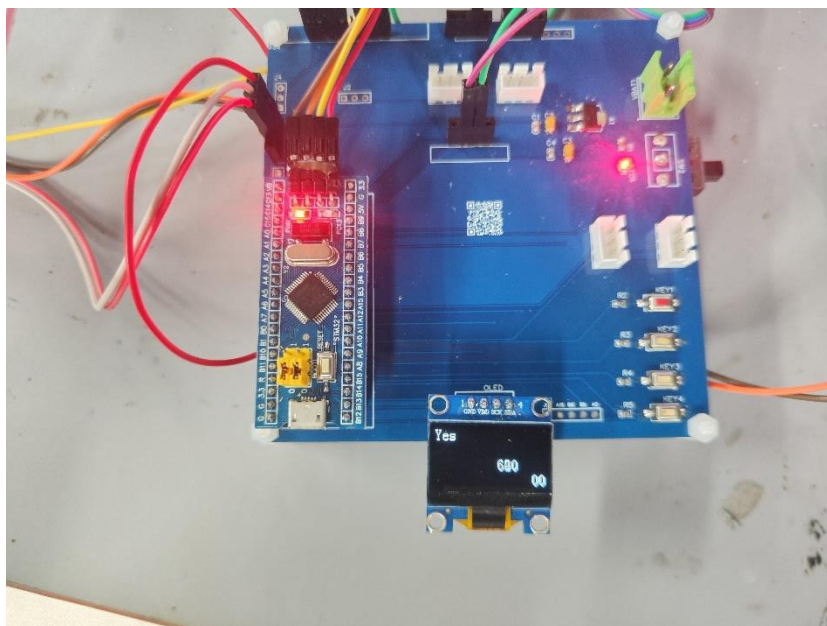


图 78 按键交互

当用户输入错误密码时，如下图所示：

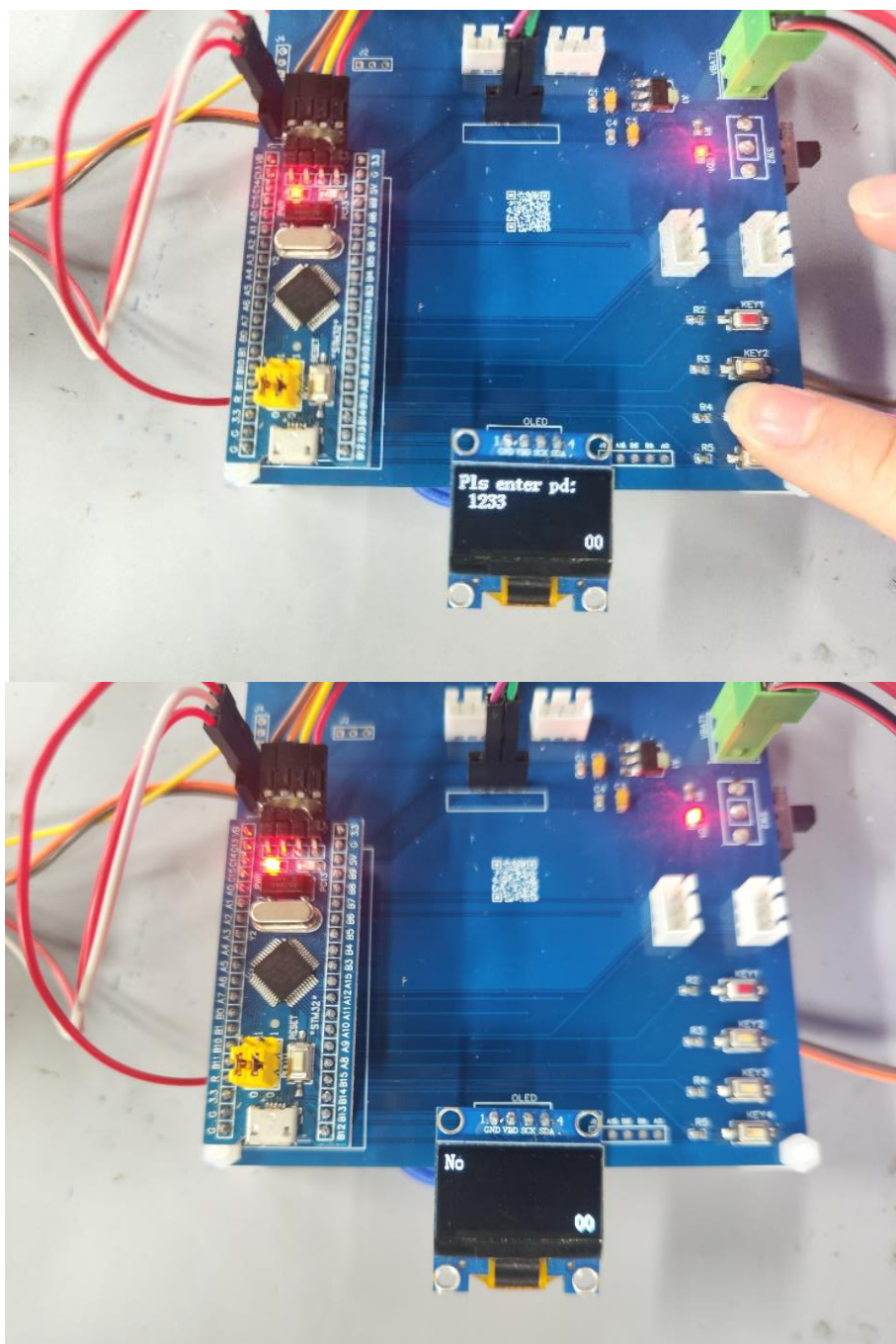


图 79 按键交互

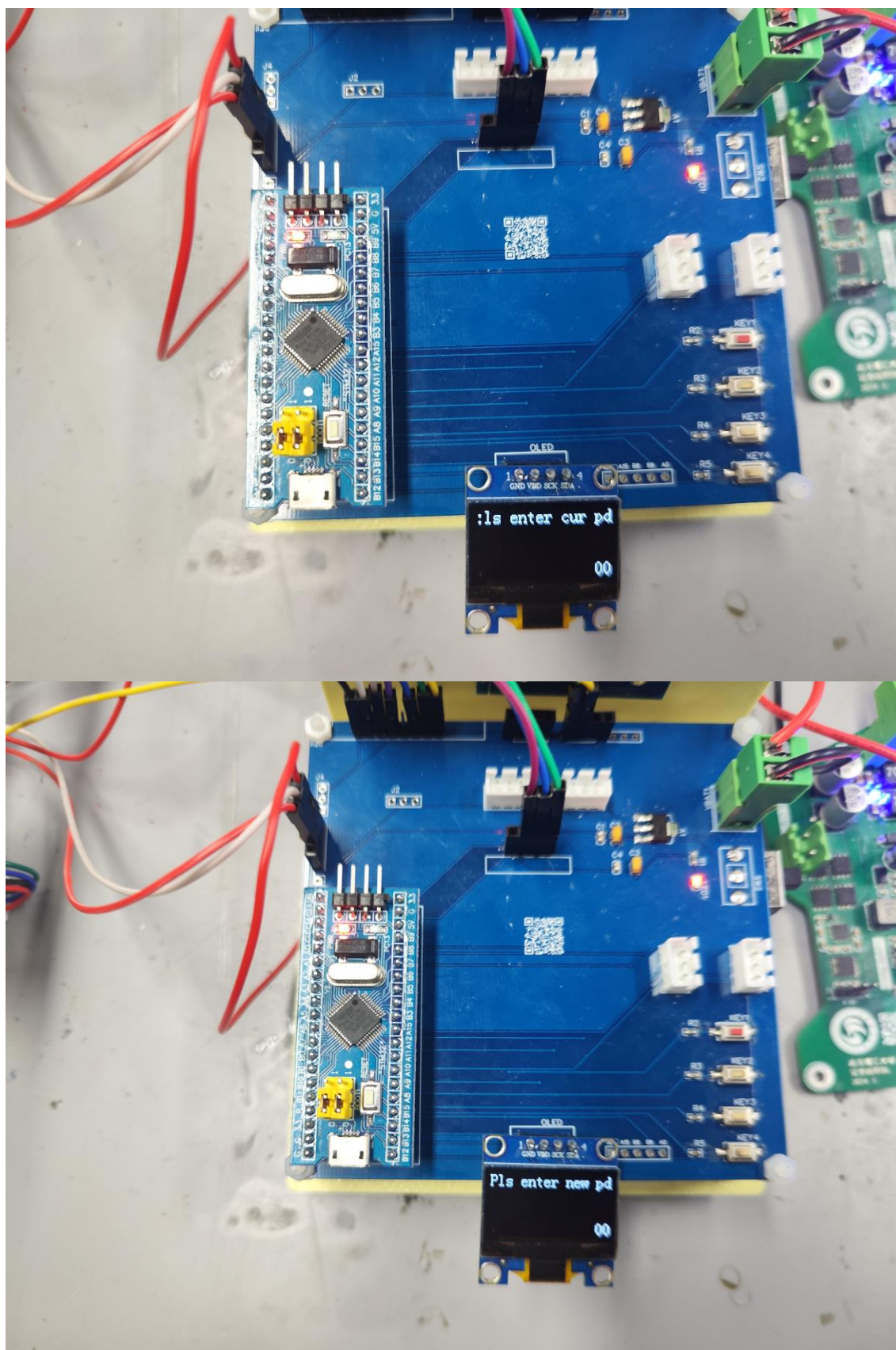


图 80 修改密码交互

## 5.1.2 RFID 控制

### 5.1.2.1 RFID 刷卡模式



当按下按键 key1 时，进入 RFID 刷卡的二级菜单，如下图所示：

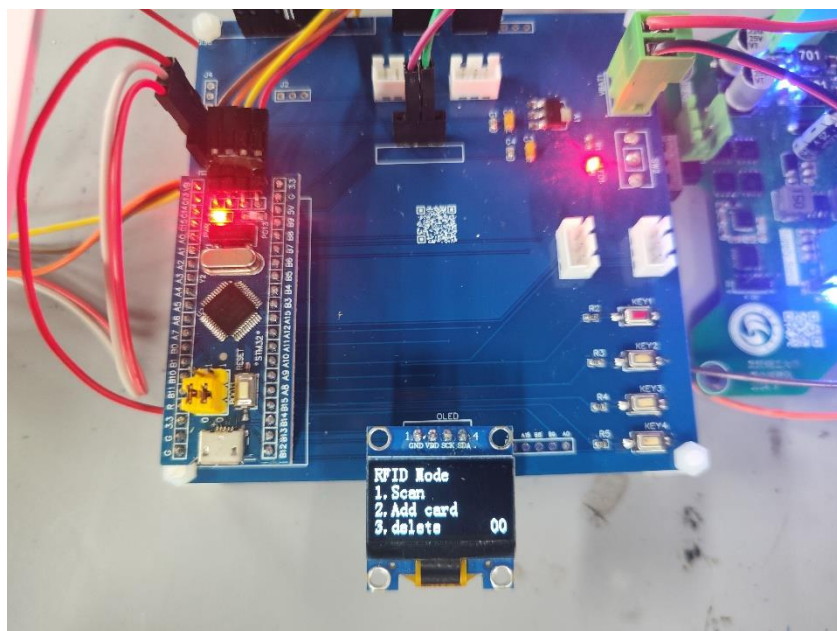


图 81 RFID 刷卡

#### 5.1.2.2 RFID 写卡模式

在 RFID 刷卡的二级菜单中，按下 Key1 后，系统将进入刷卡模式。当用户刷卡时，若所使用的 ID 卡已经存入 STM32 的 Flash 中，图中效果为使用手机 NFC 刷卡演示，如下图所示：

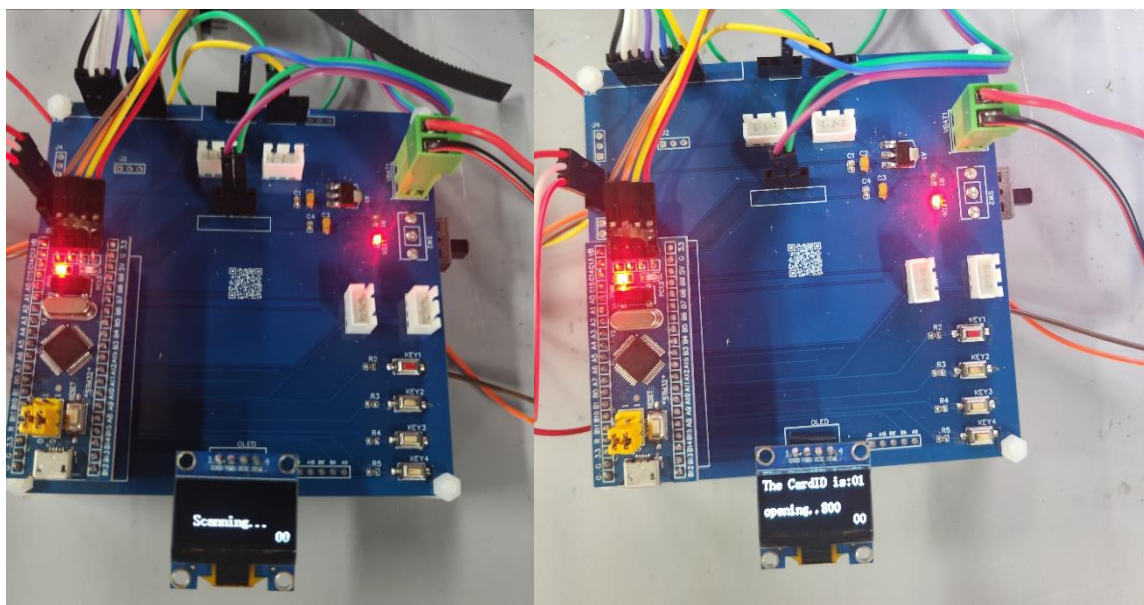


图 82 RFID 写卡

#### 5.1.2.3 RFID 删卡模式

重新回到 RFID 刷卡的二级菜单中，按下 Key3 后，系统将进入删卡模式。

根据前文可知，本系统对于 UID 的最大容量为 4。当进入删卡模式时界面如下所示：

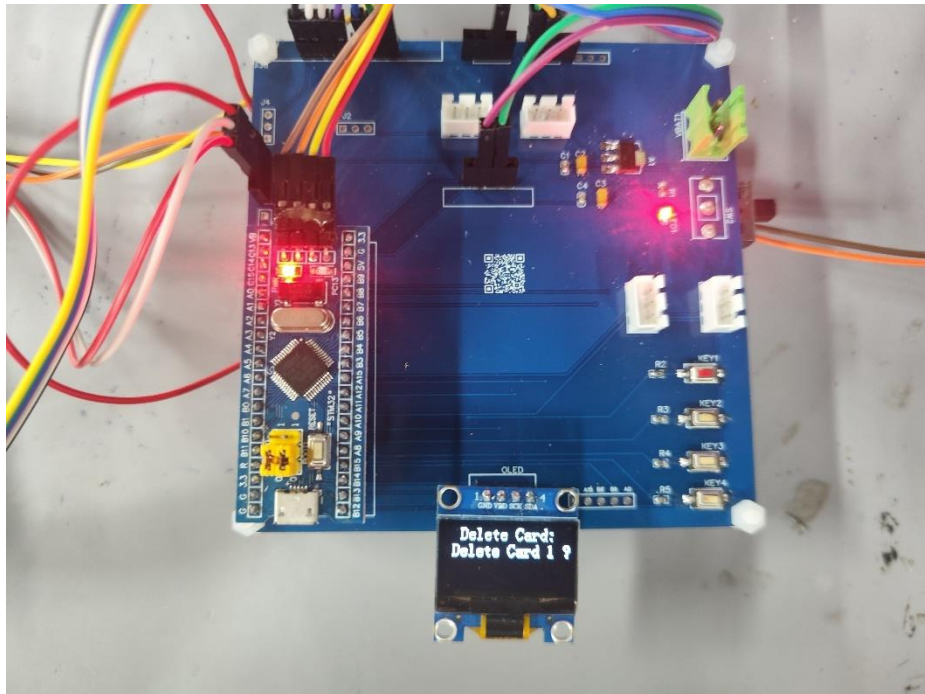


图 83 RFID 删卡

当按下两次 Key1 时，可以实现选中擦除 Flash 中目标 UID 的效果：

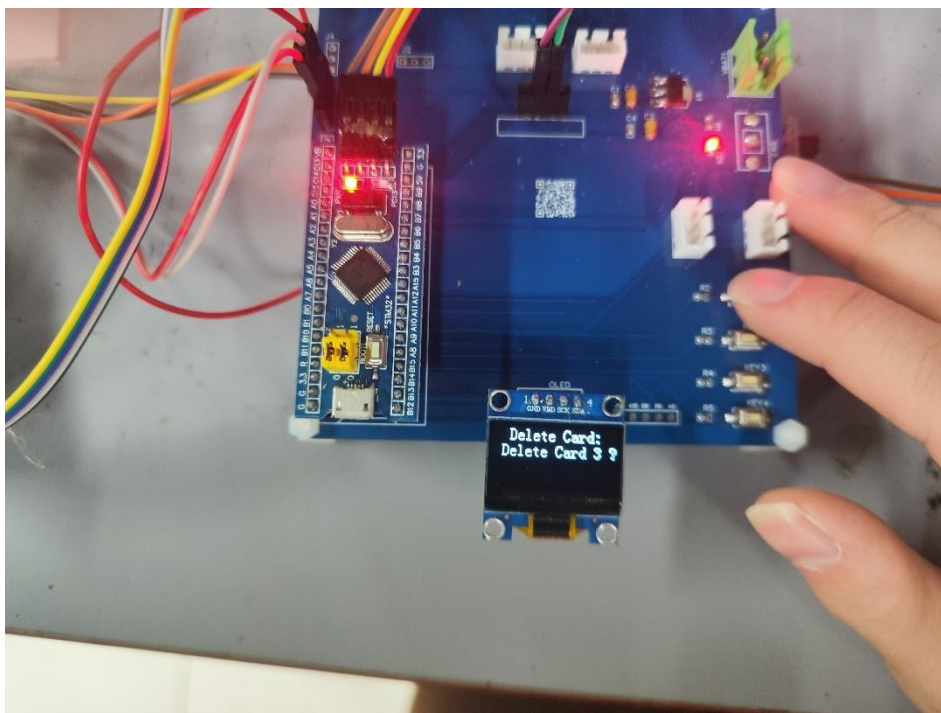


图 84 RFID 删卡

当再按下一次 Key2 时，效果如下：



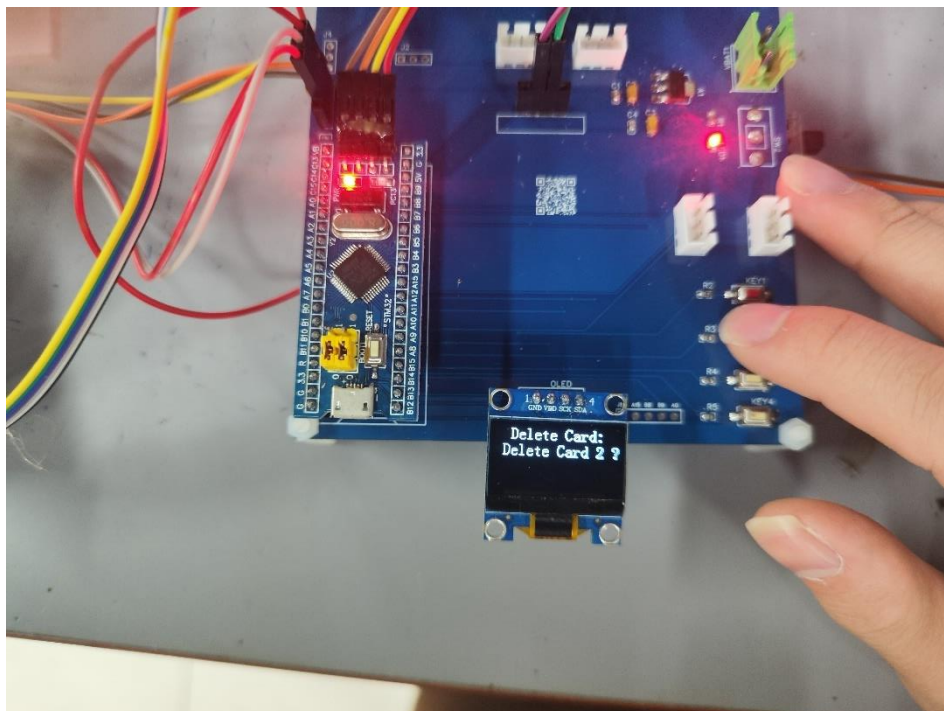


图 85 RFID 删卡

下图演示效果为将上文写入到 STM32 的 Flash 中的 UID 进行擦除，实现删卡效果，如下图所示：

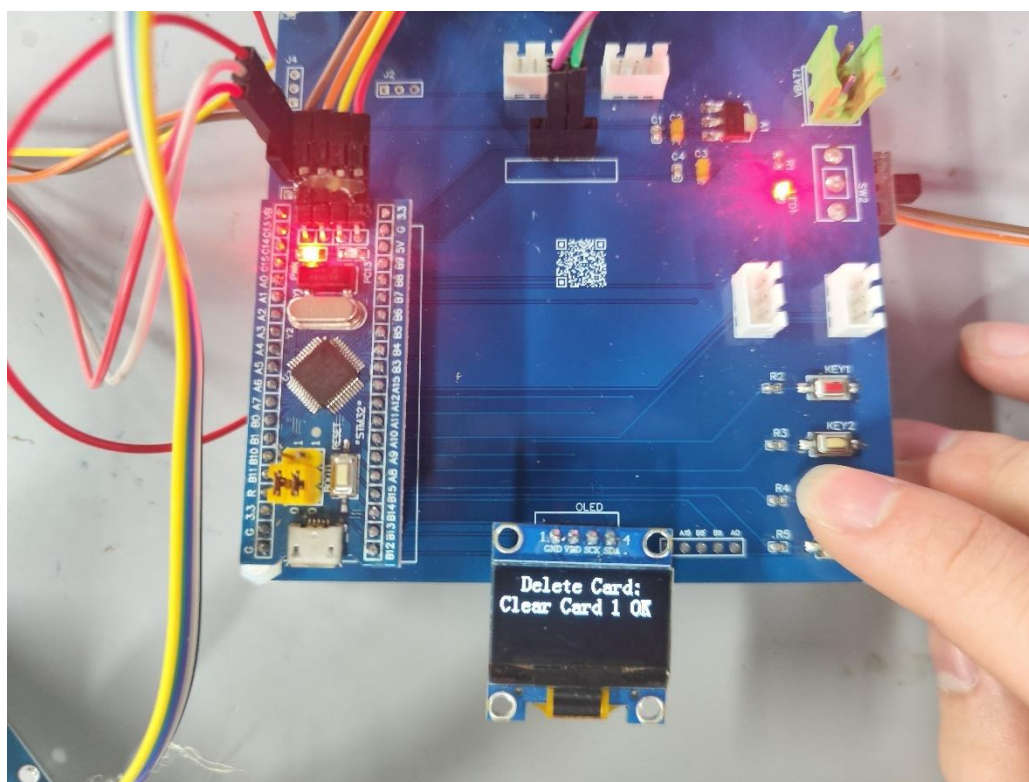


图 86 RFID 删卡

### 5.1.3 蓝牙通讯控制

#### 5.1.3.1 ESP32 与蓝牙串口助手 APP

当 ESP32 上电使用时，会打开系统的蓝牙，并一直在监视窗口打印“等待蓝牙连接”。此时打开手机蓝牙串口 APP 时，选中 ESP32 的蓝牙，进行连接；同时，在 OLED 一级菜单部分，按下 Key3，演示效果图下：

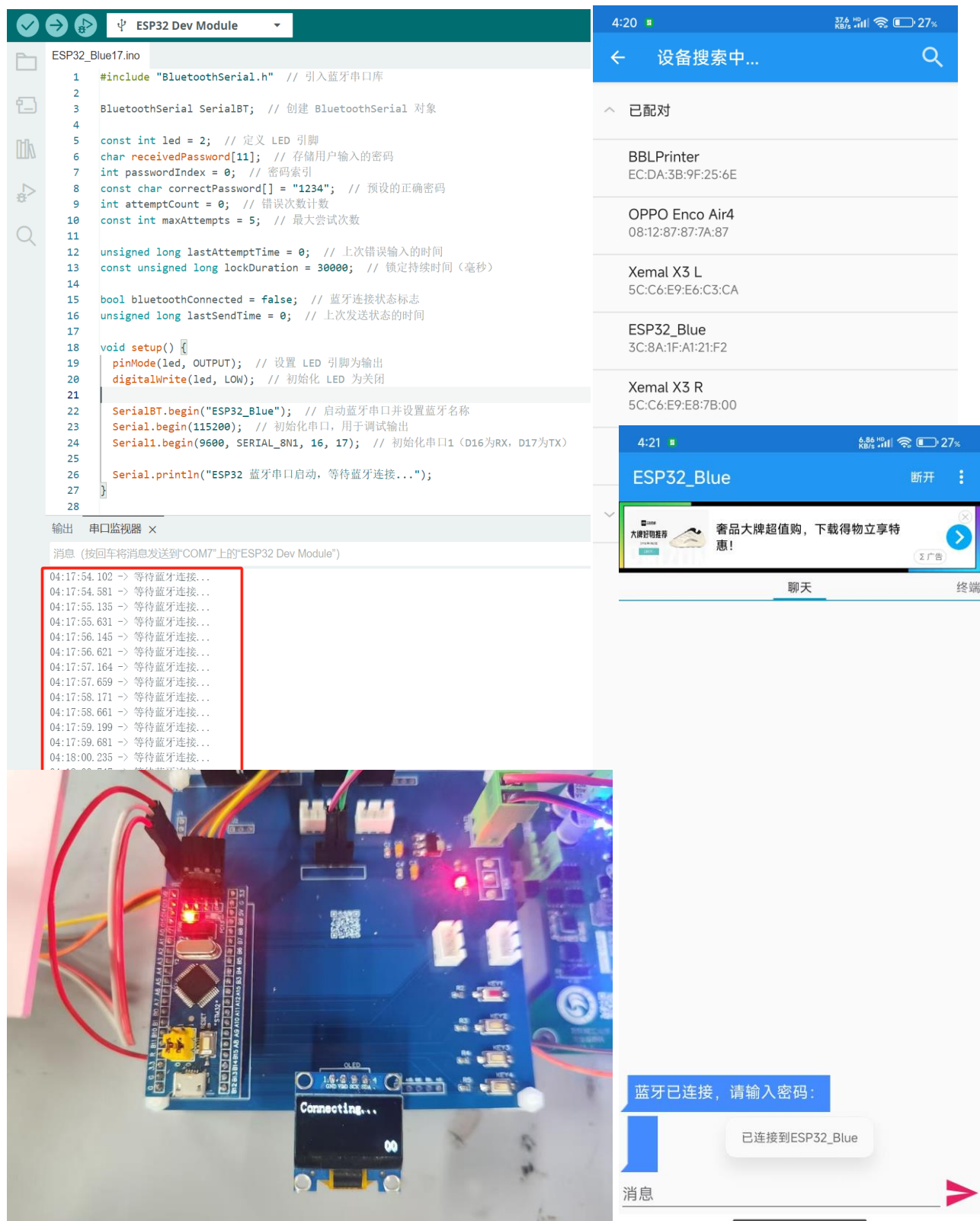


图 87 ESP32 通讯演示

当串口助手输入密码“1234”时，ESP32、串口助手和 OLED 界面如下：

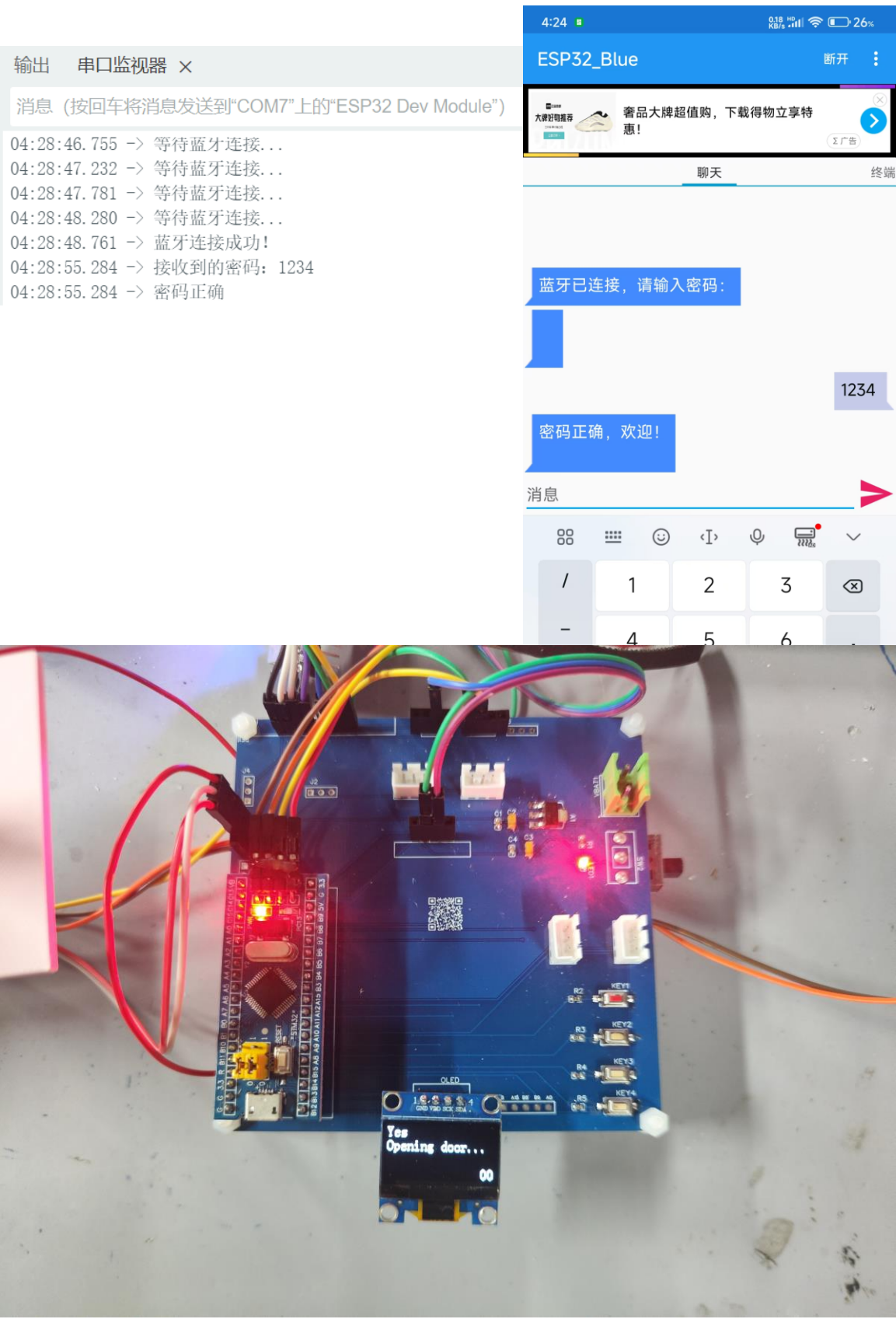


图 88 ESP32 通讯演示

当在串口助手输入密码“123”时，ESP32、串口助手和 OLED 界面如下：



04:39:47.135 -> 接收到的密码: 123  
04:39:47.135 -> 密码错误

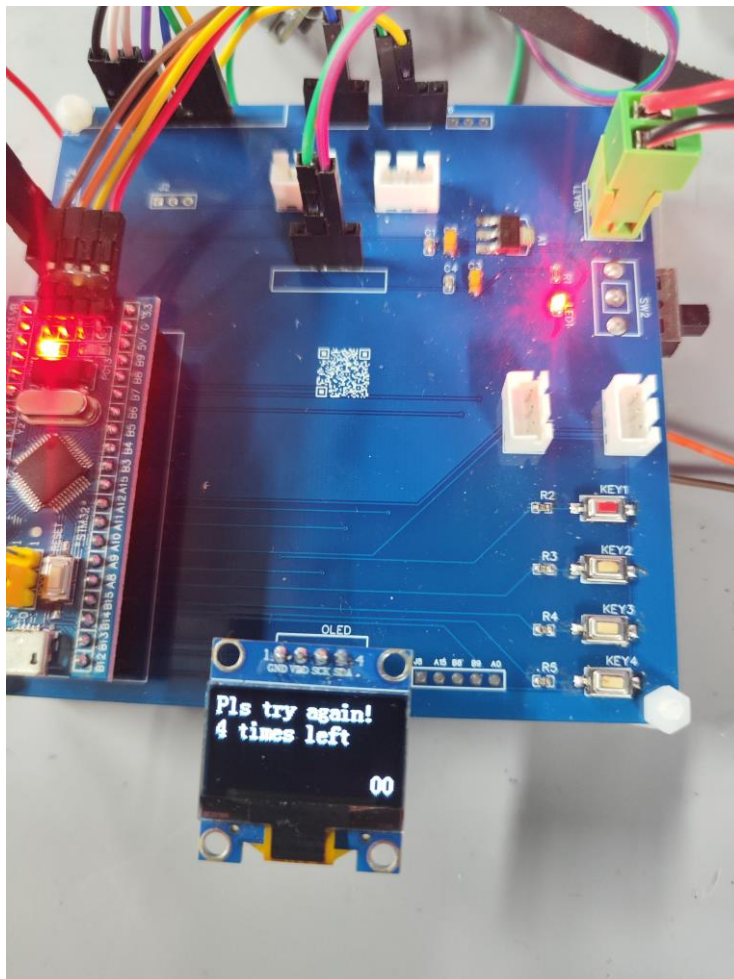


图 89 ESP32 通讯演示

#### 5.1.4 核心部分整体实物图

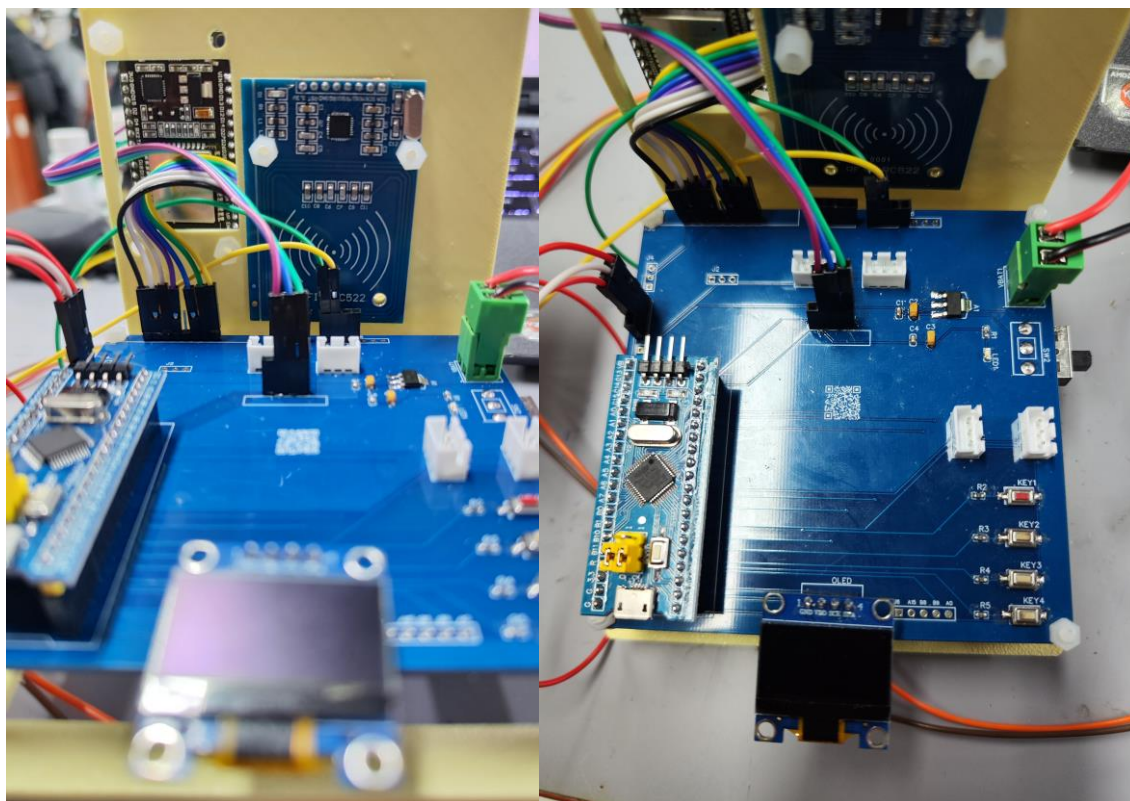


图 90 整体实物图

## 5.2 门禁驱动实物

为了模拟门禁驱动部分，根据采用的步进电机的转轴特点，设计了一个门框将其底部和步进电机的转轴部分结合起来，实现通过控制转轴转动实现门框开关的效果。

下图为使用 SolidWorks 进行的门框建模图：





图 91 门框建模图

下图为模型实物图：

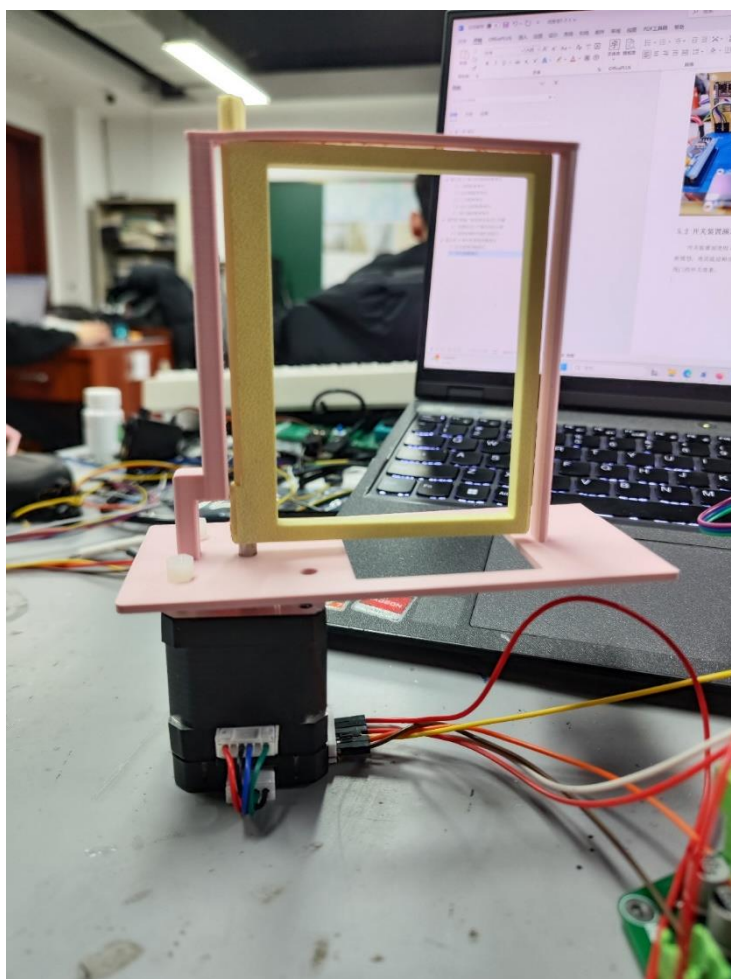


图 92 驱动实物图

### 5.3 电源模块实物

由于步进电机采用 12V 以上供电，而 STM32 系统板没有 12V 部分，因此采用被外部供电的方式。此处电源模块使用第十九届智能汽车竞赛设计的驱动板，其提供了 3.3V、5V、12V 三个供电端，能够直接为提供板载电压和电机驱动电压。

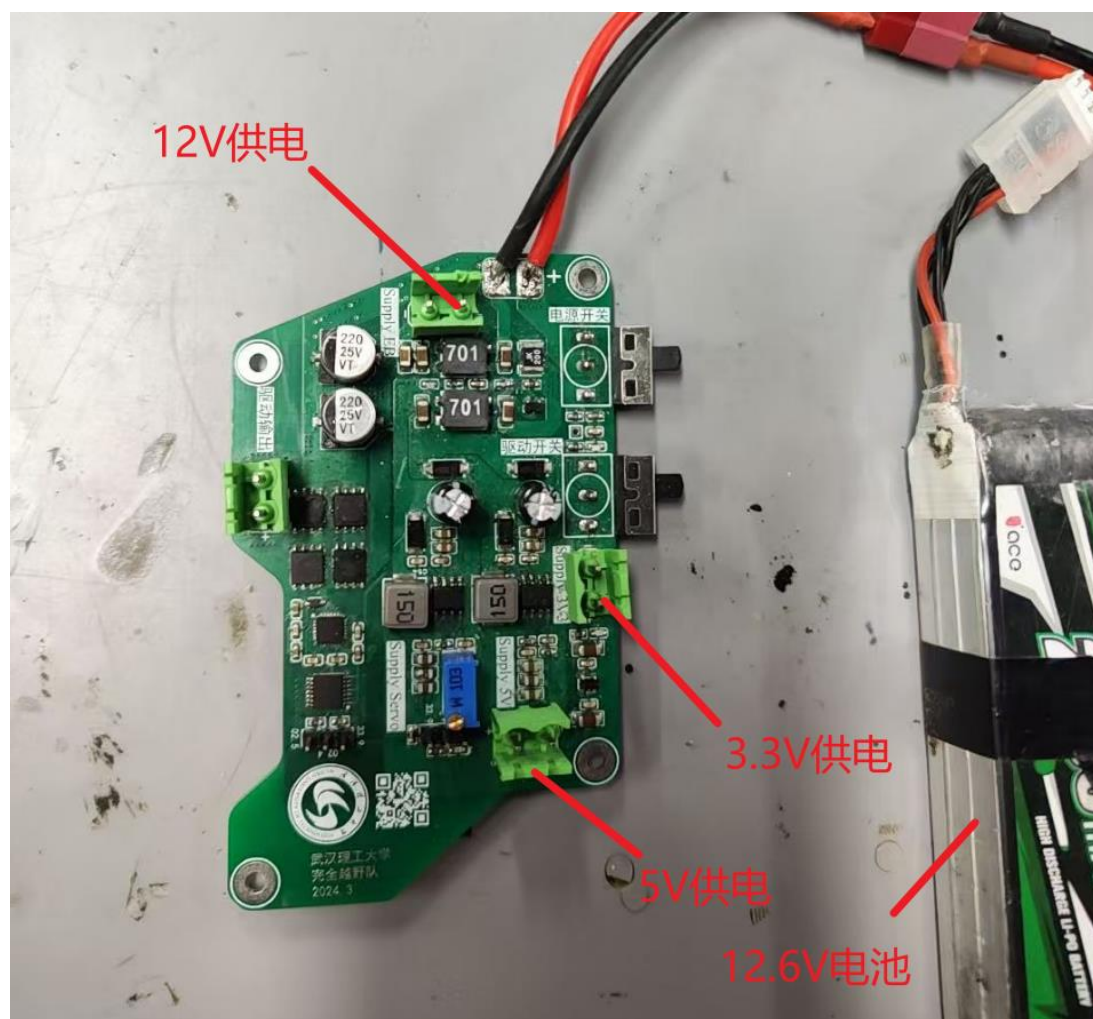


图 93 电源实物图

## 第六章 课程设计总结

本次课程设计任务圆满完成，分为选做和必做两部分。对于选做部分，内容相对简单，主要涉及的就是把单片机课程实验中的 PCB 进行复现，并将功能实现。主要围绕定时器、中断、串口通讯、LCD 显示、数码管显示和 ADC 采集这几部分。根据之前实验的经验，我只是稍微地进行了程序移植，并将全部程序通过按键逻辑集成到了一个程序中，使得演示更加方便。

对于选做部分，内容量相对较大，所做的智能门禁系统涉及按键交互、OLED 菜单编写、RFID 配置、蓝牙通讯连接和一些拓展的功能部分，除此之外，还有建模设计和 PCB 设计这几部分。工程量相对较大，本次选做部分的 PCB 除了外设模块之外均由自己设计，包括 STM32 拓展板、STM32 系统板和电源模块部分。对于软件编程部分，得益于现有开源的框架，如 RFID 识别，极大地便利了我的软件编写，但其中工作量最大的部分还是在交互界面方面的设计。为了在演示的过程中更加流畅，各部分衔接紧密不出现鸡肋的情况，我在交互方面花费了较大的功夫和力气。除了完成最基本的功能之外，我还考虑到了门的自动关闭、低功耗、密码多次错误的设计，如果没有这几部分，反而会显得系统只有单纯的控制，而没有智能化的体现。

写到这一部分，也是意味着本次单片机课设的接近尾声。自我感觉，这次单片机课设的工程量较大，时间相对紧张。从最开始 12 月初布置的任务开始，我便将以前设计的 32 主控板 PCB 发给嘉立创打板，也是在几天之内设计好了必做部分的 51PCB，但是由于第一次画板的失误，在焊接验证之后发现部分引脚出错，于是进行了第二次画板设计。非常幸运地，第二次制作的 PCB 没问题，在 12 月中旬时便完成了对于必做部分的设计；

而对于选做部分，我在 12 月中旬时便完成了最小系统板的焊接，也是非常有成就感地一次性焊接成功，并且板子经过验证没有问题。整个选做部分的完成度还算可以，但美中不足的是在最初设计的方案中，人脸识别部分并没有加入进去。虽然我在前期进行了大量的资料查询，尝试运行了如 FaceNet 等成熟的神经网络框架，但迫于后期时间紧张，没有能够来得及将功能加入，这对于热爱深度学习领域的我来说不觉得有点可惜。

但总的来说，自去年九月份后全部国赛结束后，我便再也没有接触过较大工程类的内容，本次单片机课设也是从某个角度上让我展现了往日我引以为傲的工程能力。

另外，这也是大四前的最后 3 个必修学分，我也是在本次课设的近乎每个方面做到了尽我所能的部分，包括文档、软硬件设计，希望能够为自己的专业必修学分画一个完美的计划，也希望自己未来能够不断进步，不忘初心，在此感谢单片机牟老师的悉心教导，感觉学长们的指导，也感谢自己的努力！

——2025.1.10.00:39

## 参考文献

- [1]高汉昆. 基于 51 单片机的智能停车场管理系统设计[J]. 电子技术, 2024, 53(08):114-115.
- [2]陈长顺. ADC0832 串行模数转换器及其应用[J]. 集成电路应用, 1992, (03):19-21. DOI:10.19339/j.issn.1674-2583.1992.03.006.
- [3]吴丙浩, 武祥磊, 曲明悬. 基于物联网与 RFID 的智能门禁系统设计[J]. 绿色建造与智能建筑, 2024, (04):131-133.
- [4]李永华. ESP32 物联网智能硬件开发实战[M]. 人民邮电出版社:202212. 305.
- [5]华凯楠, 刘伟. 智能门禁系统设计[J]. 科学技术创新, 2020, (26):114-115.
- [6]混分巨兽龙某某. 基于 STM32 的最小系统电路设计(STM32F103C8T6 为例) [EB/OL]. (2024-05-03) [2025-01-10].  
[https://blog.csdn.net/black\\_sneak/article/details/138352997](https://blog.csdn.net/black_sneak/article/details/138352997).
- [7]Love、伊卡洛斯. STM32F103+RFID-RC522 模块 实现简单读卡写卡 demo[EB/OL]. (2021-04-21) [2025-01-10].  
[https://blog.csdn.net/Ikaros\\_521/article/details/115958888](https://blog.csdn.net/Ikaros_521/article/details/115958888).

## 附录一 材料清单

序号	名称	型号	数量	单价（元）
7	电阻	200Ω	100	2.94
8	电阻	9.1kΩ	100	2.94
9	电阻	10kΩ	100	2.94
10	电阻	20kΩ	100	2.94
11	电阻	30kΩ	100	2.94
12	电阻	43kΩ	100	2.94
13	电阻	1kΩ	100	2.94
14	电阻	300Ω	100	2.94
15	电阻	75kΩ	100	2.94
16	电容	0.1uF	100	2.94
17	电容	22pF	100	5.72
18	电容	33pF	100	4.91
19	蓝牙模块	HC-06	2	21.80
20	石英晶振	11.0592M	1	3.27
21	有源晶振	32.768K	1	2.07
22	CH340N	SOP-8	1	1.60
23	无源晶振	3215	1	4.46
24	拨动开关	SS12D06	6	5.22
25	无源晶振	YSX530GA	1	4.23
26	舵机	MG945	1	19.80
27	步进电机	28HYJ48	1	8.80
28	Type-C	16P	10	3.30
29	AMS1117	SOT223	1	3.19
30	按键	3.6*3.6*2.5	5	6.30
31	3D 打印耗材	PLA	1	60.03

32	单片机	STM32F103C8T6	1	16.72
33	MCU	STM32F103C8T6	1	3.20
34	震动传感器模块	SW-420	1	2.78
35	矩阵键盘	4*4	2	6.30
36	指纹模块	AS608	1	36.49
37	语音模块	ASR-PRO	1	27.80
38	OLED 显示屏	0.96 寸	1	9.90
合计				259.49