

**实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验课程： | 单片机原理与应用综合实验 | | | | | |
| 实验编号： | 选题四 | | | | | |
| 实验名称： | 智能电梯控制系统 | | | | | |
| 实验人员： | 学号 | 22111304042 | | | 22111304046 | |
| 姓名 | 张俊杰 | | | 贝新宇 | |
| 班级 | 22级物联网工程1班 | | | | |
| 实验日期： | 2024.6.17 | | | | | |
| 实验室： | 1号楼203 | | | | | |
|  |  | | | | | |
| 实验评价： |  | | | | | |
| 实验成绩： | |  | 评价日期： | |  |
|  | 指导教师： | |  | | | |

# 设计目标

（1）使用4x4的矩阵键盘模拟电梯箱内的楼层选择按钮。当按钮按下是，电梯控制系统记录对应层数（建筑共有9层）。

（2）使用步进电机驱动模块驱动步进电机转动，顺时钟转动为电梯上升，逆时针转动标识电梯下降。电机转动一圈表示电梯升降一个楼层。

（3）通过显示模块显示楼层、人数。

（4）当电梯空闲时（3秒内矩阵键盘没有键按下），电梯停留到1楼。

（5）当电梯启动前和停止后，使用LED灯和蜂鸣器实现1s的声光电提示。

（6）设置电梯具有互锁功能（运行时，电梯开不了；门开状态，不能运行），通过输出模块可视化当前电梯状态。

（7）设置电梯按钮具有记忆功能。电梯在运行是能及时接收按键的呼叫信号，以先方向后距离的原则进行判断，自行优化运行路径，运行过程中具有不可逆响应功能，任何方向呼叫均无效，应符合实际运行模式。

（8）使用点阵屏，在系统开始前流动显示“电梯 I ♡ AHNU”字样，提示该电梯系统可以正常运行；在电梯上升和下降时，点阵屏会显示对应的流动箭头，提示当前电梯运行方向。

（9）使用蜂鸣器，来限制电梯人数，当人数超过5人时，蜂鸣器发生报警提示，直到人数在5人即以下。

（10）使用串口可视化输出，通过串口通信，记忆已经按下的楼层数字，显示电梯门当前的状态，显示“door open”和“door close”字样。

# 设计方案

1. **设计封面（实物）开机后**，依次显示小组信息。
2. **按键控制电梯门**，通过按键中断控制门的关闭以及开启，初始化按键中断的端口时需要将端口设置为输入模式并且拉高电平。
3. **实现矩阵按键的楼层选择**，通过扫描检测的方式，检测到具体是哪个按键被选择，并且记忆哪些按键被选择。
4. **步进电机模拟电梯运行**，根据电生磁，对步进电机的顺时针转动以及逆时针转动给出对应的电压，控制合理的转速。当电梯停止时，LED灯亮，表示到达某层。
5. **数码管显示楼层**，根据步进电机的转动，在完成一周的转动后对数码管的显示做出修改，电机转动一圈表示电梯升降一个楼层；**数码管显示人数，**通过按键显示进出人数数量。
6. **记忆以及路径优化功能的实现**，在楼层选择时，会将对应下标的数组值置1，使用桶排序，从后往前排序，按下多个楼层后，数组中对应下标的数组值都是1，都会被识别，实现记忆，同时由于数组是从后往前遍历，优先会去高楼层，然后一步一步往下降，这里满足先方向后距离的原则。
7. **电梯互锁功能**，由于是通过按键中断控制门的关闭以及开启，所以当电梯进入运行状态时，在这之前将用于按键中断的端口的电位设置为低电平，按键将不起作用。
8. **串口通信**，将记忆楼层以及门的状态都通过串口通信显示在上位机上，同时通过重新映射printf函数实现较为方便的串口输出。
9. **自定义功能，**当人数超过5人时，蜂鸣器自动报警，发出声音；同时每次电梯开机时会在点阵屏中滚动显示“I love(图片) ahnu ”；而对于电梯的升降使用点阵屏，上升时滚动显示上箭头标志，下降时滚动显示下箭头标志。

# 原理分析

1. 初始化硬件设备，包括LCD显示屏、串口通信及GPIO端口。
2. 设置外部中断，用于检测电梯内人数变化和门的操作。
3. 使用LED矩阵显示电梯的移动方向（向上或向下）。
4. 通过主循环不断检测楼层请求，并根据请求控制电梯的运行和门的开关。

**硬件初始化：**

程序开始时，进行了一系列硬件初始化，包括延时、串口通信、LCD显示屏及GPIO端口配置。

1. 延时初始化：delay\_init();

2. 串口初始化：uart\_init(9600);

3. LCD初始化：LCD\_Init();

4. GPIO端口配置：

|  |
| --- |
| RCC\_APB2ENR|= (1ul<<5)|(1ul<<2)|(1ul<<0)|(1ul<<4)|(1ul<<8)|(1ul<<3)|(1ul<<8);  GPIOC->CRL=0x11111111;  GPIOA->CRL=0x11111111;  GPIOD->CRL=0x11111881;  GPIOB->CRL=0x11111111;  GPIOG->CRL=0x11111111; |

**显示初始页面：**

在初始化后，使用LCD屏幕，显示一些文本和信息：

|  |
| --- |
| while(stall) {  for(i=0;i<14;i++) Chinese\_Show\_one(10+i\*16,40,i,16,0);  ...  stall--;  delay\_ms(1000);  } |

`stall`变量用于控制显示初始页面的时间。

**中断初始化：**

程序设置了多个中断，主要用于检测电梯内的人数变化和门的操作。

1. PD1和PD2中断初始化：

|  |
| --- |
| void PD\_1\_2\_\_intr\_init() {  RCC\_APB2ENR |= (1ul<<5) | (1ul<<0);  GPIOD->ODR |= (1ul<<1);  GPIOD->ODR |= (1ul<<2);  AFIO\_EXTICR1 |= (3ul<<4);  AFIO\_EXTICR1 &= ~(12<<4);  AFIO\_EXTICR1 |= (3ul<<8);  AFIO\_EXTICR1 &= ~(12<<8);  EXTI\_IMR |= (1ul<<1);  EXTI\_FTSR |= (1ul<<1);  EXTI\_IMR |= (1ul<<2);  EXTI\_FTSR |= (1ul<<2);  NVIC\_ISR0 |= (1ul<<7);  NVIC\_ISR0 |= (1ul<<8);  } |

2. PB0中断初始化：

|  |
| --- |
| void PB\_0\_intr\_init() {  RCC\_APB2ENR |= (1ul<<3);  GPIOB->ODR |= (1ul<<0);  AFIO\_EXTICR1 |= (1ul<<0);  AFIO\_EXTICR1 &= ~(14<<0);  EXTI\_IMR |= (1ul<<0);  EXTI\_FTSR |= (1ul<<0);  NVIC\_ISR0 |= (1ul<<6);  } |

**中断处理函数**

1. 人数增加中断处理（EXTI1\_IRQHandler）：

|  |
| --- |
| void EXTI1\_IRQHandler() {  delay\_ms(1000);  people\_num = people\_num + 1;  if(people\_num >= 5) {  GPIOB->ODR &= ~(1ul<<1);  } else {  GPIOB->ODR |= (1ul<<1);  }  GPIOG->ODR = people\_a[people\_num];  EXTI\_PR|=(1ul<<1);  NVIC\_PR|=(1ul<<7);  } |

2. 门操作中断处理（EXTI2\_IRQHandler）：

|  |
| --- |
| void EXTI2\_IRQHandler() {  delay\_ms(100);  door = 0;  door\_opentime = 0;  GPIOD->ODR &= ~(1ul<<0);  EXTI\_PR|=(1ul<<2);  NVIC\_PR|=(1ul<<8);  delay\_ms(1000);  } |

3. 人数减少中断处理（EXTI0\_IRQHandler）：

|  |
| --- |
| void EXTI0\_IRQHandler() {  delay\_ms(1000);  people\_num = people\_num - 1;  if(people\_num >= 5) {  GPIOB->ODR &= ~(1ul<<1);  } else {  GPIOB->ODR |= (1ul<<1);  }  GPIOG->ODR = people\_a[people\_num];  EXTI\_PR|=(1ul<<0);  NVIC\_PR|=(1ul<<6);  } |

**主循环**

程序进入主循环，主要实现以下功能：

1. **检测楼层请求**：

使用键盘输入检测用户的楼层请求，并更新目标楼层floor\_tag。

|  |
| --- |
| for (i = 0;i<20;i++) {  floor\_tag = 0;  x = 0;  y = 0;  GPIOC->ODR = 0x0f;  if (GPIOC->IDR&(1ul<<4)) x = 4;  ...  floor\_reg[floor\_tag] = 1;  delay\_ms(1000);  printf("%d\n", floor\_tag);  } |

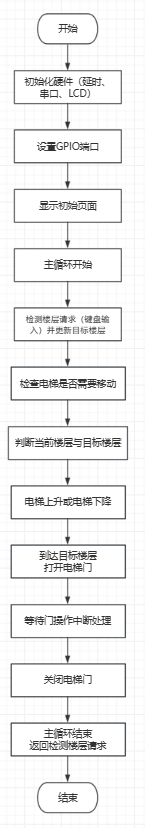
2. **电梯运行及门操作**：

根据请求的目标楼层floor\_reg，控制电梯的移动及门的开关。

|  |
| --- |
| for (i = 10;i>=1;i--) {  if (floor\_reg[i] == 1) {  if (floor\_cur == i) continue;  printf("door open\n");  door = 1;  door\_opentime = 0;  GPIOD->ODR |= (1ul<<0);  while(1) {  if (door == 0) break;  }  printf("door close\n");  GPIOD->ODR &= ~(1ul<<0);  ...  if (floor\_cur<i) {  elevatorup();  while(1) {  for (tmp = 0;tmp<512;tmp++) {  GPIOD->ODR = 0x10;  delay\_ms(10);  GPIOD->ODR = 0x20;  delay\_ms(10);  ...  }  cir = cir + 1;  GPIOA->ODR = floor\_a[floor\_cur+cir];  if (cir==i-floor\_cur) break;  }  } else {  elevatordown();  while(1) {  for (tmp = 0;tmp<512;tmp++) {  GPIOD->ODR = 0x80;  delay\_ms(10);  GPIOD->ODR = 0x10;  delay\_ms(10);  ...  }  cir = cir + 1;  GPIOA->ODR = floor\_a[floor\_cur-cir];  if (cir==floor\_cur-i) break;  }  }  floor\_reg[i] = 0;  floor\_cur = i;  GPIOD->ODR |= (1ul<<3);  delay\_ms(6000);  GPIOD->ODR &= ~(1ul<<3);  printf("door open\n");  door = 1;  door\_opentime = 0;  GPIOD->ODR |= (1ul<<0);  while(1) {  if (door == 0) break;  door\_opentime = door\_opentime + 1;  delay\_ms(1000);  if (door\_opentime == 6) break;  }  printf("door close\n");  GPIOD->ODR &= ~(1ul<<0);  floor\_tag = 0;  }  } |

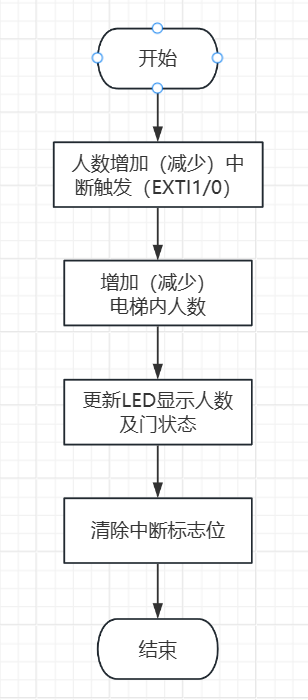
# 程序流程

**主程序：**

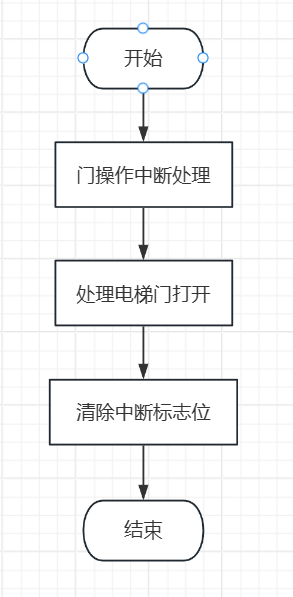


**子程序：**

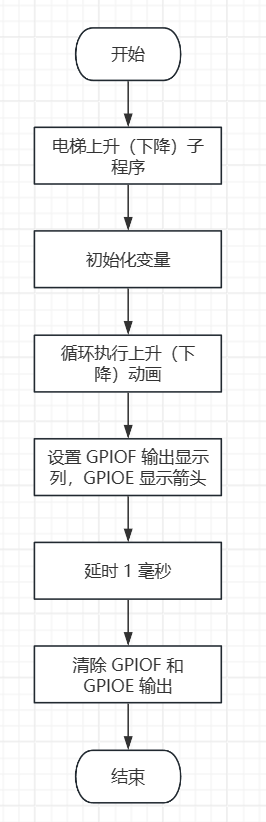
**人数增加（减少）中断处理**



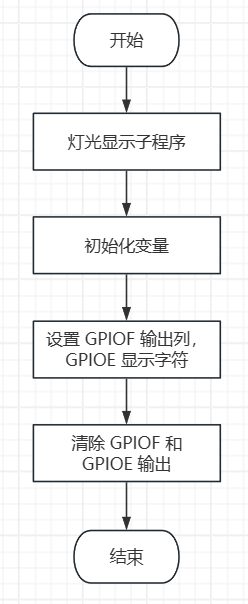
**门操作中断处理**



**电梯上升（下降）子程序**



**灯光显示子程序**



# 验证方案

**1. 基本功能测试**

**1.1 电梯上升和下降功能**

**步骤：**

在电梯处于一楼时，按下上升按钮，观察电梯是否上升到二楼并停下。

在电梯处于顶楼时，按下下降按钮，观察电梯是否下降到一楼并停下。

**预期结果：**

电梯在按下按钮后，能够按照指示准确移动到相应楼层并停下。

**1.2 楼层显示功能**

**步骤：**在电梯移动过程中，观察电梯内外的楼层显示屏。

**预期结果**：楼层显示屏应正确显示电梯当前所在的楼层。

**1.3 乘客人数显示功能**

**步骤：**在电梯内加入不同数量的乘客，观察乘客人数显示屏。

**预期结果：**显示屏应正确显示电梯内的乘客人数。

**2. 安全功能测试**

**2.1 超载报警功能**

**步骤：**在电梯内加入超过最大载荷的乘客，观察电梯是否发出超载报警。

**预期结果：**电梯应发出报警，并且无法启动。

**2.2 开关门功能**

**步骤：**

按下开门按钮，观察电梯门是否打开。

按下关门按钮，观察电梯门是否关闭。

**预期结果：**电梯门应在按下相应按钮后准确执行开门和关门操作。

**3. 界面显示测试**

**3.1 初始界面显示**

**步骤：**重启电梯控制系统，观察初始界面显示内容。

**预期结果：**初始界面应显示系统信息和相关数据。

**3.2 动态信息显示**

**步骤：**在电梯上升、下降和停留时，观察显示屏是否正确更新相关信息（如楼层、乘客人数等）。

**预期结果：**显示屏应实时更新并显示当前电梯状态和信息。

**4. 故障处理测试**

**4.1 电梯停电故障**

**步骤：**模拟电梯运行中突然停电，观察电梯的应急反应。

**预期结果：**电梯应立即停止，并在恢复电源后正常运行。

**4.2 电梯卡层故障**

**步骤：**模拟电梯在楼层间卡住的情况，观察系统的反应。

**预期结果：**系统应发出故障报警，并在维修后恢复正常运行。

# 存在的问题

**1. 功能性问题**

**1.1 楼层识别错误**

电梯在移动过程中无法正确识别楼层，导致停靠楼层与目标楼层不一致。

**可能原因：**编码逻辑错误，导致楼层数值计算错误。

**解决方案：**复查编码逻辑，确保楼层识别算法正确。

**1.2 楼层显示不准确**

电梯内外的楼层数码管显示的楼层信息与实际楼层不符。

**可能原因：**数码管连接不良或损坏。

**解决方案：**检查连接，确保硬件正常工作。

**2. 安全性问题**

**2.1 超载检测不灵敏**

电梯在超载情况下无法及时发出报警或启动保护机制。

**可能原因：**控制逻辑错误，未能正确处理超载信号。

**解决方案：**调试逻辑，确保超载信号能够及时触发报警和保护机制。

**2.2 门控制故障**

电梯门无法正常打开或关闭，可能卡住或误操作。

**可能原因：**控制逻辑错误，导致门控操作异常。

**解决方案：**调试控制逻辑，确保门控操作准确可靠。

**3. 用户界面问题**

**3.1 显示屏乱码或显示不全**

电梯显示屏出现乱码或部分信息显示不全。

**可能原因：**显示屏硬件故障或连接不良。

**解决方案：**更换或修理显示屏，确保硬件连接正常。

3.2 按钮响应不灵敏

电梯内外的按钮按下后反应迟钝或无响应。

**可能原因：**按钮硬件故障或接触不良。

**解决方案：**更换或维修按钮，确保硬件正常。

**4. 系统稳定性问题**

**4.1 运行过程中卡顿或停顿**

电梯在运行过程中出现卡顿或停顿现象，无法顺畅运行。

**可能原因：**电机或控制系统负载过大。软件算法效率低下，导致系统响应延迟。

**解决方案：**检查并优化电机及控制系统的负载。优化软件算法，提高系统运行效率。

# 同组同学分工

|  |  |
| --- | --- |
| 张俊杰 | 贝新宇 |
| 代码编写  PPT制作  硬件连接  视频拍摄 | 代码编写  Word编辑  硬件连接  视频剪辑 |

# 总结与体会

在本次电梯控制系统项目中，我们从项目的设计、开发、测试到最终的实现，经历了完整的流程。通过共同努力，我们达成了以下成果：

**系统设计与实现：**设计并实现了电梯控制系统的硬件部分，包括传感器、电机、控制电路等。编写了系统的软件部分，实现了电梯的各项功能，如楼层识别、显示屏控制、开关门控制等。

**测试与调试：**设计并执行了测试方案，对系统进行了功能测试、安全性测试和稳定性测试。通过多次调试，修复了系统中存在的各种问题，确保系统的可靠性和稳定性。

**项目文档与汇报：**编写了详细的项目文档，记录了项目的设计、实现、测试等各个环节的内容。制作了项目展示材料，并进行了项目汇报与答辩。

通过本次项目的实践，我们不仅掌握了电梯控制系统的设计与实现方法，还提升了我们的团队合作能力、问题解决能力和系统化思维能力。在未来的学习和工作中，我们将继续保持这种积极的学习态度和合作精神，不断提升自己的专业能力，为实现更高的目标而努力。

# 参考文献：

1. 基于单片机的电梯控制系统设计

原文链接：https://blog.csdn.net/qq1744828575/article/details/136530642

1. 基于单片机的电梯控制系统（八楼控制+LCD1602+L298N+点击+按键+原理图、源代码、proteus仿真、动画）

原文链接：<https://blog.csdn.net/cqtianxingkeji/article/details/136052257>

1. STM32 基础实验蜂鸣器发声

原文链接：<https://blog.csdn.net/m0_69455439/article/details/124932955>

1. STM32矩形（矩阵）按键（键盘）输入控制LED灯—4\*4矩阵按键源码解析

原文链接：<https://blog.csdn.net/2301_79794623/article/details/133967592>

1. 基于STM32的步进电机驱动设计

原文链接：<https://blog.csdn.net/Dustinthewine/article/details/126530052>

# 对象协作图

