

《嵌入式系统》

(实验五 Linux驱动与应用设计实验)

厦门大学信息学院软件工程系 曾文华

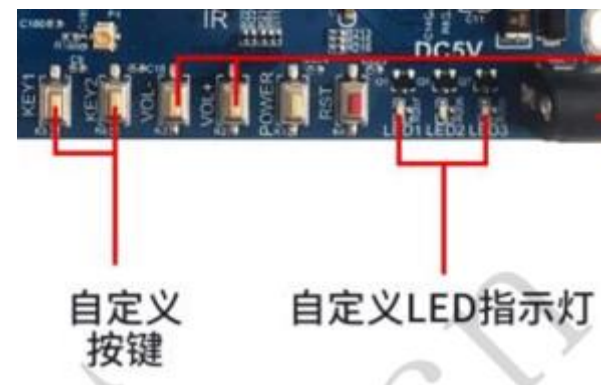
2024年11月6日

目录

- 设计实验1-1: 按键控制LED灯
- 设计实验1-2: 按键控制蜂鸣器
- 设计实验1-3: 按键控制蜂鸣器 (底板)
- 设计实验1-4: 按键控制步进电机
- 设计实验1-5: 按键控制直流电机
- 设计实验1-6: 按键控制舵机
- 设计实验1-7: 按键控制继电器
- 设计实验2-1: 小键盘控制LED灯
- 设计实验2-2: 小键盘控制蜂鸣器
- 设计实验2-3: 小键盘控制蜂鸣器 (底板)
- 设计实验2-4: 小键盘控制步进电机
- 设计实验2-5: 小键盘控制直流电机
- 设计实验2-6: 小键盘控制舵机
- 设计实验2-7: 小键盘控制继电器
- 设计实验3-1: 红外遥控器控制LED灯
- 设计实验3-2: 红外遥控器控制蜂鸣器
- 设计实验3-3: 红外遥控器控制蜂鸣器 (底板)
- 设计实验3-4: 红外遥控器控制步进电机
- 设计实验3-5: 红外遥控器控制直流电机
- 设计实验3-6: 红外遥控器控制舵机
- 设计实验3-7: 红外遥控器控制继电器
- 设计实验4-1: 电子钟
- 设计实验4-2: 数码管显示ADC值
- 设计实验4-3: 数码管显示温度值
- 挑战实验5-1: RS-485双机通信 (1)
- 挑战实验5-2: RS-485双机通信 (2)
- 挑战实验6-1: CAN总线双机通信 (1)
- 挑战实验6-2: CAN总线双机通信 (2)

设计实验1-1： 按键控制LED灯

- 要求： 使用2个按键控制2个LED灯的亮灭
 - 按KEY1键，LED1灯亮；再按KEY1键，LED1灯灭
 - 按KEY2键，LED2灯亮；再按KEY2键，LED2灯灭




- 编程思路： 对按键（keys）和LED灯（gpio_leds）的实验程序进行修改

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > keys		
<input type="checkbox"/> 名称	修改日期	
farsight_keys.c	2023/8/31 11:45	
keys.c	2024/9/20 16:14	
Makefile	2024/9/17 16:47	

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > gpio_leds		
<input type="checkbox"/> 名称	修改日期	
gpio_leds.c	2024/9/20 15:12	
leds_test.c	2024/9/20 15:26	
Makefile	2024/9/20 15:25	

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > keys_gpio_leds >

<input type="checkbox"/> 名称 ^	修改日期	类型
 keys_gpio_leds.c	2024/9/18 8:36	C 文件

提示 (1)

- 只需编写应用程序“**keys_gpio_leds.c**”，放在“**keys_gpio_leds**”目录中
- 设备号在应用程序中定义

```
#define KeyDevice "/dev/farsight_keys"           //按键的设备号  
#define LedDevice "/dev/leds_ctl"               //基于GPIO的LED灯设备号
```

```
fd_key=open(KeyDevice,O_RDWR);  
if (fd_key < 0)  
{  
    perror("Can't open file farsight_keys,Check your path");  
    return -1;  
}  
  
fd_led=open(LedDevice,O_RDWR);  
if (fd_led < 0)  
{  
    perror("Can't open file farsight_led,Check your path");  
    return -1;  
}
```

提示 (2)

- 应用程序的编译:

```
cd /home/linux/workdir/fs3399/application/keys_gpio_leds  
aarch64-linux-gnu-gcc keys_gpio_leds.c -o keys_gpio_leds  
cp keys_gpio_leds /mnt/hgfs/share/
```

- 应用程序的运行:

```
insmod farsight_keys.ko  
  
insmod gpio_leds.ko  
  
chmod 777 keys_gpio_leds  
./keys_gpio_leds  
  
rmmod farsight_keys  
rmmod gpio_leds
```

设计实验1-2： 按键控制蜂鸣器

- 要求： 使用按键KEY1控制蜂鸣器的响/不响
 - 按KEY1键，蜂鸣器响； 再按KEY1键，蜂鸣器不响









自定义
按键



蜂鸣器

- 编程思路： 对按键（keys）和蜂鸣器（buzzer）的实验程序进行修改

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > keys		
<input type="checkbox"/> 名称	修改日期	
 farsight_keys.c	2023/8/31 11:45	
 keys.c	2024/9/20 16:14	
 Makefile	2024/9/17 16:47	

此电脑 > Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > buzzer		
<input type="checkbox"/> 名称	修改日期	
 buzzer.c	2024/9/13 11:30	
 buzzer_driver.c	2024/9/13 11:32	
 Makefile	2024/9/17 15:45	

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > keys_buzzer >

☐ 名称

^

修改日期

类型



keys_buzzer.c




2024/9/18 10:10

C 文件




设计实验1-3： 按键控制蜂鸣器（底板）

- 要求：使用按键KEY1控制蜂鸣器（底板）的响/不响
 - 按KEY1键，实验箱底板上蜂鸣器响；再按KEY1键，实验箱底板上的蜂鸣器不响
- 编程思路：对按键（keys）和蜂鸣器（底板）（buzzer_pwm）的实验程序进行修改
- 执行程序前，先将D15拨到左边，其余在右边

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > keys



<input type="checkbox"/> 名称	修改日期
 farsight_keys.c	2023/8/31 11:45
 keys.c	2024/9/20 16:14
 Makefile	2024/9/17 16:47

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > buzzer_pwm

<input type="checkbox"/> 名称	修改日期
 buzzer_pwm.c	2024/9/21 8:50
 buzzer_pwm_driver.c	2024/9/13 8:08
 Makefile	2024/9/21 8:48



D15 拨到左边，其余在右边

此电脑 > Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > keys_buzzer_pwm >			
<input type="checkbox"/>	名称 ^	修改日期	类型
	 keys_buzzer_pwm.c	2024/9/18 10:50	C 文件

设计实验1-4： 按键控制步进电机

- **要求：使用2个按键控制步进电机的转动**
 - 按KEY1键，步进电机顺时针转；再按KEY1，停转
 - 按KEY2键，步进电机逆时针转；再按KEY2，停转
- 编程思路：对按键（keys）和步进电机（stepper）的实验程序进行修改，需要使用**多线程**
- 编译程序：
 - `aarch64-linux-gnu-gcc keys_stepper.c -o keys_stepper -lpthread`
- 执行程序前，先将D8、D9、D10、D11拨到左边，其余在右边


D8、D9、D10、D11拨到左边，其余在右边



Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > keys		
<input type="checkbox"/>	名称	修改日期
<input checked="" type="checkbox"/>	farsight_keys.c	2023/8/31 11:45
<input checked="" type="checkbox"/>	keys.c	2024/9/20 16:14
<input type="checkbox"/>	Makefile	2024/9/17 16:47

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > stepper		
<input type="checkbox"/>	名称	修改日期
<input type="checkbox"/>	Makefile	2024/9/21 9:16
<input checked="" type="checkbox"/>	stepper.c	2024/9/13 8:20
<input checked="" type="checkbox"/>	stepper_driver.c	2024/9/13 8:18

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > keys_stepper >

<input type="checkbox"/> 名称 ^	修改日期	类型
 keys_stepper.c	2024/9/18 11:00	C 文件

多线程程序样例



pthread.c - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "pthread.h"
```

```
#define BUFFER_SIZE 16
```

```
struct prodcons {
    int buffer[BUFFER_SIZE];
    pthread_mutex_t lock;
    int readpos, writepos;
    pthread_cond_t notempty;
    pthread_cond_t notfull;
```

```
};
```

```
/* the actual data */
```

```
/* mutex ensuring exclusive access to buffer */
```

```
/* positions for reading and writing */
```

```
/* signaled when buffer is not empty */
```

```
/* signaled when buffer is not full */
```

```
void init(struct prodcons * b)
{
    pthread_mutex_init(&b->lock, NULL);
    pthread_cond_init(&b->notempty, NULL);
    pthread_cond_init(&b->notfull, NULL);
    b->readpos = 0;
    b->writepos = 0;
}
```

```
void put(struct prodcons * b, int data)
{
    pthread_mutex_lock(&b->lock);

    /* Wait until buffer is not full */
    while ((b->writepos + 1) % BUFFER_SIZE == b->readpos)
    {
        printf("wait for not full\n");
        pthread_cond_wait(&b->notfull, &b->lock);
    }

    /* Write the data and advance write pointer */
    b->buffer[b->writepos] = data;
    b->writepos++;

    if (b->writepos >= BUFFER_SIZE)
        b->writepos = 0;

    /* Signal that the buffer is now not empty */
    pthread_cond_signal(&b->notempty);
    pthread_mutex_unlock(&b->lock);
}
```

```
int get(struct prodcons * b)
{
    int data;
    pthread_mutex_lock(&b->lock);

    /* Wait until buffer is not empty */
    while (b->writepos == b->readpos)
    {
        printf("wait for not empty\n");
        pthread_cond_wait(&b->notempty, &b->lock);
    }

    /* Read the data and advance read pointer */
    data = b->buffer[b->readpos];
    b->readpos++;
    if (b->readpos >= BUFFER_SIZE)
        b->readpos = 0;

    /* Signal that the buffer is now not full */
    pthread_cond_signal(&b->notfull);
    pthread_mutex_unlock(&b->lock);
    return data;
}
```



```
#define OVER (-1)
```

```
struct prodcons buffer;
```

```
→ void * producer(void * data)
{
    int n;

    for (n = 0; n < 100; n++)
    {
        printf(" put-->%d\n", n);
        put(&buffer, n);
    }

    put(&buffer, OVER);
    printf("producer stopped!\n");
    return NULL;
}
```

```
→ void * consumer(void * data)
{
    int d;

    while (1)
    {
        d = get(&buffer);
        if (d == OVER ) break;
        printf("      %d-->get\n", d);
    }

    printf("consumer stopped!\n");
    return NULL;
}
```

```
int main(void)
{
    pthread_t th_a, th_b;
    void * retval;

    init(&buffer);

    → pthread_create(&th_a, NULL, producer, 0);
    pthread_create(&th_b, NULL, consumer, 0);

    /* Wait until producer and consumer finish. */
    → pthread_join(th_a, &retval);
    pthread_join(th_b, &retval);

    return 0;
}
```

多线程程序样例的运行

- 将“**pthread**”文件夹拷贝到Ubuntu的**/home/linux/workdir/fs3399/application**目录下
- 在Ubuntu的“终端”上执行：
 - `cd /home/linux/workdir/fs3399/application/pthread`
 - `gcc pthread.c -o pthread_pc -lpthread`
 - `aarch64-linux-gnu-gcc pthread.c -o pthread -lpthread`
 - `cp pthread /mnt/hgfs/share/`
 - `./pthread_pc`

```
linux@linux-pc:~/workdir/fs3399/application/pthread$ ./pthread_pc
```

```
put-->0
put-->1
put-->2
put-->3
put-->4
put-->5
put-->6
put-->7
put-->8
put-->9
put-->10
put-->11
put-->12
put-->13
put-->14
put-->15
```

```
wait for not full
```

```
0-->get
1-->get
2-->get
3-->get
4-->get
5-->get
6-->get
7-->get
8-->get
9-->get
10-->get
11-->get
12-->get
13-->get
14-->get
```

```
wait for not empty
```

```
put-->16
put-->17
put-->18
put-->19
put-->20
put-->21
put-->22
put-->23
put-->24
put-->25
put-->26
put-->27
put-->28
put-->29
put-->30
```

```
wait for not full
```

```
15-->get
16-->get
17-->get
18-->get
19-->get
20-->get
21-->get
22-->get
23-->get
24-->get
25-->get
26-->get
27-->get
28-->get
29-->get
```

```
wait for not empty
```

```
put-->31
put-->32
```

```
put-->81
put-->82
put-->83
put-->84
put-->85
put-->86
put-->87
```

```
wait for not full
```

```
72-->get
73-->get
74-->get
75-->get
76-->get
77-->get
78-->get
79-->get
80-->get
81-->get
82-->get
83-->get
84-->get
85-->get
86-->get
```

```
wait for not empty
```

```
put-->88
put-->89
put-->90
put-->91
put-->92
put-->93
put-->94
put-->95
put-->96
put-->97
put-->98
put-->99
```

```
producer stopped!
```

```
87-->get
88-->get
89-->get
90-->get
91-->get
92-->get
93-->get
94-->get
95-->get
96-->get
97-->get
98-->get
99-->get
```

```
consumer stopped!
```

```
linux@linux-pc:~/workdir/fs3399/application/pthread$
```


设计实验1-5： 按键控制直流电机

- 要求：使用2个按键控制直流电机的转动
 - 按KEY1键，直流电机逆时针转；再按KEY1键，直流电机停转
 - 按KEY2键，直流电机顺时针转；再按KEY2键，直流电机停转
- 编程思路：对按键（keys）和直流电机（dcmotor）的实验程序进行修改
- 执行程序前，先将D12、D13拨到左边，其余在右边

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > keys		
<input type="checkbox"/> 名称	修改日期	
farsight_keys.c	2023/8/31 11:45	
keys.c	2024/9/20 16:14	
Makefile	2024/9/17 16:47	

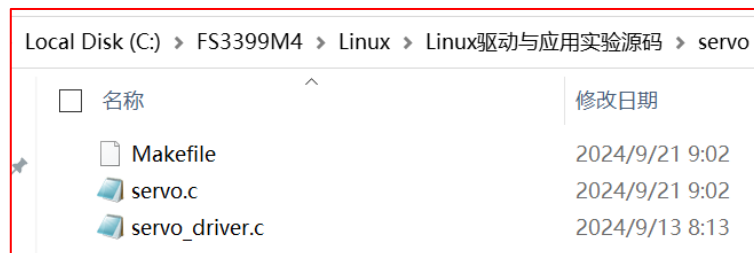
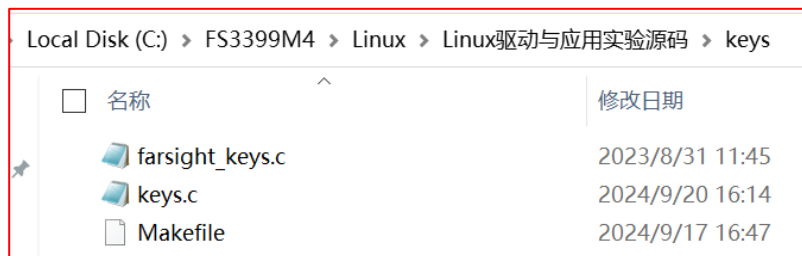
Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > dcmotor		
<input type="checkbox"/> 名称	修改日期	
dcmotor.c	2024/9/13 8:34	
dcmotor_driver.c	2024/9/21 9:28	
Makefile	2024/9/21 9:28	



Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > keys_dcmotor >		
<input type="checkbox"/> 名称	修改日期	类型
 keys_dcmotor.c	2024/9/18 11:45	C 文件

设计实验1-6： 按键控制舵机

- 要求：使用按键KEY1控制舵机的转动
 - 按下KEY1，舵机转动；再按下KEY1，舵机不转动
- 编程思路：对按键（keys）和舵机（servo）的实验程序进行修改，需要使用多线程
- 执行程序前，先将D6拨到左边，其余在右边




D6 拨到左边，其余在右边

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > keys_servo >

☐ 名称

修改日期

类型




 keys_servo.c




2024/9/18 15:06

C 文件

设计实验1-7： 按键控制继电器

- 要求：使用按键KEY1控制继电器的打开/关闭
 - 按下KEY1，继电器打开；再按下KEY1，继电器关闭
- 编程思路：对按键（keys）和继电器（relay）的实验程序进行修改
- 执行程序前，先将D16拨到左边，其余在右边

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > keys		
<input type="checkbox"/> 名称	修改日期	
 farsight_keys.c	2023/8/31 11:45	
 keys.c	2024/9/20 16:14	
 Makefile	2024/9/17 16:47	

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > relay		
<input type="checkbox"/> 名称	修改日期	
 Makefile	2024/9/21 8:25	
 relay.c	2024/9/21 8:26	
 relay_driver.c	2024/9/21 8:28	



D16 拨到左边，其余在右边

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > keys_relay >

☐ 名称



修改日期

类型



keys_relay.c

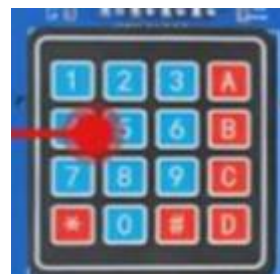
2024/9/18 15:19

C 文件

设计实验2-1：小键盘控制LED灯

- 要求：使用小键盘的3个键控制3个LED灯的亮灭

- 按下键“1”，LED1灯亮；再按下键“1”，LED1灯灭
- 按下键“2”，LED2灯亮；再按下键“2”，LED2灯灭
- 按下键“3”，LED3灯亮；再按下键“3”，LED3灯灭



自定义LED指示灯

- 编程思路：对小键盘/数码管（zlg72xx）和LED灯（gpio_leds）的实验程序进行修改

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > zlg72xx	
名称	修改日期
Makefile	2024/9/17 17:35
zlg72xx.c	2024/9/21 10:39
zlg72xx_driver.c	2023/8/31 11:45


Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > gpio_leds	
名称	修改日期
gpio_leds.c	2024/9/20 15:12
leds_test.c	2024/9/20 15:26
Makefile	2024/9/20 15:25

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > zlg72xx_gpio_leds >

☐ 名称 ^

修改日期

类型

 zlg72xx_gpio_leds.c

2024/9/18 15:42

C 文件

设计实验2-2：小键盘控制蜂鸣器

- 要求：使用小键盘的键“1”控制蜂鸣器的响/不响
 - 按下键“1”，蜂鸣器响；再按下键“1”，蜂鸣器不响
- 编程思路：对小键盘/数码管（zlg72xx）和蜂鸣器（buzzer）的实验程序进行修改

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > zlg72xx

<input type="checkbox"/> 名称	修改日期
Makefile	2024/9/17 17:35
zlg72xx.c	2024/9/21 10:39
zlg72xx_driver.c	2023/8/31 11:45

此电脑 > Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > buzzer


<input type="checkbox"/> 名称	修改日期
buzzer.c	2024/9/13 11:30
buzzer_driver.c	2024/9/13 11:32
Makefile	2024/9/17 15:45

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > zlg72xx_buzzer >

☐ 名称

修改日期

类型

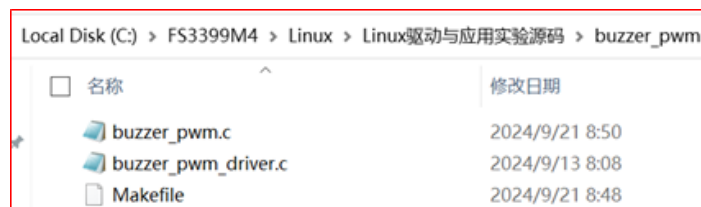
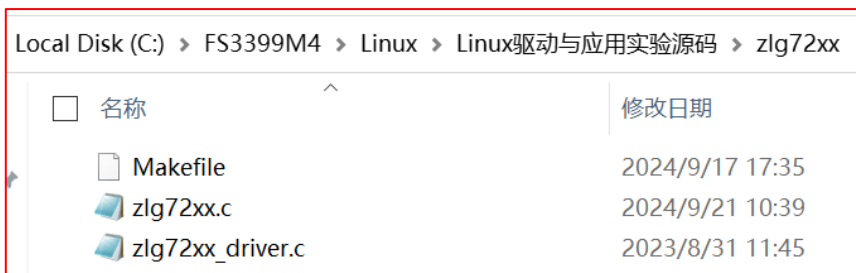
 zlg72xx_buzzer.c

2024/9/18 16:26

C 文件

设计实验2-3：小键盘控制蜂鸣器（底板）

- 要求：使用小键盘的键“1”控制蜂鸣器（底板）的响/不响
 - 按下键“1”，蜂鸣器（底板）响；再按下键“1”，蜂鸣器（底板）不响
- 编程思路：对小键盘/数码管（zlg72xx）和蜂鸣器（底板）（buzzer_pwm）的实验程序进行修改
- 执行程序前，先将D15拨到左边，其余在右边



D15 拨到左边，其余在右边

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > zlg72xx_buzzer_pwm >



名称



修改日期

类型



zlg72xx_buzzer_pwm.c

2024/9/18 16:39

C 文件

设计实验2-4：小键盘控制步进电机

- **要求：使用小键盘的2个键控制步进电机的转动**
 - 按下键“1”，步进电机顺时针转；再按下键“1”，步进电机停转
 - 按下键“2”，步进电机逆时针转；再按下键“2”，步进电机停转
- 编程思路：对小键盘/数码管（zlg72xx）和步进电机（stepper）的实验程序进行修改，需要使用**多线程**
- 执行程序前，先将D8、D9、D10、D11拨到左边，其余在右边

D8、D9、D10、D11拨到左边，其余在右边



Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > zlg72xx	
<input type="checkbox"/> 名称	修改日期
<input type="checkbox"/> Makefile	2024/9/17 17:35
<input type="checkbox"/> zlg72xx.c	2024/9/21 10:39
<input type="checkbox"/> zlg72xx_driver.c	2023/8/31 11:45


Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > stepper	
<input type="checkbox"/> 名称	修改日期
<input type="checkbox"/> Makefile	2024/9/21 9:16
<input type="checkbox"/> stepper.c	2024/9/13 8:20
<input type="checkbox"/> stepper_driver.c	2024/9/13 8:18

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > zlg72xx_stepper >

☐ 名称

修改日期

类型

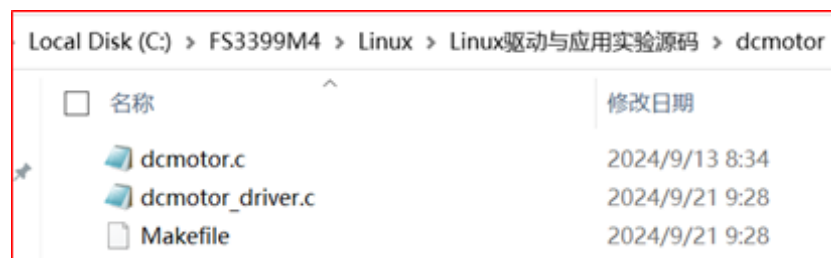
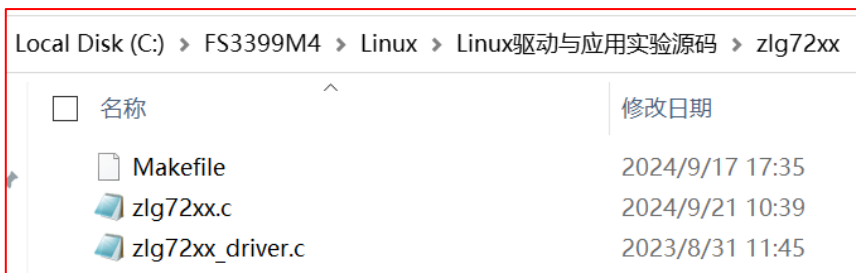
 zlg72xx_stepper.c


2024/9/18 16:54

C 文件

设计实验2-5：小键盘控制直流电机

- **要求：使用小键盘的2个键控制直流电机的转动**
 - 按下键“1”，直流电机逆时针转；再按下键“1”，直流电机停转
 - 按下键“2”，直流电机顺时针转；再按下键“2”，直流电机停转
- 编程思路：对小键盘/数码管（zlg72xx）和直流电机（dcmotor）的实验程序进行修改
- 执行程序前，先将D12、D13拨到左边，其余在右边



Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > zlg72xx_dcmotor >		
<input type="checkbox"/> 名称	修改日期	类型
 zlg72xx_dcmotor.c	2024/9/18 17:14	C 文件

设计实验2-6：小键盘控制舵机

- **要求：使用小键盘的键“1”控制舵机的转动**
 - 按下键“1”，舵机转动；再按下键“1”，舵机不转
- 编程思路：对小键盘/数码管（zlg72xx）和舵机（servo）的实验程序进行修改，需要使用**多线程**
- 执行程序前，先将D6拨到左边，其余在右边




D6 拨到左边，其余在右边

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > zlg72xx		
<input type="checkbox"/> 名称	修改日期	
<input type="checkbox"/> Makefile	2024/9/17 17:35	
<input type="checkbox"/> zlg72xx.c	2024/9/21 10:39	
<input type="checkbox"/> zlg72xx_driver.c	2023/8/31 11:45	

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用实验源码 > servo		
<input type="checkbox"/> 名称	修改日期	
<input type="checkbox"/> Makefile	2024/9/21 9:02	
<input type="checkbox"/> servo.c	2024/9/21 9:02	
<input type="checkbox"/> servo_driver.c	2024/9/13 8:13	

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > zlg72xx_servo >

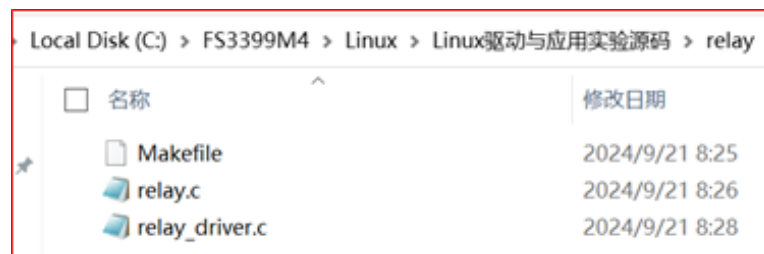
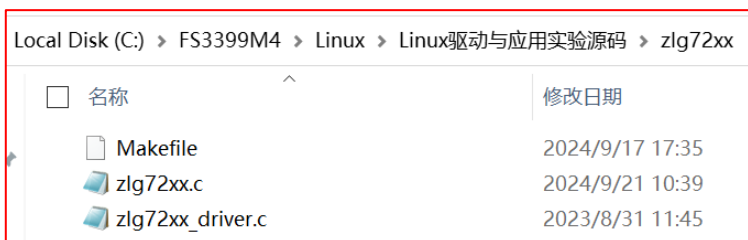
<input type="checkbox"/> 名称 ^	修改日期	类型
 zlg72xx_servo.c	2024/9/18 22:24	C 文件

设计实验2-7：小键盘控制继电器



- **要求：使用小键盘的键“1”控制继电器的打开/关闭**
 - 按下键“1”，打开继电器；再按下键“1”，关闭继电器
- 编程思路：对小键盘/数码管（zlg72xx）和继电器（relay）的实验程序进行修改
- 执行程序前，先将D16拨到左边，其余在右边



D16 拨到左边，其余在右边



此电脑 > Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > zlg72xx_relay >

<input type="checkbox"/>	名称	修改日期	类型
	 zlg72xx_relay.c	2024/9/18 22:30	C 文件

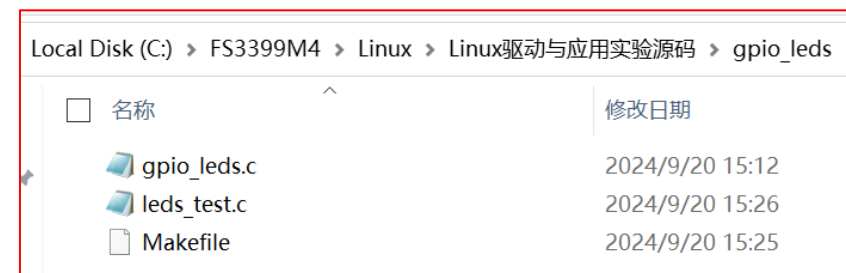
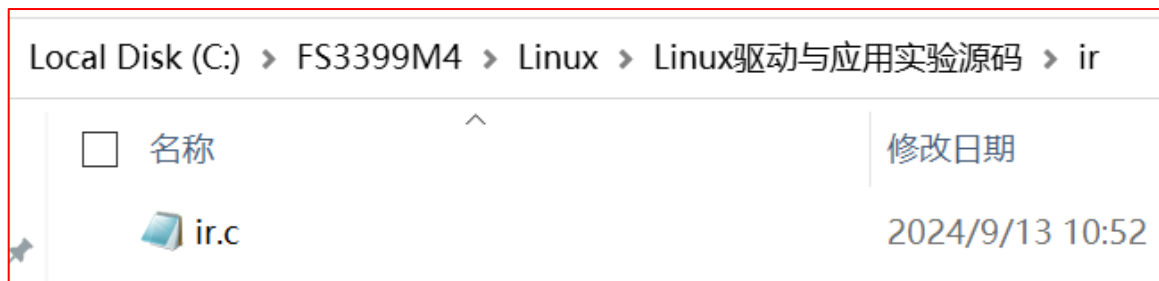
设计实验3-1： 红外遥控器控制LED灯

- **要求：使用红外遥控器的3个按键控制3个LED灯的亮灭**
 - 按红外遥控器的键“1”，LED1灯亮；再按键“1”，LED1灯灭
 - 按红外遥控器的键“2”，LED2灯亮；再按键“2”，LED2灯灭
 - 按红外遥控器的键“3”，LED3灯亮；再按键“3”，LED3灯灭




自定义LED指示灯

- **编程思路：对红外遥控器（ir）和LED灯（gpio_leds）的实验程序进行修改**

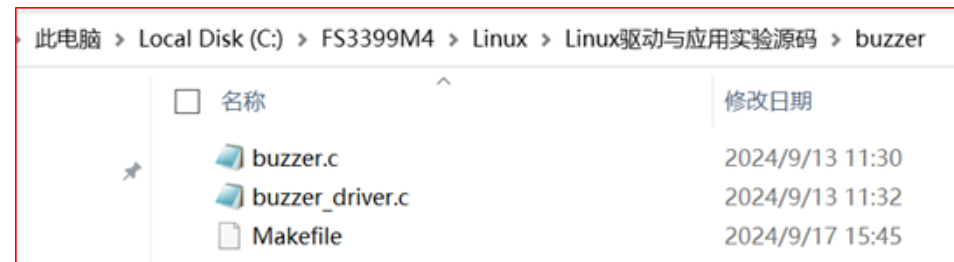
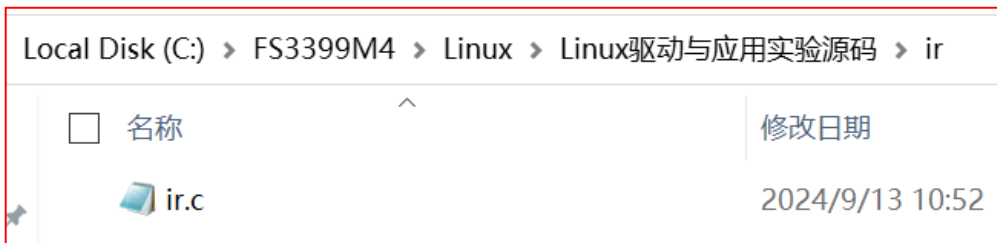


Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > ir_gpio_leds >

<input type="checkbox"/> 名称 ^	修改日期	类型
 ir_gpio_leds.c	2024/9/8 21:51	C 文件

设计实验3-2： 红外遥控器控制蜂鸣器

- 要求：使用红外遥控器的键“1” 控制蜂鸣器的响/不响
 - 按红外遥控器的键“1”，蜂鸣器响；再按键“1”，蜂鸣器不响
- 编程思路：对红外遥控器（ir）和蜂鸣器（buzzer）的实验程序进行修改

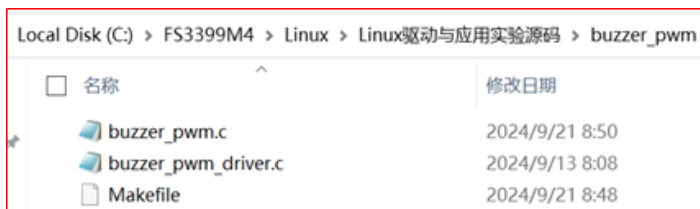
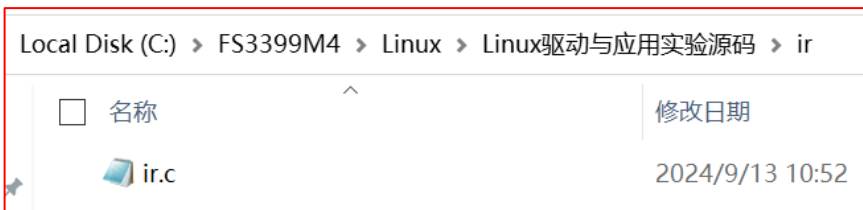


Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > ir_buzzer >

<input type="checkbox"/> 名称 ^	修改日期	类型
 ir_buzzer.c	2024/9/19 11:08	C 文件


设计实验3-3： 红外遥控器控制蜂鸣器（底板）

- 要求：使用红外遥控器的键“1”控制蜂鸣器（底板）响/不响
 - 按红外遥控器的键“1”，蜂鸣器（底板）响；再按键“1”，蜂鸣器（底板）不响
- 编程思路：对红外遥控器（ir）和蜂鸣器（底板）（buzzer_pwm）的实验程序进行修改
- 执行程序前，先将D15拨到左边，其余在右边



D15 拨到左边，其余在右边

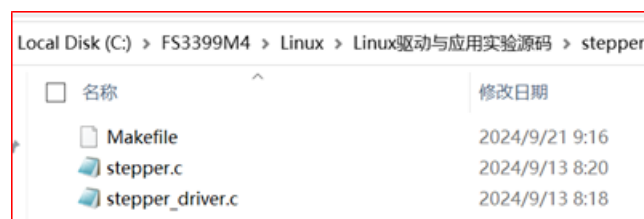
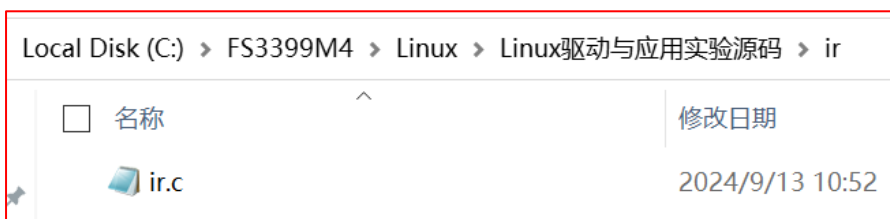
Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > ir_buzzer_pwm >


<input type="checkbox"/> 名称	修改日期	类型
 ir_buzzer_pwm.c	2024/9/8 18:05	C 文件

设计实验3-4： 红外遥控器控制步进电机

- **要求：使用红外遥控器的2个键控制步进电机的转动**
 - 按红外遥控器的键“1”，步进电机顺时针转；再按键“1”，步进电机停转
 - 按红外遥控器的键“2”，步进电机逆时针转；再按键“2”，步进电机停转
- 编程思路：对红外遥控器（ir）和步进电机（stepper）的实验程序进行修改，需要实验**多线程**
- 执行程序前，先将D8、D9、D10、D11拨到左边，其余在右边

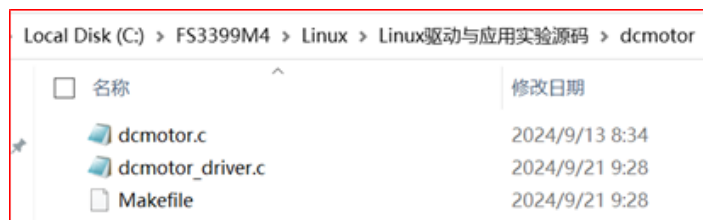
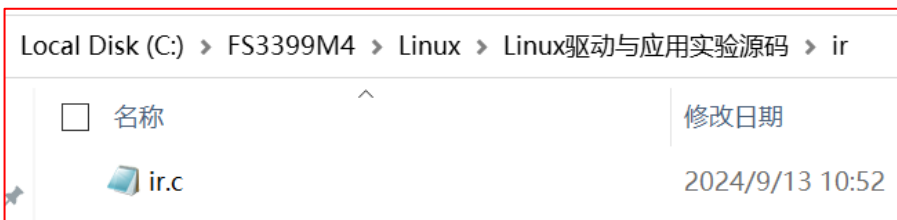
D8、D9、D10、D11拨到左边，其余在右边



Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > ir_stepper		
<input type="checkbox"/>	名称 ^	修改日期
<input checked="" type="checkbox"/>	 ir_stepper.c	2024/9/19 11:18
		C 文件

设计实验3-5： 红外遥控器控制直流电机

- **要求：使用红外遥控器的2个键控制直流电机的转动**
 - 按红外遥控器的键“1”，直流电机顺时针转；再按键“1”，直流电机停转
 - 按红外遥控器的键“2”，直流电机逆时针转；再按键“2”，直流电机停转
- 编程思路：对红外遥控器（ir）和直流电机（dcmotor）的实验程序进行修改
- 执行程序前，先将D12、D13拨到左边，其余在右边



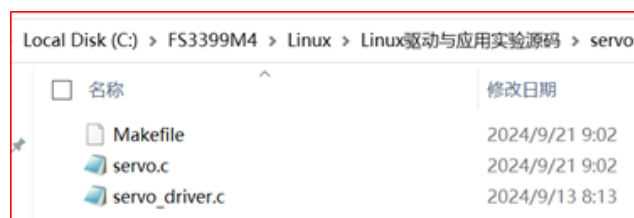
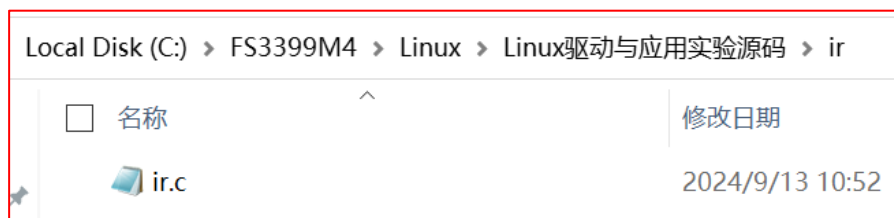
此电脑 > Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > ir_dcmotor >			
<input type="checkbox"/>	名称 ^	修改日期	类型
024年下半年)	 ir_dcmotor.c	2024/9/19 11:50	C 文件

设计实验3-6： 红外遥控器控制舵机

- 要求：使用红外遥控器的键“1” 控制舵机的转动
 - 按红外遥控器的键“1”，舵机转动；再按键“1”，舵机停转
- 编程思路：对红外遥控器（ir）和舵机（servo）的实验程序进行修改，需要使用多线程
- 执行程序前，先将D6拨到左边，其余在右边



D6 拨到左边，其余在右边



此电脑 > Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > ir_servo >

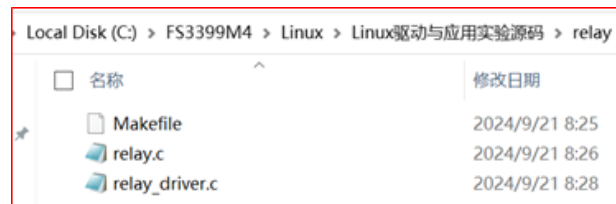
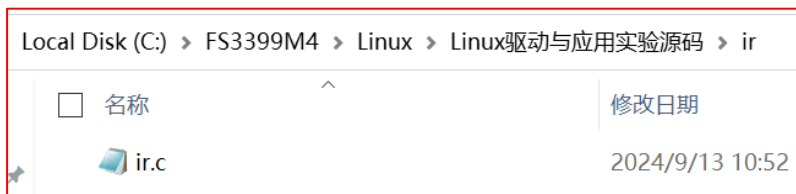
	<input type="checkbox"/> 名称	修改日期	类型
 024年下半年)	 ir_servo.c	2024/9/19 12:01	C 文件

设计实验3-7： 红外遥控器控制继电器


- **要求：使用红外遥控器的键“1”控制继电器的打开/关闭**
 - 按红外遥控器的键“1”，打开继电器；再按键“1”，关闭继电器
- 编程思路：对红外遥控器（ir）和继电器（relay）的实验程序进行修改
- 执行程序前，先将D16拨到左边，其余在右边



D16 拨到左边，其余在右边



Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > ir_relay >


<input type="checkbox"/> 名称 ^	修改日期	类型
 ir_relay.c	2024/9/19 12:13	C 文件

设计实验4-1： 电子钟

- 电子钟：

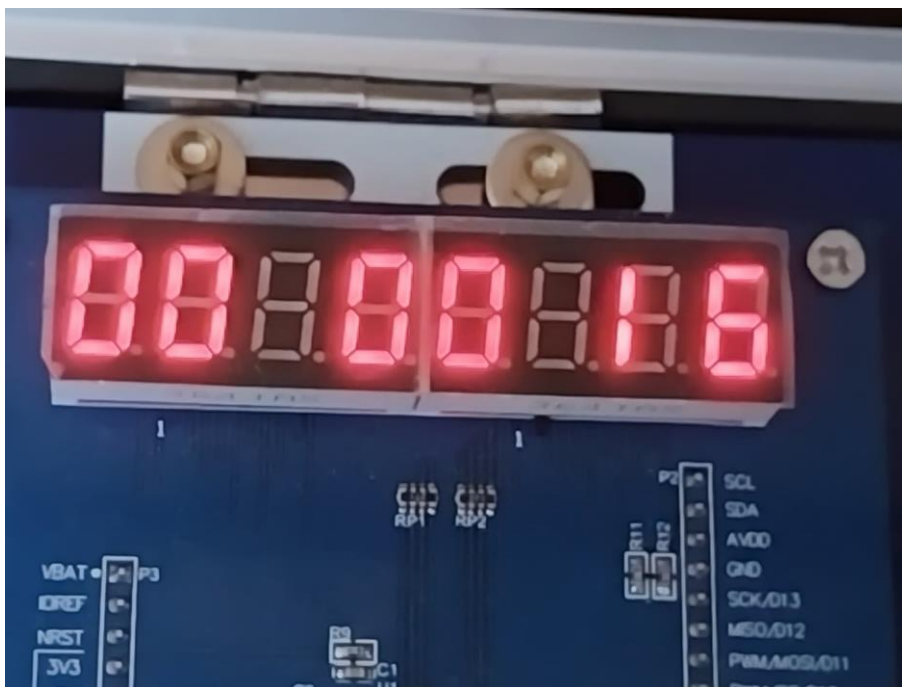
- 在数码管上显示时、分、秒
- 显示格式： 23-59-50
- 每1秒钟时间变化一次
- 修改程序， 获取系统的时钟（当前时钟）， 并在数码管上显示！

Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > zlg72xx_clock >

<input type="checkbox"/> 名称 ^	修改日期	类型
 zlg72xx_clock.c	2024/9/19 16:07	C 文件

提示

- 需要修改小键盘/数码管的驱动程序“**zlg72xx_driver.c**”，使其能够显示“-”、“0.”、……、“9.”




有两个地方！

在这里增加显示“-”、“0.”、……、“9.”的程序

```
switch(buf[i])
{
    case '0': val[1] = 0xFC; break;
    case '1': val[1] = 0x0C; break;
    case '2': val[1] = 0xDA; break;
    case '3': val[1] = 0xF2; break;
    case '4': val[1] = 0x66; break;
    case '5': val[1] = 0xB6; break;
    case '6': val[1] = 0xBE; break;
    case '7': val[1] = 0xE0; break;
    case '8': val[1] = 0xFE; break;
    case '9': val[1] = 0xF6; break;
    case 'a':
    case 'A': val[1] = 0xEE; break;
    case 'b':
    case 'B': val[1] = 0x3E; break;           //7F
    case 'c':
    case 'C': val[1] = 0x9C; break;
    case 'd':
    case 'D': val[1] = 0x7A; break;           //3F
    case 'e':
    case 'E': val[1] = 0x9E; break;
    case 'f':
    case 'F': val[1] = 0x8E; break;
    case ' ': val[1] = 0x00; break;
    case '.':
        if(val[1] != 0x00)
            val[1] |= 0x01;
        break;
    default:
        val[1] = 0x00; break;
}
```

第5次实验代码 (答案) > Linux驱动与应用设计实验源码 (答案) > zlg72xx_driver

名称	修改日期	类型
 Makefile	2024/10/27 20:03	文件
 zlg72xx_driver.c	2024/10/27 20:12	C 文件

```
cd /home/linux/workdir/fs3399/application/zlg72xx_driver  
  
make  
  
cp zlg72xx_driver.ko /mnt/hgfs/share/
```

先将实验箱“app”目录中的“zlg72xx_driver.ko”文件删除，
然后再将电脑“share”目录中的“zlg72xx_driver.ko”传到实
验箱的“app”目录中

```
insmod zlg72xx_driver.ko
```

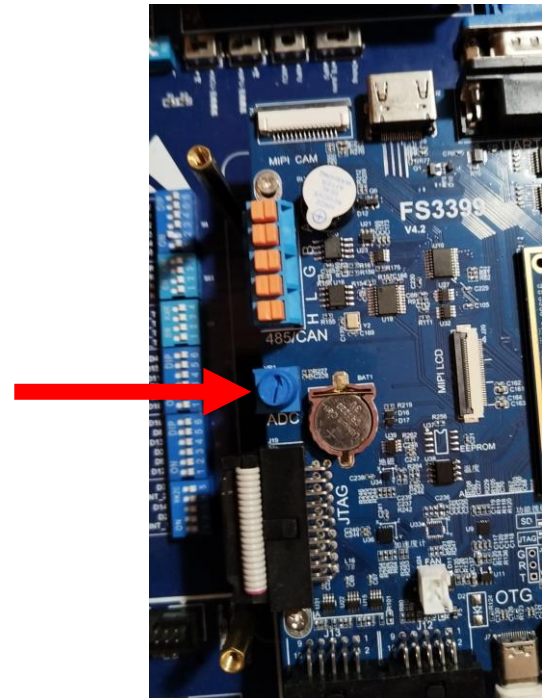
```
chmod 777 zlg72xx_clock  
./zlg72xx_clock
```

```
rmmod zlg72xx_driver
```

设计实验4-2：数码管显示ADC值

- 在数码管上显示ADC信号采集值：
 - 将实验8（ADC信号采集）得到的电位器对应的数值在数码管上显示
 - 显示的数值范围为：0 ~ 1023
 - 旋转电位器，数值会变化（从0变化到1023）

电位器

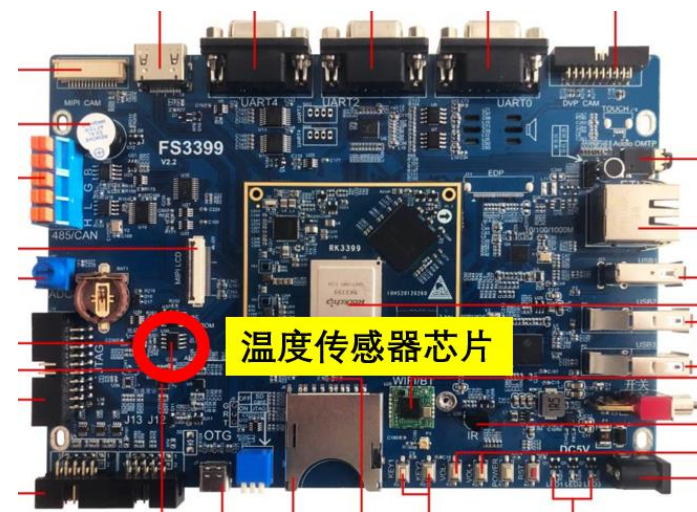


Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > zlg72xx_adc >

<input type="checkbox"/> 名称	修改日期	类型
 zlg72xx_adc.c	2024/9/19 22:15	C 文件

设计实验4-3：数码管显示温度值

- 在数码管上显示温度传感器采集值：
 - 将实验9（温度传感器）得到的数值（温度值）在数码管上显示
 - 显示的格式为： **XX.XXX**
 - 将手指头按在温度传感器芯片上，会改变温度值

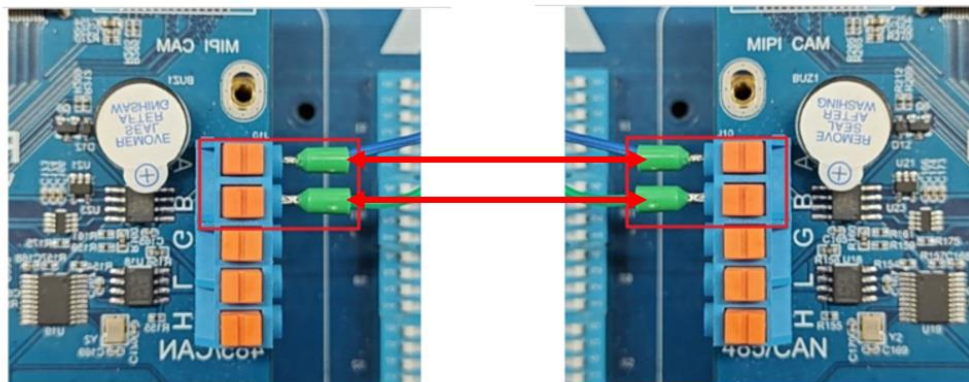


Local Disk (C:) > FS3399M4 > Linux > Linux驱动与应用设计实验源码 (答案) > zlg72xx_lm75 >

<input type="checkbox"/> 名称 ^	修改日期	类型
 zlg72xx_lm75.c	2024/9/19 17:32	C 文件

挑战实验5-1： RS-485双机通信 （1）

- RS485双机通信（1）：
 - 在第1台电脑的键盘上按任意键，会在第2台电脑的显示器上显示该按键值
 - 同样，在第2台电脑的键盘上按任意键，会在第1台电脑的显示器上显示该按键值

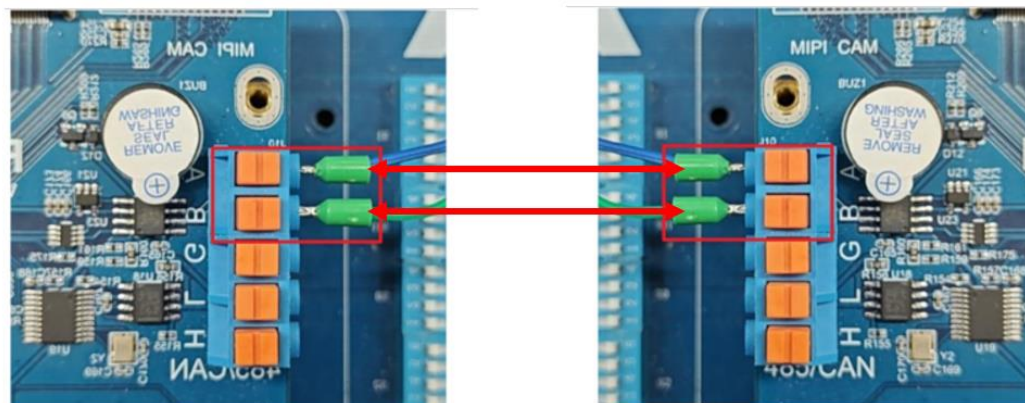


Local Disk (C:) > 嵌入式系统 (2024年下半年) > 第4次实验代码 (答案) > Linux驱动与应用设计实验源码 (答案) > rs485_1




<input type="checkbox"/> 名称 ^	修改日期	类型	大小
 rs485_1.c	2024/10/5 17:41	C 文件	4 KB


挑战实验5-2： RS-485双机通信 (2)

- RS485双机通信 (2) :
 - 将第1台电脑上的文件通过RS-485总线传送到第2台电脑上



Local Disk (C:) > 嵌入式系统 (2024年下半年) > 第4次实验代码 (答案) > Linux驱动与应用设计实验源码 (答案) > rs485_2

<input type="checkbox"/> 名称	修改日期	类型 ^	大小
 rs485_2_r.c	2024/10/5 17:44	C 文件	4 KB
 rs485_2_s.c	2024/10/5 17:43	C 文件	4 KB
 test.txt	2019/7/4 11:43	文本文档	1 KB

 test.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#include <stdio.h>
```

```
#include <fcntl.h>
```

```
int main()
```

```
{
```

```
    printf("hello world! \n");
```

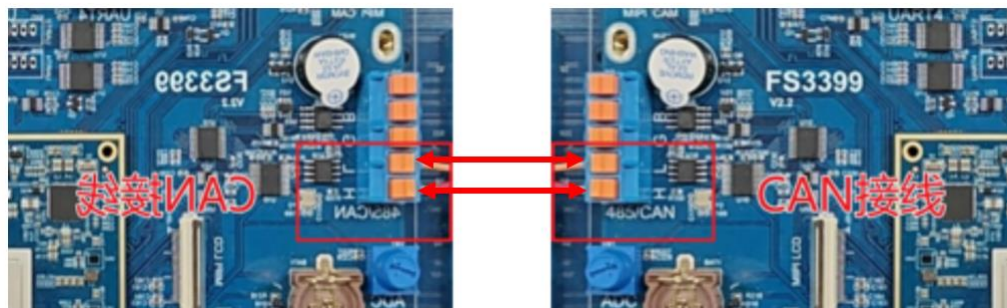
```
    return 0;
```

```
}
```

挑战实验6-1： CAN总线双机通信 （1）

- CAN总线双机通信（1）：

- 在第1台电脑的键盘上按任意键，会在第2台电脑的显示器上显示该按键值
- 同样，在第2台电脑的键盘上按任意键，会在第1台电脑的显示器上显示该按键值

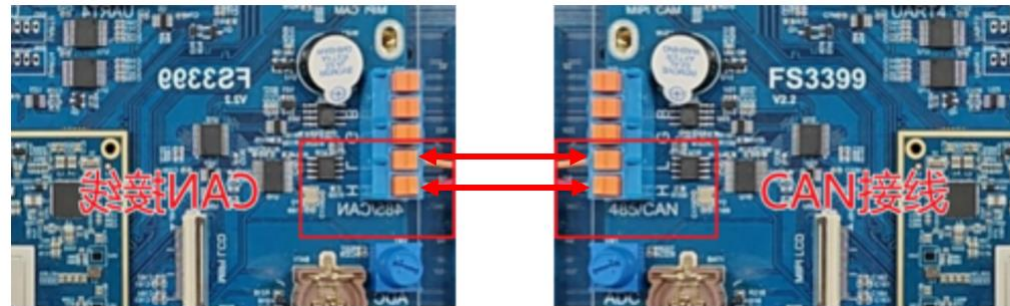


Local Disk (C:) > 嵌入式系统 (2024年下半年) > 第4次实验代码 (答案) > Linux驱动与应用设计实验源码 (答案) > can_1









<input type="checkbox"/> 名称	修改日期	类型 ^	大小
 can_1.c	2024/10/5 17:40	C 文件	3 KB
 libsocketcan.c	2024/10/3 21:36	C 文件	15 KB
 can_config.h	2024/10/3 21:39	H 文件	1 KB
 can_netlink.h	2024/10/3 21:38	H 文件	3 KB
 libsocketcan.h	2024/10/3 21:37	H 文件	2 KB
 sockcan.h	2024/10/3 21:43	H 文件	4 KB

挑战实验6-2：CAN总线双机通信（2）

- CAN总线双机通信（2）：
 - 将第1台电脑上的文件通过CAN总线传送到第2台电脑上



Local Disk (C:) > 嵌入式系统 (2024年下半年) > 第4次实验代码 (答案) > Linux驱动与应用设计实验源码 (答案) > can_2

<input type="checkbox"/> 名称	修改日期	类型 ^	大小
 can_2_r.c	2024/10/5 17:34	C 文件	4 KB
 can_2_s.c	2024/10/5 17:38	C 文件	4 KB
 libsocketcan.c	2024/10/3 21:36	C 文件	15 KB
 can_config.h	2024/10/3 21:39	H 文件	1 KB
 can_netlink.h	2024/10/3 21:38	H 文件	3 KB
 libsocketcan.h	2024/10/3 21:37	H 文件	2 KB
 sockcan.h	2024/10/3 21:43	H 文件	4 KB
 test.txt	2019/7/4 11:43	文本文档	1 KB

test.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#include <stdio.h>
#include <fcntl.h>

int main()
{
    printf("hello world! \n");
    return 0;
}
```


要求

- 每个同学需要完成**4个设计实验+1个挑战实验**：

- 1、设计实验1-1~1-7中选择一个（按键控制）
- 2、设计实验2-1~2-7中选择一个（小键盘控制）
- 3、设计实验3-1~3-7中选择一个（红外遥控器控制）

- 4、上述21个实验中，**必须选择1个控制步进电机的实验，所选择的3个实验必须是控制不同的设备**

- 5、设计实验4-1~4-3中任选1个（数码管显示）

- 6、挑战实验：实验5-1、5-2、6-1、6-2中任选一个（双机通信），完成挑战实验的加10分

- 请在11月19日前完成，这期间会安排若干次实验室开放时间，请同学在自己的电脑上先将程序编译通过，然后利用实验室开放时间，到实验室运行（调试）程序。

- 请按照实验报告的模板撰写实验报告，第5次实验报告提交的截止日期为**2024年11月18日晚上24点**。

- | | |
|------------------------|--------------------------|
| • 设计实验1-1：按键控制LED灯 | • 设计实验3-1：红外遥控器控制LED灯 |
| • 设计实验1-2：按键控制蜂鸣器 | • 设计实验3-2：红外遥控器控制蜂鸣器 |
| • 设计实验1-3：按键控制蜂鸣器（底板） | • 设计实验3-3：红外遥控器控制蜂鸣器（底板） |
| • 设计实验1-4：按键控制步进电机 | • 设计实验3-4：红外遥控器控制步进电机 |
| • 设计实验1-5：按键控制直流电机 | • 设计实验3-5：红外遥控器控制直流电机 |
| • 设计实验1-6：按键控制舵机 | • 设计实验3-6：红外遥控器控制舵机 |
| • 设计实验1-7：按键控制继电器 | • 设计实验3-7：红外遥控器控制继电器 |
| • 设计实验2-1：小键盘控制LED灯 | • 设计实验4-1：电子钟 |
| • 设计实验2-2：小键盘控制蜂鸣器 | • 设计实验4-2：数码管显示ADC值 |
| • 设计实验2-3：小键盘控制蜂鸣器（底板） | • 设计实验4-3：数码管显示温度值 |
| • 设计实验2-4：小键盘控制步进电机 | • 挑战实验5-1：RS-485双机通信（1） |
| • 设计实验2-5：小键盘控制直流电机 | • 挑战实验5-2：RS-485双机通信（2） |
| • 设计实验2-6：小键盘控制舵机 | • 挑战实验6-1：CAN总线双机通信（1） |
| • 设计实验2-7：小键盘控制继电器 | • 挑战实验6-2：CAN总线双机通信（2） |

Thanks