

Assignment 008: Lab 8: 网络LED矩阵显示器

一、实验目的

掌握Linux设备驱动程序的开发过程；

理解I2C总线协议；

复习socket编程（网络原理课）；

实现一个网络访问的LED矩阵显示器。

二、实验器材

硬件

pcDuino v2板一块；

5V/1A电源一个；

microUSB线一根；

面包板一块；

8x8 LED矩阵一块（不带I2C控制器）；

360Ω 1/8W电阻8颗，或360Ω 排阻1颗；

面包线若干。

以下为自备（可选）器材：

PC（Windows/Mac OS/Linux）一台；

USB-TTL串口线一根（FT232RL芯片或PL2303芯片）；

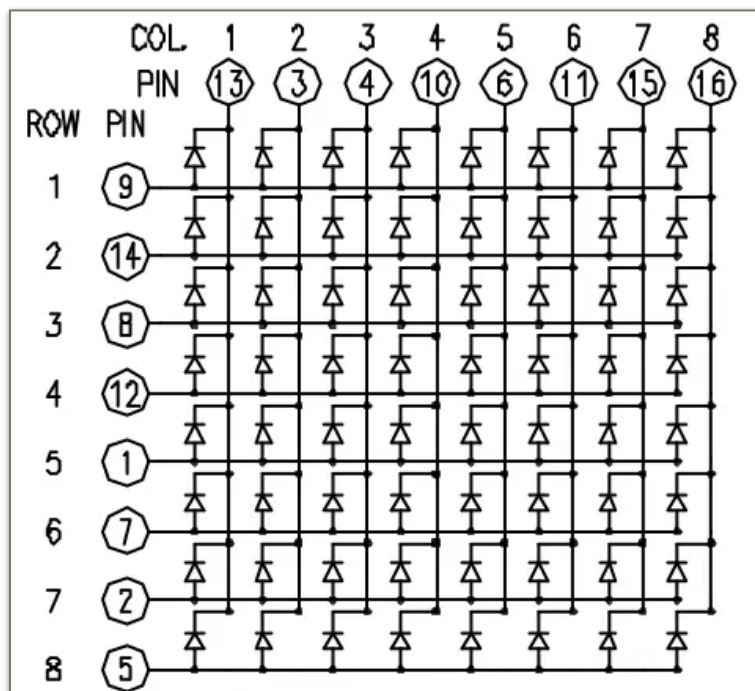
以太网线一根（可能还需要路由器等）。

软件

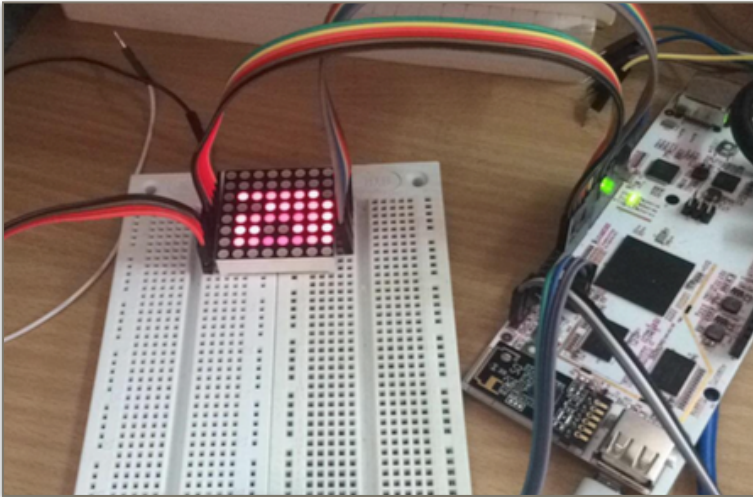
编译软件

三、实验步骤

1 LedMatrix 原理图：



2 实际连线



3 GPIO 库控制 LEDMatrix

定义引脚

```
//define the pins for COL
byte ROW[] = {
  7,2,15,4,8,14,9,12};
//define the pins for ROW
byte COL[] = {
  3,10,11,6,13,5,1,0};
// {0,1,5,13,6,11,10,3};
```

字模的定义(以a为例)

```
9 byte zimu_a[] ={
10 0b00000000,
11 0b00000000,
12 0b01111110,
13 0b00011111,
14 0b01111111,
15 0b01100111,
16 0b01111111,
17 0b00000000,
18 };
19
```

键盘输入获取

```
char catstr__[2];
gets(catstr__);
printf("%s\n",catstr__ );
bitmap=rebitmap(catstr__);
```

loop循环

```
count--;
// printf("--%d\n",count);
if(count==0){
  char catstr__[2];
  gets(catstr__);
  printf("%s\n",catstr__ );
  bitmap=rebitmap(catstr__);
  count=10000;
}
```

分时复用LedMatrix:

```
for(i=0;i<8;i++){  
    pickcol(i);  
    for (j = 0; j < 8; ++j)  
    {  
        pickrow(j,(bitmap[i] >> j) & 0b00000001);  
    }  
}
```

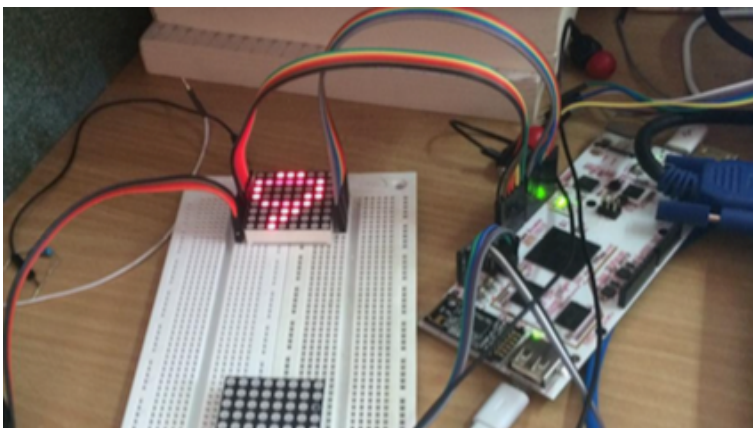
```
void pickcol(int colno){  
    int i;  
    clearrow();  
    for(i=0;i<8;i++){  
        digitalWrite(COL[i], HIGH);  
    }  
    digitalWrite(COL[colno], LOW);  
}  
  
void pickrow(int rowno,int l_h){  
    digitalWrite(ROW[rowno],l_h);  
}
```

```
void clearrow(){  
    int i;  
    for (i = 0; i < 8; ++i)  
    {  
        /* code */  
        digitalWrite(ROW[i],LOW);  
    }  
}
```

Rebitmap函数的定义

```
byte* rebitmap(char* catstr){  
    if(strcmp(catstr,"0")==0){  
        return zimu_0;  
    }  
    else if(strcmp(catstr,"1")==0){  
        return zimu_1;  
    }  
}
```

4 运行显示



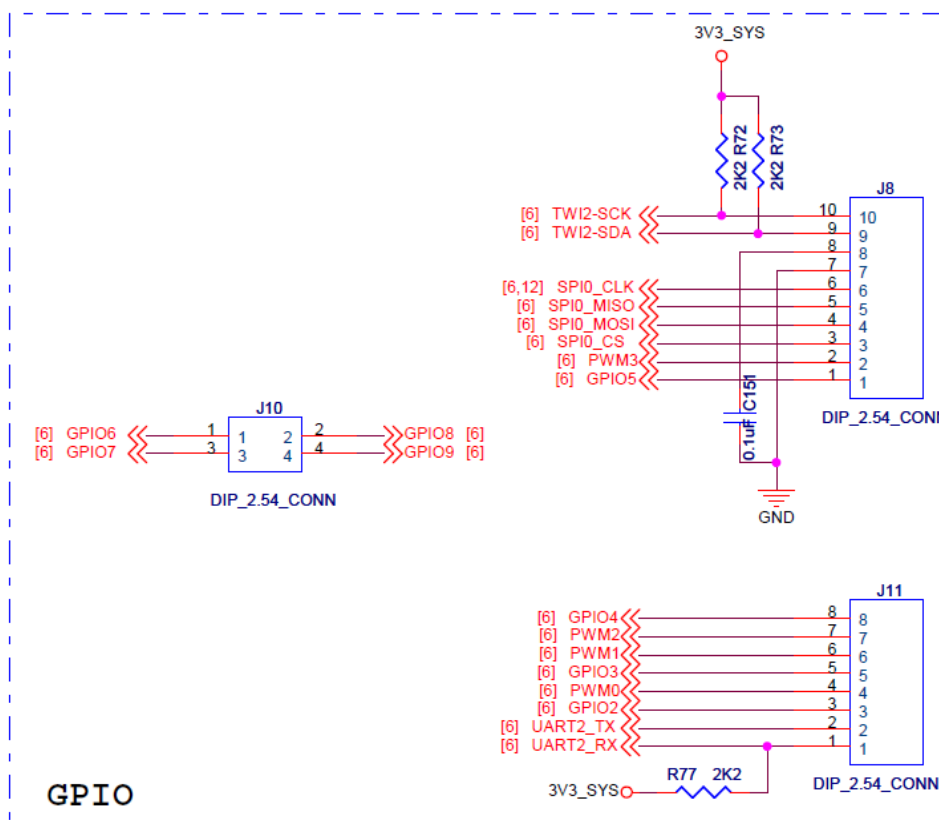
5 编写Linux 设备驱动控制LEDMatrix(在Win系统上实验)

1) 安装linux-headers

sudo apt-get install pcduino-linux-headers-3.4.29+

```
COM3 - PuTTY
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 10.180.32.1 0.0.0.0 UG 0 0 0 wlan4
default 192.168.100.1 0.0.0.0 UG 0 0 0 usb0
10.180.32.0 * 255.255.240.0 U 2 0 0 wlan4
192.168.100.0 * 255.255.255.0 U 1 0 0 usb0
root@ubuntu:~# sudo apt-get update && sudo apt-get install pcduino-linux-headers-3.4.29+
Ign http://ppa.launchpad.net precise InRelease
Ign http://ports.ubuntu.com precise InRelease
Ign http://ports.ubuntu.com precise-security InRelease
Ign http://ports.ubuntu.com precise-updates InRelease
Hit http://ppa.launchpad.net precise Release.gpg
Get:1 http://ports.ubuntu.com precise Release.gpg [198 B]
Hit http://ppa.launchpad.net precise Release
Get:2 http://ports.ubuntu.com precise-security Release.gpg [198 B]
Hit http://ppa.launchpad.net precise/main Sources
Hit http://ppa.launchpad.net precise/main armhf Packages
Ign http://ppa.launchpad.net precise/main TranslationIndex
Get:3 http://ports.ubuntu.com precise-updates Release.gpg [198 B]
Hit http://ports.ubuntu.com precise Release
Get:4 http://ports.ubuntu.com precise-security Release [49.6 kB]
Ign http://ppa.launchpad.net precise/main Translation-en
61k [4 Release 30.1 kB/49.6 kB 61k] [Connecting to www.wiimu.com (122.224.6.49)]
```

2) PCduino GPIO原理



3) 代码

GPIO寄存器

```
//定义与硬件相关的宏
//基地址
#define BASE_ADDRESS 0x01c20800
//PB_CFG 寄存器地址
#define PB_CFG0 (BASE_ADDRESS+0x24)
//PB_DAT
#define PB_DAT (BASE_ADDRESS+0x34)
//PH_CFG寄存器的地址
#define PH_CFG0 (BASE_ADDRESS+0xFC) //PH_CFG1
#define PH_CFG1 (BASE_ADDRESS+0x100) //PH_CFG2
//PH_DAT寄存器的地址
#define PH_DAT (BASE_ADDRESS+0x10C)
//PI_CFG
#define PI_CFG0 (BASE_ADDRESS+0x120)
#define PI_CFG1 (BASE_ADDRESS+0x124)
#define PI_CFG2 (BASE_ADDRESS+0x128)
//PI_DAT
#define PI_DAT (BASE_ADDRESS+0x130)
```

设备号

```
//申请设备号
err=alloc_chrdev_region(&dev_number,0,DEV_COUNT,DEV_NAME);
if(err){
    printk("alloc device number fail\n");
    return err;
}
//如果申请成功, 打印主设备号
printk("major number: %d\n",MAJOR(dev_number));
```

申请空间

```
//创建设备文件
device_create(classp,NULL,dev_number,"%s",DEV_NAME);
printk("/dev/%s create success\n",DEV_NAME);
//为ledmatrix_buffer分配空间
ledmatrix_buffer=(unsigned char*)kmalloc(LED_BUF_SIZE,GFP_KERNEL);
if(ledmatrix_buffer==NULL){
    printk("分配内存失败\n");
    return -1;
}
memset(ledmatrix_buffer,0,LED_BUF_SIZE);
```

GPIO输出

```
//设置ph7-ph5设置为输出
tmp=__ph_cfg0;
tmp&=0x000fffff;
tmp|=0x11100000;
*__ph_cfg0=tmp;
printk("__ph_cfg0:%ld\n",__ph_cfg0);
//!!!!!!设置ph14-ph8设置为输出
tmp=__ph_cfg1;
```

寄存器与端口的匹配

```
void ledmatrix_digitalwrite(int pin,int val){
    printk("pin:%d val:%d\n",pin,val);
    volatile unsigned long tmp;
    switch(pin){
    case 0://pi19
        tmp=__pi_dat;
        if(val==0){
            //置为0
            tmp&=~(1<<19);
            *__pi_dat=tmp;
        }
        else{
            //置为1
            tmp|=(1<<19);
            *__pi_dat=tmp;
        }
        break;
    }
```

显示

```
while(t>0){//循环刷新显示字符
    catstr__=ledmatrix_buffer[index];
    //printk("char: %c\n",char_show);
    // ledmatrix_setcharacter(char_show);
    bitmap=rebitmap(catstr__);
    int i;
    for(i=0;i<7;i++){
        ledmatrix_digitalwrite(row[i],HIGH);
    }
    for(i=0;i<7;i++){
        ledmatrix_digitalwrite(col[i],LOW);
    }
    ledmatrix_digitalwrite(16,0);
    ledmatrix_digitalwrite(15,1);
    int count=0;
    for(count=0;count<500;count++){
        ledmatrix_display();
    }
    index=index+1;
    if(index==length-1){
        index=0;
        t--;
    }
}
```

结果

