Assignment 008: Lab 8：网络 LED 矩阵显示器

陆鹏 3110101636

实验目的：

这个实验的目的是掌握编写 Linux 设备驱动程序，学习 Linux 下 I2C 总线的模拟实现方式，并以之实现一个网络访 问的 LED 矩阵显示器。

配合课程:

第八次：Linux 设备驱动程序

实验内容：

1. 掌握 Linux 设备驱动程序的开发过程；
2. 理解 I2C 总线协议；
3. 复习 socket 编程（网络原理课）；
4. 实现一个网络访问的 LED 矩阵显示器。

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）；

软件:

- 编译软件。

实验步骤:

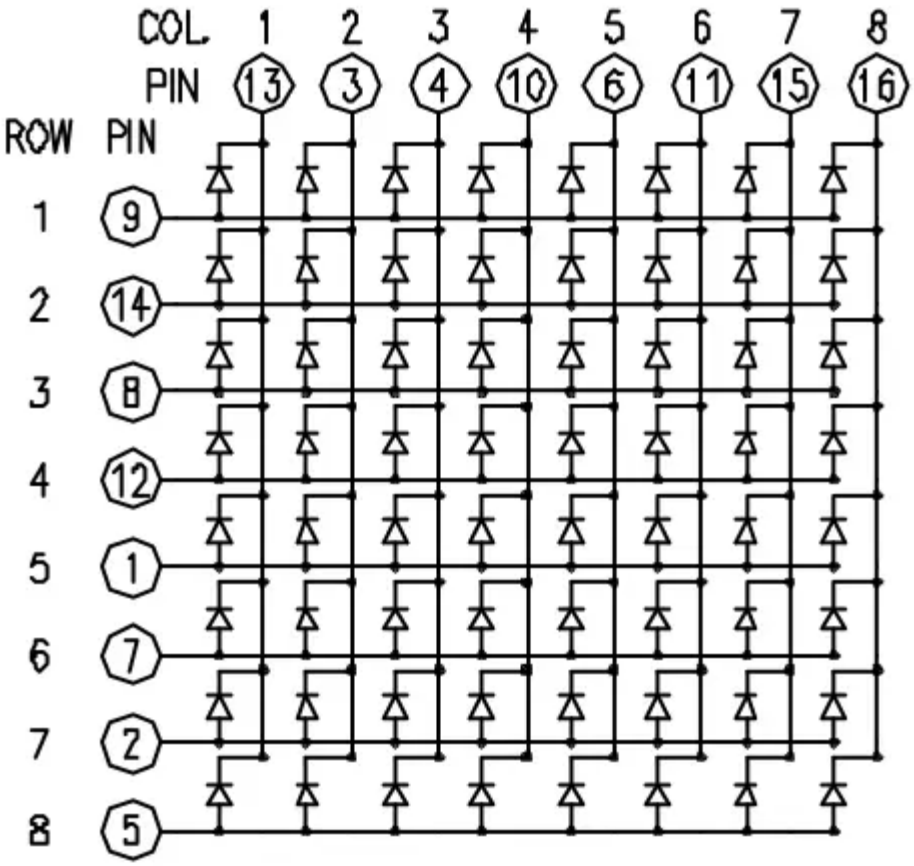
1. 设计外部设备方案，画连线示意图；
2. 在面包板上连线，完成外部电路；
3. 编写 Linux 应用程序，能通过第六次实验的 GPIO 库控制 LED 矩阵显示字母数字；
4. 编写 Linux 设备驱动程序，能通过寄存器操纵 GPIO 控制 LED 矩阵，将这个 LED 矩阵做成/dev/ledmatrix，之后能 通过 cat 命令输出字母数字来显示；
5. 编写 Linux 应用程序，能通过 TCP 接受一个连接，将发来的文字在 LED 矩阵上流动显示出来。

实验报告要求:

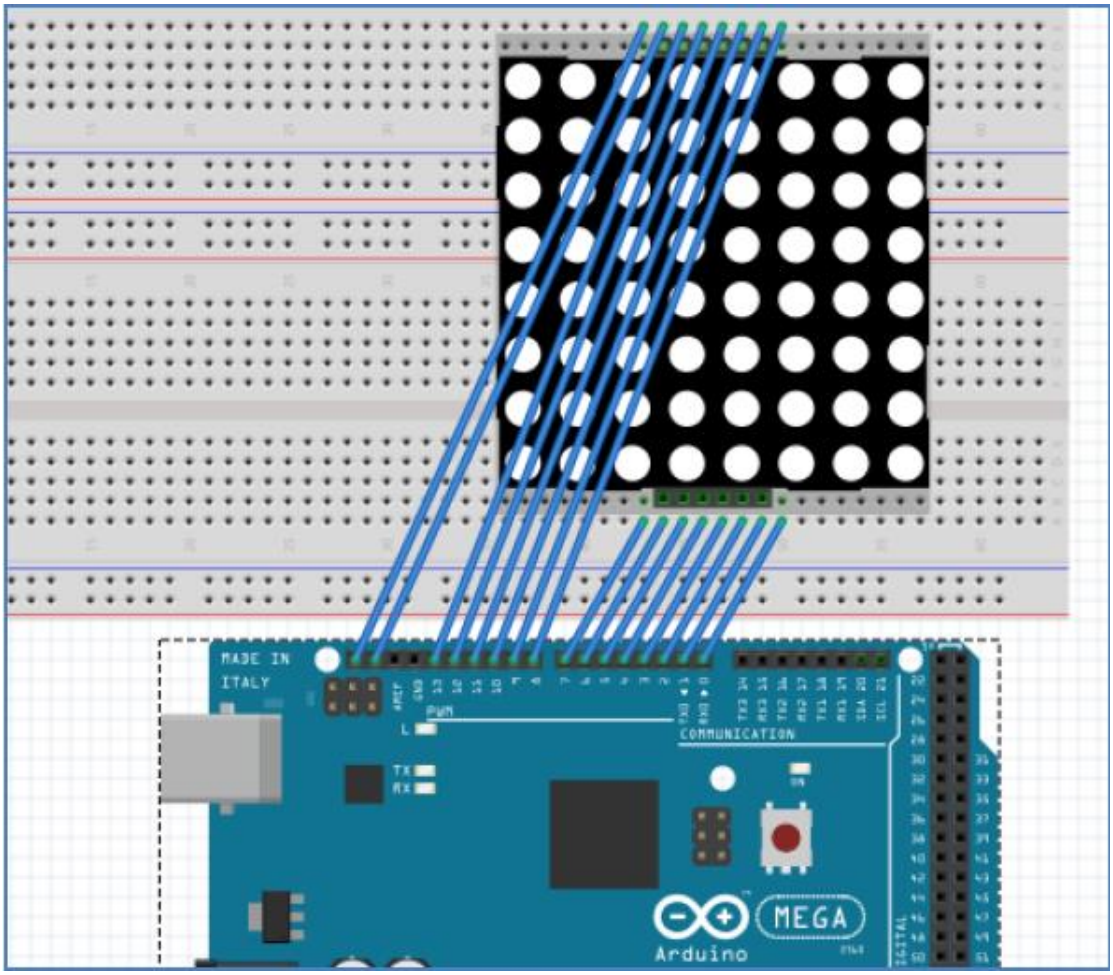
1. 画出你所实际实施的连接示意图；
2. 给出所用的器材的列表；
3. 用 Fritzing 画出外部设备的连线图，附实物照片；
4. 描述所做的实验步骤，给出各步操作的命令和结果；
5. 给出代码并解释；
6. 将所做作品拍摄视频上传到优酷，给出优酷的视频网址；
7. 说明其他所做的扩展内容的情况。

具体实验步骤：

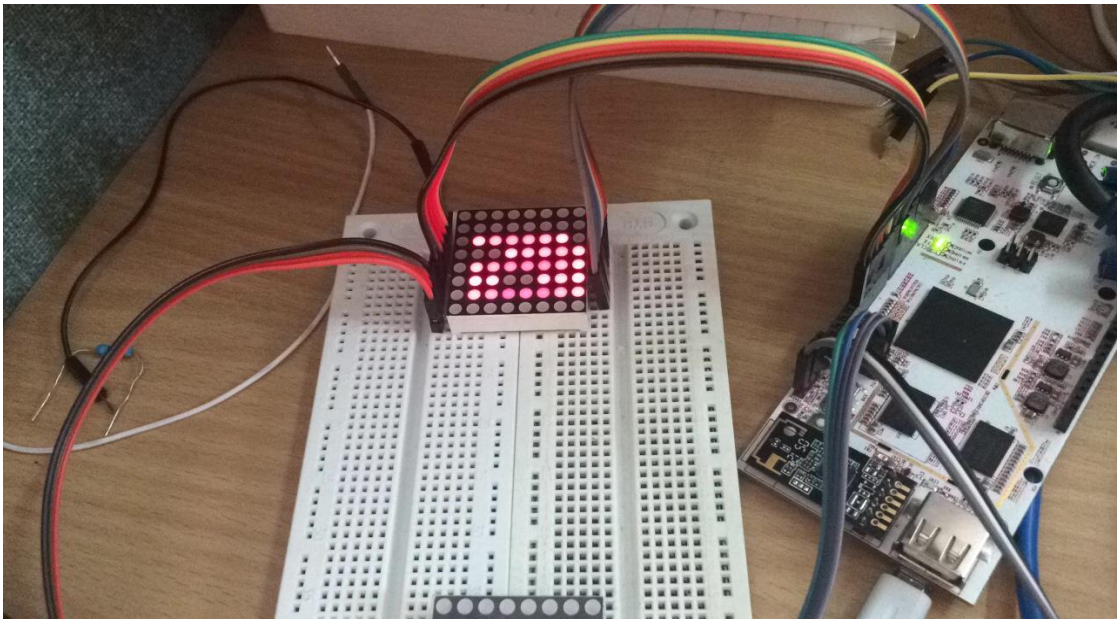
1. LedMatrix 原理图：



2. 连接图：
   a) Frizing 设计图：



b) 实际连接图：



3. GPIO 库控制 LEDMatrix：

a) 关键代码解释：

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};
```

字模定义——这一部分我已共享至群共享：
例如 a 字模：

```
byte zimu_a[] ={
  0b00000000,
  0b00000000,
  0b01111110,
  0b00011111,
  0b01111111,
  0b01100111,
  0b01111111,
  0b00000000,
};
```

通过 gets 函数获取键盘输入，并通过 rebitmap 函数返回对应字模：

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

Loop 中循环取字模：

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

分时复用使用 LedMatrix：

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

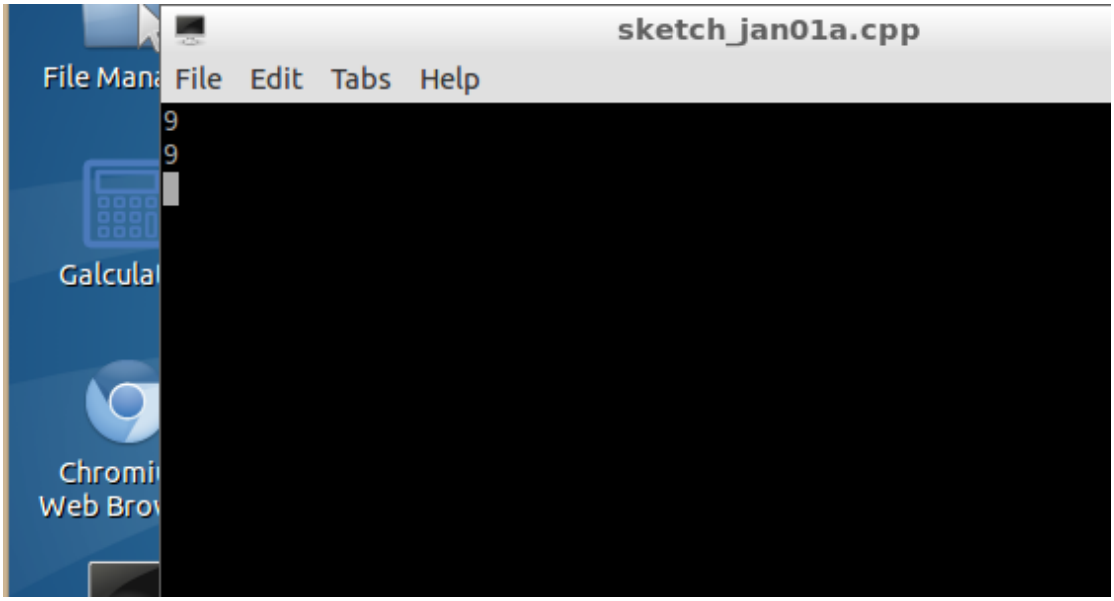
```
void pickcol(int colno){
  int i;
  clearrow();
  for(i=0;i<8;i++){
    digitalWrite(COL[i], HIGH);
  }
  digitalWrite(COL[colno], LOW);
}

void pickrow(int rowno,int l_h){
  digitalWrite(ROW[rowno],l_h);
}
```

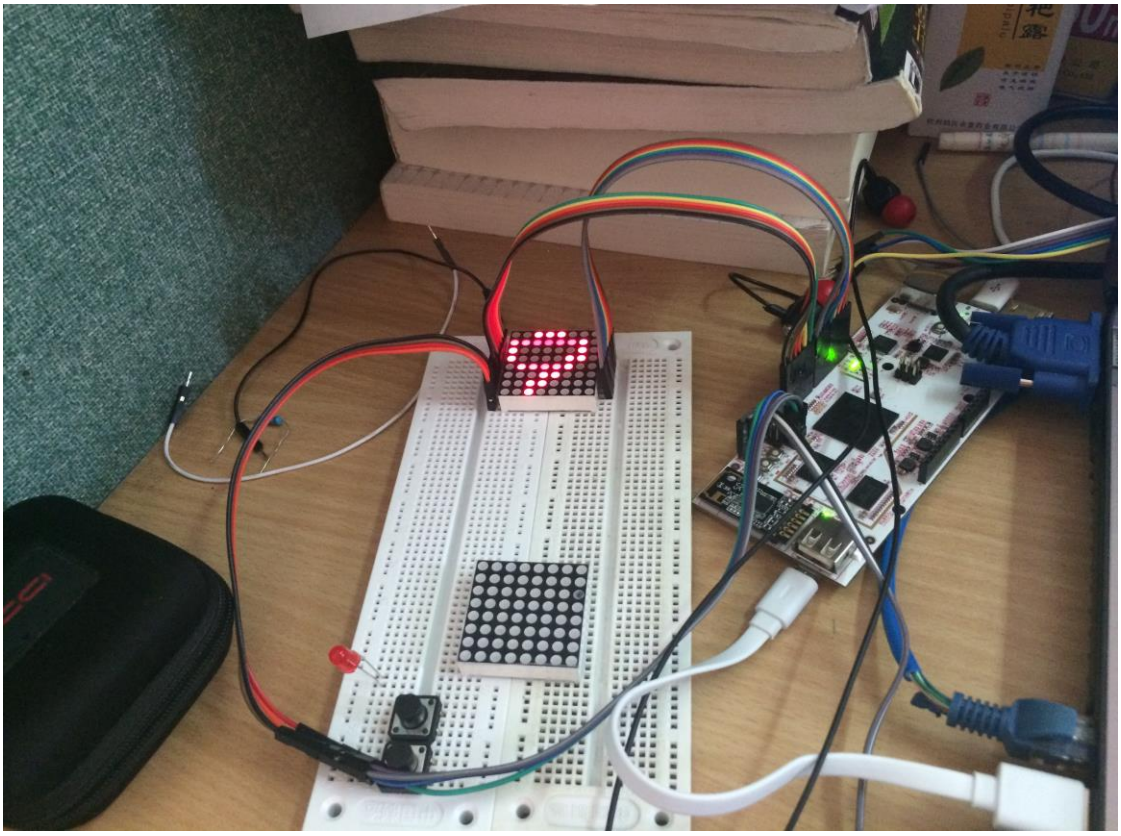
```
void clearrow(){
  int i;
  for (i = 0; i < 8; ++i)
  {
    /* code */
    digitalWrite(ROW[i],LOW);
  }
}
```

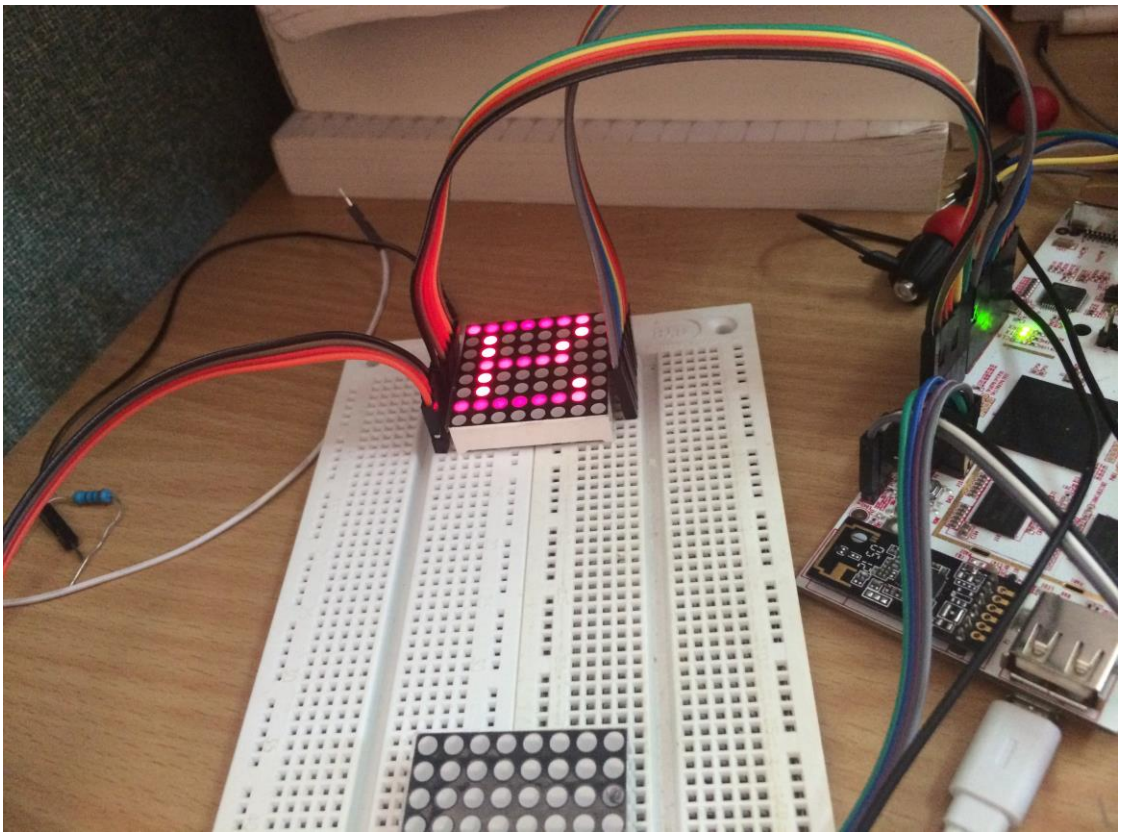
Rebitmap 函数：

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

b) 运行截图——显示 9：


sketch_jan01a.cpp
File Edit Tabs Help
9
9





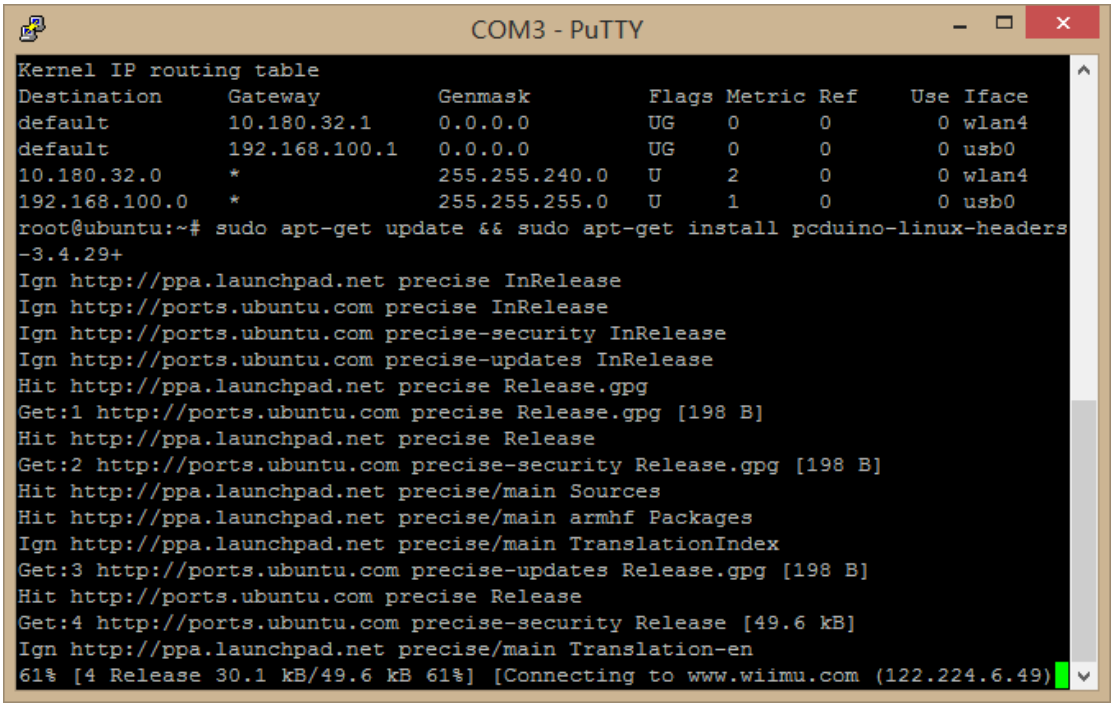
c) 完整代码请见附件 scr.zip 中的 “arduinoIDE 循环等待输入字符.c”:

4. 编写 Linux 设备驱动程序控制 LEDMatrix，通过 cat 命令输出字母数字来显示：

a) Linux 驱动开发测试：

i. 安装 linux-headers：

$ sudo apt-get update && 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
61% [4 Release 30.1 kB/49.6 kB 61%] [Connecting to www.wiimu.com (122.224.6.49)
```

ii. 编写一个测试驱动：参考自 http://www.oschina.net/question/1174645_122127 提供的教程

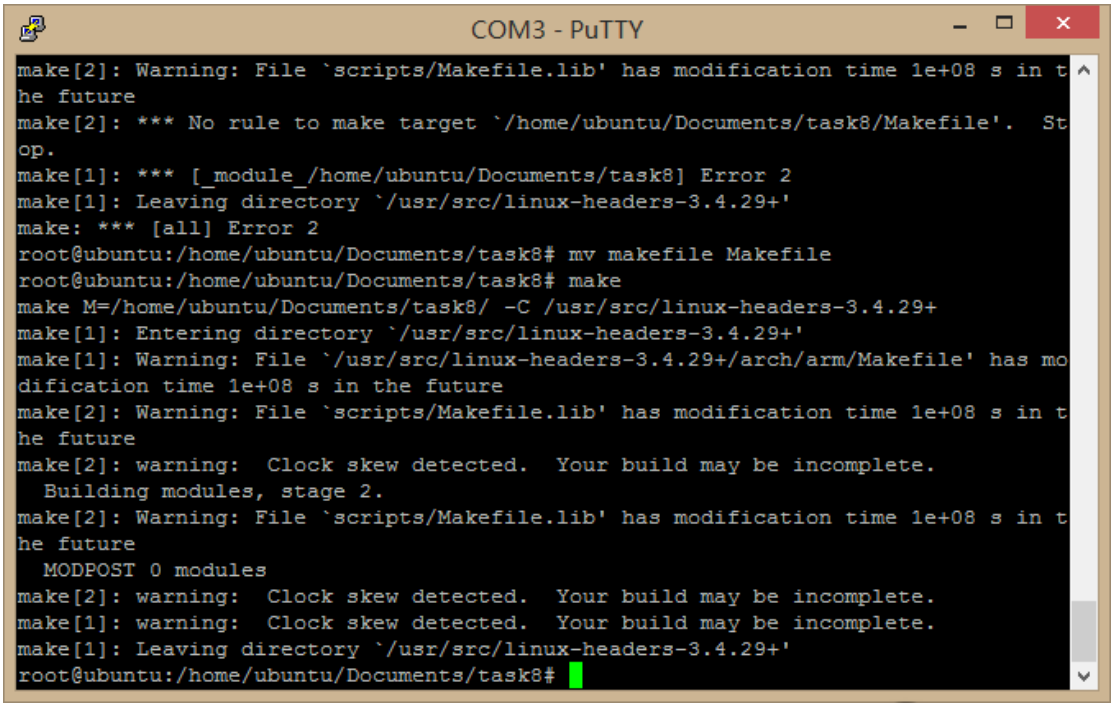
```
COM3 - PuTTY
        rm *.mod.c

root@ubuntu:/home/ubuntu/Documents/task8# cat drivecs.c

#include <linux/init.h>
#include <linux/module.h>

static int pcduino_hello_init(void)
{
    printk("Hello, pcDuino\n");
    return 0;
}

static void pcduino_hello_exit(void)
{
    printk("Bye, pcDuino\n");
}

MODULE_LICENSE("GPL");
MODULE_AUTHOR("pcDuino Team");
module_init(pcduino_hello_init);
module_exit(pcduino_hello_exit);

root@ubuntu:/home/ubuntu/Documents/task8#
```

iii. 编写 Makefile

```
COM3 - PuTTY
dification time 1e+08 s in the future
make[2]: Warning: File `scripts/Makefile.lib' has modification time 1e+08 s in t
he future
make[2]: warning:  Clock skew detected.  Your build may be incomplete.
  Building modules, stage 2.
make[2]: Warning: File `scripts/Makefile.lib' has modification time 1e+08 s in t
he future
  MODPOST 0 modules
make[2]: warning:  Clock skew detected.  Your build may be incomplete.
make[1]: warning:  Clock skew detected.  Your build may be incomplete.
make[1]: Leaving directory `/usr/src/linux-headers-3.4.29+'
root@ubuntu:/home/ubuntu/Documents/task8# cat Makefile

obj -m =hello.o
all:
        make M=/home/ubuntu/Documents/task8/ -C /usr/src/linux-headers-3.4.29+
clean:
        rm *.o
        rm *.ko
        rm *.order
        rm *.symvers
        rm *.mod.c

root@ubuntu:/home/ubuntu/Documents/task8#
```

iv. 编译并测试驱动程序

$make

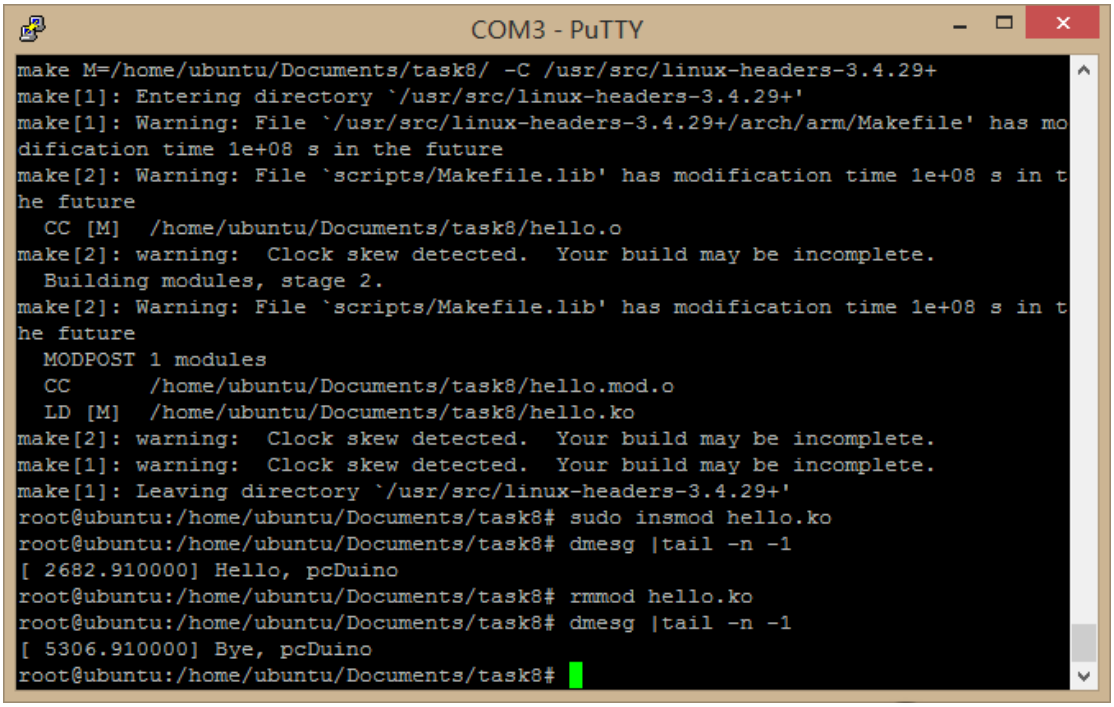


加载驱动

sudo insmod hello.ko

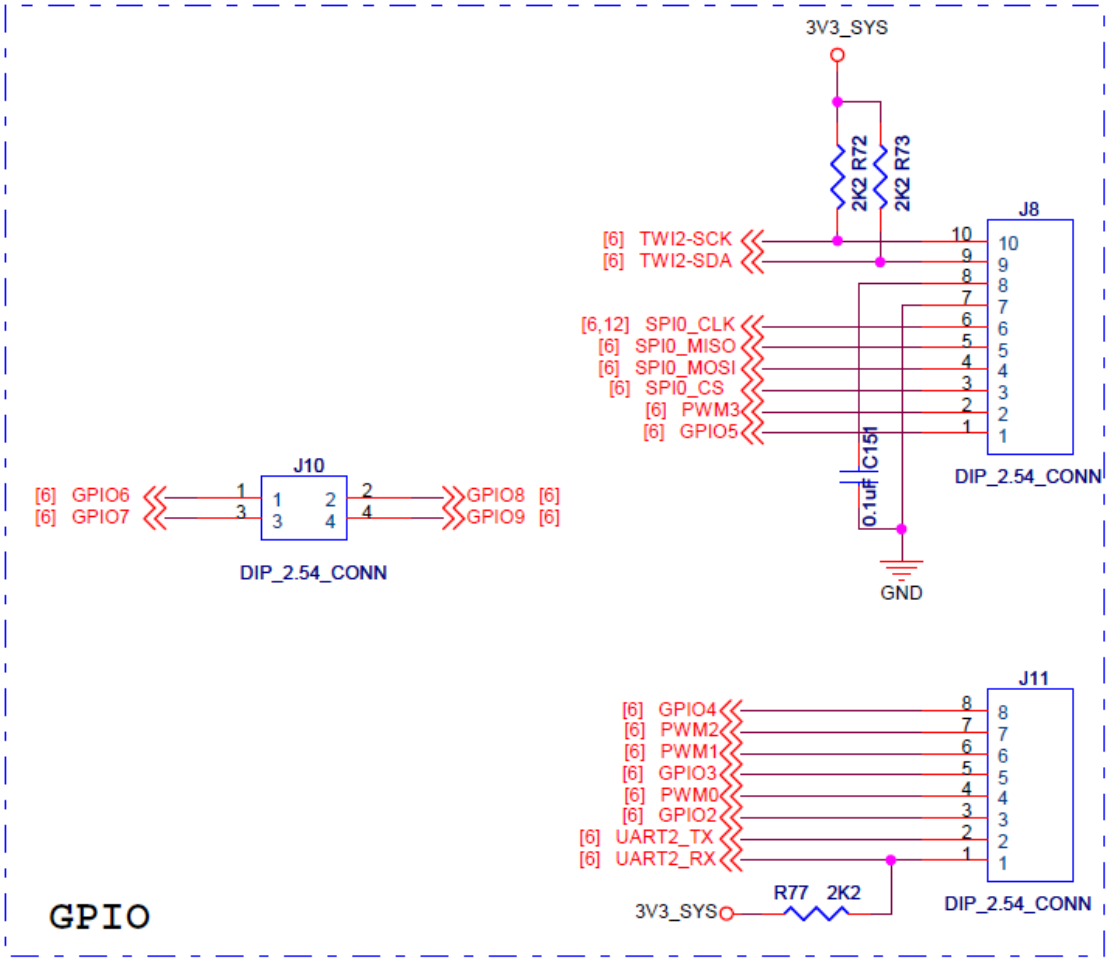
卸载驱动

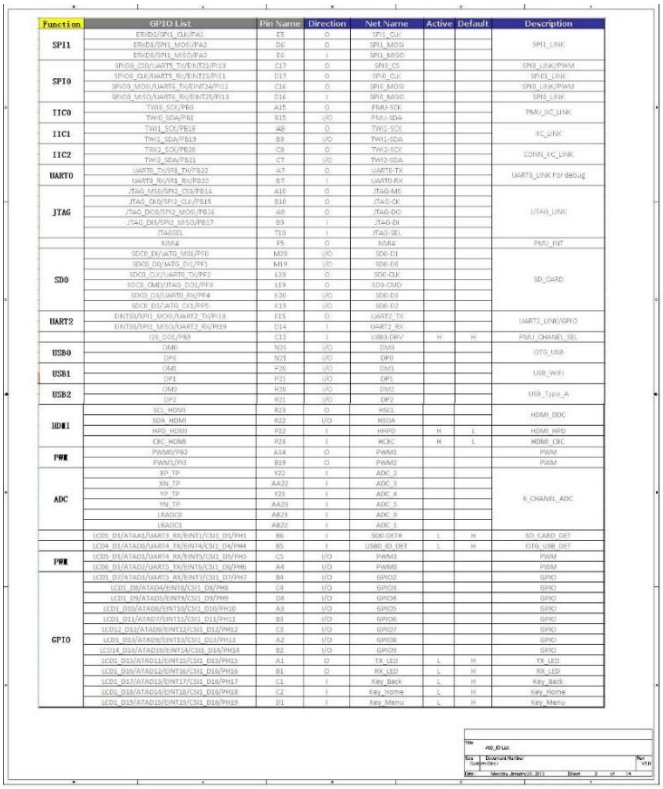
sudo rmmod hello.ko



b) PCduino GPIO 原理:

PCDuino GPIO：





Function	GPIO List	Pin Name	Direction	Net Name	Active	Default	Description
SPI1	ERxD2/SPI1_CLK/PA1	E5	O	SPI1_CLK			SPI1_LINK
	ERxD3/SPI1_MOSI/PA2	D6	O	SPI1_MOSI			
	ERxD0/SPI1_MISO/PA3	E6	I	SPI1_MISO			
SPI0	SPI0_CS0/UART5_Tx/EINT22/PI10	C17	O	SPI0_CS			SPI0_LINK/PWM
	SPI0_CLK/UART5_Rx/EINT23/PI11	D17	O	SPI0_CLK			SPI0_LINK
	SPI0_MOSI/UART6_Tx/EINT24/PI12	C16	O	SPI0_MOSI			SPI0_LINK/PWM
	SPI0_MISO/UART6_Rx/EINT25/PI13	D16	I	SPI0_MISO			SPI0_LINK
TIC0	TWI0_SCL/PB0	A15	O	PMU-SCL			PMU_I2C_LINK
	TWI0_SDA/PB1	B15	I/O	PMU-SDA			
TIC1	TWI1_SCL/PB18	A8	O	TWI1-SCL			I2C_LINK
	TWI1_SDA/PB19	B9	I/O	TWI1-SDA			
TIC2	TWI2_SCL/PB20	C9	O	TWI2-SCL			CONN_I2C_LINK
	TWI2_SDA/PB21	C7	I/O	TWI2-SDA			
UART0	UART0_Tx/PB22	A7	O	UART0-TX			UART3_LINK For debug
	UART0_Rx/PB23	B7	I	UART0-RX			
JTAG	JTAG_MS0/SPI2_CS3/PB14	A10	O	JTAG-MS			JTAG_LINK
	JTAG_CK0/SPI2_CLK/PB15	B10	O	JTAG-CK			
	JTAG_DO0/SPI2_MOSI/PB16	A9	O	JTAG-DO			
	JTAG_DI0/SPI2_MISO/PB17	B9	I	JTAG-DI			
	JTAGSEL	T10	I	JTAG-SEL			
	KM4	P5	O	KM4			PMU_INT
SD0	SDC0_D0/uATG_MS1/PF0	M19	I/O	SD0-D0			SD_CARD
	SDC0_D1/uATG_DI1/PF1	M19	I/O	SD0-D1			
	SDC0_CLK/UART0_TX/PF2	L19	O	SD0-CLK			
	SDC0_CMD/JTAG_DO1/PF3	L19	O	SD0-CMD			
	SDC0_D3/UART0_RX/PF4	K20	I/O	SD0-D3			
	SDC0_D2/uATG_CK1/PF5	K19	I/O	SD0-D2			
UART2	EINT19/SPI1_MOSI/UART2_TX/PI18	E15	O	UART2_TX			UART2_LINK/GPIO
	EINT20/SPI1_MISO/UART2_RX/PI19	D14	I	UART2_RX			
	USB_DO1/PB9	C13	I	USB9-DRV	H	H	PMU_CHANEL_SEL
USB0	DM0	N20	I/O	DM0			OTG_USB
	DP0	N23	I/O	DP0			
USB1	DM1	P20	I/O	DM1			USB_WIFI
	DP1	P21	I/O	DP1			
USB2	DM2	R20	I/O	DM2			USB_Type_A
	DP2	R21	I/O	DP2			
HDMI	SCL_HDMI	R19	O	HSCL			HDMI_DDC
	SDA_HDMI	R22	I/O	HSDA			
	HPD_HDMI	P22	I	HHPD	H	L	HDMI_HPD
	CEC_HDMI	R23	I	HCEC	H	L	HDMI_CEC
PWM	PWM0/PB2	A14	O	PWM1			PWM
	PWM1/PB3	B19	O	PWM2			PWM
ADC	XP_TP	Y22	I	ADC_2			6_CHANEL_ADC
	XN_TP	AA22	I	ADC_3			
	YP_TP	Y23	I	ADC_4			
	YN_TP	AA23	I	ADC_5			
	LRADC0	AB23	I	ADC_0			
	LRADC1	AB22	I	ADC_1			
	LCD1_D1/ATAD1/UART3_RX/EINT1/CSI1_D1/PH1	B6	I	SD0-DETN	L	H	SD_CARD_DET
	LCD4_D1/ATAD0/UART4_TX/EINT4/CSI1_D4/PH4	B5	I	USB0_ID_DET	L	H	OTG_USB_DET
PWM	LCD5_D1/ATAD1/UART4_RX/EINT5/CSI1_D5/PH5	C5	I/O	PWM5			PWM
	LCD6_D1/ATAD2/UART5_TX/EINT6/CSI1_D6/PH6	A4	I/O	PWM6			PWM
GPIO	LCD1_D7/ATAD8/UART5_RX/EINT7/CSI1_D7/PH7	B4	I/O	GPIO2			GPIO
	LCD1_D8/ATAD4/EINT8/CSI1_D8/PH8	C4	I/O	GPIO3			GPIO
	LCD1_D9/ATAD5/EINT9/CSI1_D9/PH9	D4	I/O	GPIO4			GPIO
	LCD1_D10/ATAD6/EINT10/CSI1_D10/PH10	A3	I/O	GPIO5			GPIO
	LCD1_D11/ATAD7/EINT11/CSI1_D11/PH11	B3	I/O	GPIO6			GPIO
	LCD12_D12/ATAD8/EINT12/CSI1_D12/PH12	C3	I/O	GPIO7			GPIO
	LCD1_D13/ATAD9/EINT13/CSI1_D13/PH13	A2	I/O	GPIO8			GPIO
	LCD14_D14/ATAD10/EINT14/CSI1_D14/PH14	B2	I/O	GPIO9			GPIO
	LCD1_D15/ATAD11/EINT15/CSI1_D15/PH15	A1	O	TX_LED	L	H	TX_LED
	LCD1_D16/ATAD12/EINT16/CSI1_D16/PH16	B1	O	RX_LED	L	H	RX_LED
	LCD1_D17/ATAD13/EINT17/CSI1_D16/PH17	C1	I	Key_Back	L	H	Key_Back
	LCD1_D18/ATAD14/EINT18/CSI1_D16/PH18	C2	I	Key_Home	L	H	Key_Home
	LCD1_D19/ATAD15/EINT19/CSI1_D16/PH19	D1	I	Key_Menu	L	H	Key_Menu

查找 A10 的开发手册，得到每个 GPIO 接口的寄存分别为

GPIO0	PI19
GPIO1	PI18
GPIO2	PH7
GPIO3	PH6
GPIO4	PH8
GPIO5	PB2
GPIO6	PI3
GPIO7	PH9
GPIO8	PH10
GPIO9	PH5
GPIO10	PI10
GPIO11	PI12
GPIO12	PI13
GPIO13	PI11
GPIO14	PH11
GPIO15	PH12
GPIO16	PH13
GPIO17	PH14

GPIO 基地址：

Module Name	Base Address
PIO	0x01C20800

c) 关键代码解释：

GPIO 寄存器地址宏定义：

```
//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
#define PH_CFG1   (BASE_ADDRESS+0x100)    //PH
//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)
```

申请设备号：

```
//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));
```

创建设备与申请空间：

```
//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);
```

清空缓存与映射内存：

```
//首先清空缓存
memset(ledmatrix_buffer,0,LED_BUF_SIZE);
//将PH_CFG0-2这个硬件寄存器的地址，映射到linux内存，并获取映射后的地址
//通过对这个地址的操作，就可以控制PH_CFG0-2
__ph_cfg0 = (volatile unsigned long*)ioremap(PH_CFG0,4);
__ph_cfg1 = (volatile unsigned long*)ioremap(PH_CFG1,4);
__ph_dat=(volatile unsigned long*)ioremap(PH_DAT,4);

__pi_cfg0=(volatile unsigned long*)ioremap(PI_CFG0,4);
__pi_cfg1=(volatile unsigned long*)ioremap(PI_CFG1,4);
__pi_cfg2=(volatile unsigned long*)ioremap(PI_CFG2,4);
__pi_dat=(volatile unsigned long*)ioremap(PI_DAT,4);

__pb_cfg0=(volatile unsigned long*)ioremap(PB_CFG0,4);
__pb_dat=(volatile unsigned long*)ioremap(PB_DAT,4);
```

设置 GPIO 为输出：

```
    //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;
```

将寄存器与 PCDuino GPIO 端口匹配：

```
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;
```

ledmatrix_display：

```
void ledmatrix_display(){
  int i;
  int j;
    for(i=0;i<8;i++){
        pickcol(i);
        for (j = 0; j < 8; ++j)
        {
          /* code */
          // int rowp=(bitmap[i] >> j) & 0b00000001;
          pickrow(j,(bitmap[i] >> j) & 0b00000001);
          // pickrow(j,1);
          // delay(del);
        }
      }
}
```

：

循环显示：

```
while(t>0){//循环刷新显示字符
    catstr__=ledmatrix_buffer[index];
    //printk("char: %c\n",char_show);
    // ledmatrix_setcharater(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--;
    }
```

d) 实际运行：

已经录制成视频上传至：

http://v.youku.com/v_show/id_XNzI5Mjg0NzE2.html

e) 完整代码请见 src.zip 中的 ledmatrix.c：

5. 编写 Linux 应用程序通过 TCP 接受一个连接，将发来的文字在 LED 矩阵上流动显示出来。

a) 编写 TCP Server：

i. 打开驱动对应文件：

```
//打开驱动对应的设备文件
fd=open("/dev/ledmatrix",O_RDWR);
if(fd<0){
  printf("open /dev/ledmatrix error\n");
  return -1;
}
```

写入接收到的字符：

```
iDataNum=recv(new_fd,buffer,4096,0);
if(iDataNum<0)
{
    perror("Recv\n");
    exit(1);
}
printf("buffer:%s\n",buffer);
write(fd,buffer,sizeof(buffer));
```

b) 编写 TCP Client：

i. 创建连接：

```
if(connect(sockfd,(struct sockaddr *)(&server_addr),sizeof(struct sockaddr))==-1)
{
 fprintf(stderr,"Connect error:%s\n",strerror(errno));
 exit(1);
}
while(1)
```

ii. 读入字符串，并发送到 Server：

```
while(1)
{
        char sendbuffer[200];
        printf("Please input your word:\n");
        scanf("%s",sendbuffer);
        printf("\n");
        if(strcmp(sendbuffer,"quit")==0)
            break;
        send(sockfd,sendbuffer,sizeof(sendbuffer),0);
        recv(sockfd,recvbuffer,200,0);
        printf("recv data of my world is :%s\n",recvbuffer);
}
```

c) 实际运行：

已经录制成视频上传至：

http://v.youku.com/v_show/id_XNzI5MzIyNTcy.html

d) 完整代码请见 src.c 中的 Server.c