

# 浙江大学

## 本科实验报告

课程名称: 嵌入式系统

姓 名: Jessie Peng

学 院: 计算机科学与技术学院

系:

专 业:

学 号: 3160104765

指导教师: 王总辉

2019 年 6 月 23 日

# 浙江大学实验报告

课程名称: 嵌入式系统 实验类型: 综合

实验项目名称: 实验 7: 字符设备驱动程序

学生姓名: Jessie Peng 专业: \_\_\_\_\_ 学号: \_\_\_\_\_

实验地点: 曹西 501 实验日期: 2019 年 6 月 23 日 指导老师: 王总辉

## 一、实验目的和要求

- 了解嵌入式 Linux 的 GPIO 的使用方式
- 了解嵌入式 Linux 的 Arduino 接口库
- 掌握使用面包板搭简单的外部电路
- 掌握 Linux 设备驱动程序的开发过程
- 掌握在内核中访问外设寄存器，操纵外设的方法

## 二、实验内容和原理

### 【实验器材——硬件】

- 树莓派 3B+实验板
- 5V/1A 电源
- microUSB 线
- 面包板
- 8x8 LED 矩阵
- MAX7219
- 360 欧，1/8 瓦电阻
- 面包线若干

### 【实验器材——软件】

- PC 上的 SSH 软件

- 交叉编译软件 arm-linux-gcc
  - 画图工具 Fritzing
  - 实验 7 步骤

**【实验内容——调用 Arduino 库，实现 LED 显示文字功能】**

- 设计硬件方案，画连线示意图
  - 在面包板上连线，完成外部电路
  - 编写 C/C++ 程序，采用 Arduino-ish 库访问 GPIO，实现在矩阵上显示文字或图案

【实验内容——调用内核驱动程序，实现 LED 显示文字功能】

- 编写字符设备驱动程序的内核模块，直接访问 GPIO 控制寄存器
  - 编写 C 语言程序，调用 write() 送来的单个字符在矩阵上显示出来

### 三、实验步骤和结果记录

## 【使用 wiringPi 驱动点阵】

使用命令 `git clone git://git.drogon.net/wiringPi` 下载 wiringPi 库，然后进入 wiringPi 目录使用命令 `./build` 运行脚本进行编译和安装。安装完后用 `gpio` 命令进行测试：

```

pi@raspberrypi:~/Lab7/wiringPi $ gpio -v
gpio version: 2.50
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

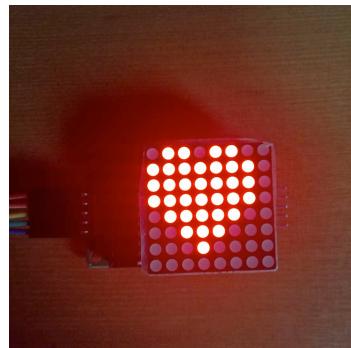
Raspberry Pi Details:
 Type: Pi 3B+, Revision: 03, Memory: 1024MB, Maker: Sony
 * Device tree is enabled.
 => Raspberry Pi 3 Model B Plus Rev 1.3
 * This Raspberry Pi supports user-level GPIO access.
pi@raspberrypi:~/Lab7/wiringPi $ gpio readall
+-----Pi 3B+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+
|      | 3.3v |       |      |   |          |   |      |      |      |      | |
| 2    | 8    | SDA.1 | IN  | 1 | 3          | 4 |      |      | 5v  |      |
| 3    | 9    | SCL.1 | IN  | 1 | 5          | 6 |      |      | 5v  |      |
| 4    | 7    | GPIO.7 | IN  | 1 | 7          | 8 | 1    | ALT5 | TxD | 15 | 14 |
|      |      | 0v    |      |   |          |   |      |      |      |      |
| 17   | 0    | GPIO.0 | IN  | 0 | 11         | 12 | 0    | IN   | GPIO.1 | 1  | 18 |
| 27   | 2    | GPIO.2 | IN  | 0 | 13         | 14 |      |      | 0v  |      |
| 22   | 3    | GPIO.3 | IN  | 0 | 15         | 16 | 0    | IN   | GPIO.4 | 4  | 23 |
|      |      | 3.3v |       |      |          |   |      |      |      |      |
| 10   | 12   | MOSI  | IN  | 0 | 19         | 20 |      |      | 0v  |      |
| 9    | 13   | MISO  | IN  | 0 | 21         | 22 | 0    | IN   | GPIO.6 | 6  | 25 |
| 11   | 14   | SCLK  | IN  | 0 | 23         | 24 | 1    | IN   | CE0  | 10 | 8  |
|      |      | 0v    |       |      |          |   |      |      |      |      |
| 0    | 30   | SDA.0 | IN  | 1 | 27         | 28 | 1    | IN   | SCL.0 | 31 | 1  |
| 5    | 21   | GPIO.21 | IN | 1 | 29         | 30 |      |      | 0v  |      |
| 6    | 22   | GPIO.22 | IN | 1 | 31         | 32 | 0    | IN   | GPIO.26 | 26 | 12 |
| 13   | 23   | GPIO.23 | IN | 0 | 33         | 34 |      |      | 0v  |      |
| 19   | 24   | GPIO.24 | IN | 0 | 35         | 36 | 0    | IN   | GPIO.27 | 27 | 16 |
| 26   | 25   | GPIO.25 | IN | 0 | 37         | 38 | 0    | IN   | GPIO.28 | 28 | 20 |
|      |      | 0v    |       |      |          |   |      |      |      |      |
+-----Pi 3B+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+
pi@raspberrypi:~/Lab7/wiringPi $ 
```

编写程序 matrix.c，按照 ASCII 码顺序每 500ms 显示一个字符。

```
1. #include <stdio.h>
2. #include <wiringPi.h>
3.
4. #define PIN_DIN 0
5. #define PIN_CLK 1
6. #define PIN_CS 2
7.
8. unsigned char disp1[256][8] = { //ASCII
9. ...
10. };
11.
12. void write_byte(unsigned char data)
13. {
14.     unsigned char i;
15.     digitalWrite(PIN_CS, LOW);
16.     for (i = 8; i >= 1; i--)
17.     {
18.         digitalWrite(PIN_CLK, LOW);
19.         digitalWrite(PIN_DIN, data&0x80);
20.         data = data << 1;
21.         digitalWrite(PIN_CLK, HIGH);
22.     }
23. }
24.
25. void write(unsigned char addr, unsigned char data)
26. {
27.     digitalWrite(PIN_CS, LOW);
28.     write_byte(addr);
29.     write_byte(data);
30.     digitalWrite(PIN_CS, HIGH);
31. }
32.
33. void init()
34. {
35.     write(0x09, 0x00);
36.     write(0x0a, 0x03);
37.     write(0x0b, 0x07);
38.     write(0x0c, 0x01);
39.     write(0x0f, 0x00);
40. }
41.
42. int main()
```

```
43. {
44.     unsigned char i, j;
45.
46.     wiringPiSetup();
47.     pinMode(PIN_DIN, OUTPUT);
48.     pinMode(PIN_CS, OUTPUT);
49.     pinMode(PIN_CLK, OUTPUT);
50.
51.     init();
52.     while(1)
53.     {
54.         for (j = 0; j < 256; j++)
55.         {
56.             for (i = 1; i < 9; i++)
57.             {
58.                 write(i, disp1[j][i-1]);
59.             }
60.             printf("0x%02X\n", j);
61.             delay(500);
62.         }
63.     }
64.     return 0;
65. }
```

使用 gcc matrix.c -lwiringPi 编译，运行效果如下：



### 【编写内核模块驱动点阵】

编写 driver.c 并在 Ubuntu 上交叉编译成内核模块，该模块加载时会在/dev/目录下生成一个 matrix 文件，向该文件写入字符则触发 matrix\_write 函数。

```
1. #define BUFFERSIZE 128
2. #define DELAYTIME 1
```

```
3. unsigned char disp[BUFFERSIZE];
4. int head = 0, tail = 0;
5. static struct timer_list timer;
6.
7. void Matrix_next_display(unsigned long);
8.
9. void ptr_inc(int *ptr){
10.    *ptr = (*ptr + 1) % BUFFERSIZE;
11. }
12.
13. static void timer_register(struct timer_list* ptimer){
14.    init_timer(ptimer);
15.    ptimer->data = DELAYTIME;
16.    ptimer->expires = jiffies + (DELAYTIME * HZ);
17.    ptimer->function = Matrix_next_display;
18.    add_timer(ptimer);
19. }
20.
21. void disp_start(void){
22.    timer_register(&timer);
23. }
24.
25. void Matrix_next_display(unsigned long data){
26.    if (head != tail){
27.        unsigned char *ptr = Pi;
28.        unsigned char c = disp[head];
29.        if ('0' <= c && c <= '9'){
30.            ptr = digits[c - '0'];
31.        }
32.        Matrix_render(ptr);
33.        ptr_inc(&head);
34.        disp_start();
35.    }else{
36.        Matrix_clear();
37.    }
38. }
39.
40. static int matrix_write(struct file *file, const char __user *buffer,
41.                        size_t count, loff_t *ppos){
42.    int i;
43.    if (head == tail && count > 0){
44.        disp_start();
45.    }
46.    for (i=0; i<count; i++){
```

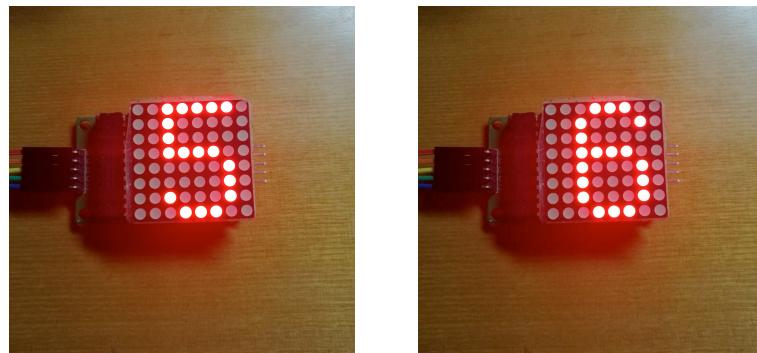
```

47.         ptr_inc(&tail);
48.         if (tail == head)
49.             ptr_inc(&head);
50.         disp[tail] = buffer[i];
51.     }
52.     return count;
53. }
54.
55. static struct file_operations matrix_fops = {
56.     .owner = THIS_MODULE,
57.     .write = matrix_write,
58.     .llseek = noop_llseek
59. };
60.
61. static struct miscdevice matrix_misc_device = {
62.     .minor = MISC_DYNAMIC_MINOR,
63.     .name = "matrix",
64.     .fops = &matrix_fops
65. };
66.
67. static int __init matrix_init(void){
68.     if (!gpio_is_valid(DIN) || !gpio_is_valid(CLK) || !gpio_is_valid(CS)){
69.         printk(KERN_INFO "GPIO_TEST: invalid GPIO\n");
70.         return -ENODEV;
71.     }
72.     misc_register(&matrix_misc_device);
73.     gpio_request(DIN, "sysfs");
74.     gpio_direction_output(DIN, 0);
75.     gpio_request(CS, "sysfs");
76.     gpio_direction_output(CS, 1);
77.     gpio_request(CLK, "sysfs");
78.     gpio_direction_output(CLK, 0);
79.     Matrix_init();
80.     printk(KERN_INFO "matrix device has been registered.\n");
81.     return 0;
82. }
83.
84. static void __exit matrix_exit(void){
85.     misc_deregister(&matrix_misc_device);
86.     printk(KERN_INFO "matrix device has been unregistered.\n");
87.     del_timer(&timer);
88. }
```

加载内核模块后用如下命令进行测试:

```
1. cd /dev  
2. sudo chown pi:pi matrix  
3. echo 0123456789 > matrix
```

效果如下：

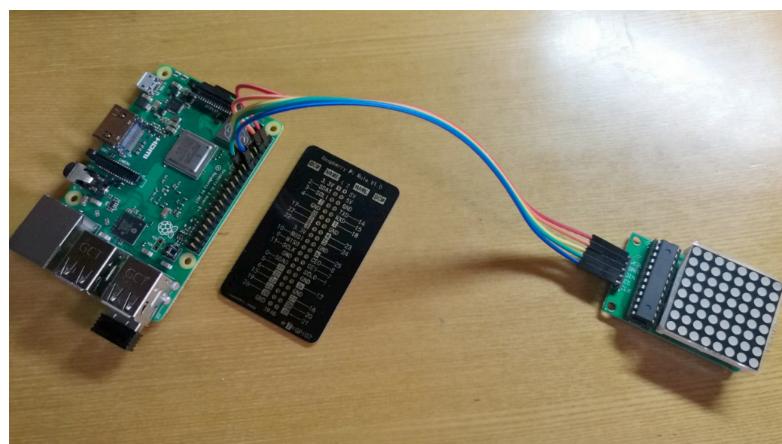


#### 四、实验结果分析

本次实验连线说明：

树莓派	MAX7219 驱动的点阵
<b>5V</b>	VCC
<b>GND</b>	GND
<b>GPIO 0 (BCM 17)</b>	DIN
<b>GPIO 2 (BCM 27)</b>	CS
<b>GPIO 1 (BCM 18)</b>	CLK

实物连线图如下：



## 五、讨论与心得

本次实验在上一次的基础上，已经熟练嵌入式 Linux 内核模块的编译和加载，继续学习了树莓派 GPIO 的操作。LED 点阵的驱动原理很好理解，总体来说实验完成得比较顺利。