

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:

指导教师: 王总辉

2019 年 6 月 24 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 8: 网络 LED 矩阵显示器

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 24 日 指导老师: 王总辉

一、实验目的和要求

- 了解 TCP/IP 原理
- 了解 Telnet 协议工作原理
- 掌握 TCP 网络应用程序设计
- 掌握嵌入式数据应用的设计方法

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
- PC 上的 USB-TTL 串口线配套的驱动程序

- PC 上的串口终端软件 screen
- 交叉编译软件 arm-linux-gcc

【实验内容——编写网络应用，接收 Telnet 客户端发送的消息，并实现 LED 显示文字功能】

- 在实验 7 的基础上，编写网络 TCP 服务端程序，接收到 Telnet 客户端发送过来的字符串后，调用驱动程序将字符串在矩阵上显示出来，每个字符停留 500ms

【实验内容——将通信日志写到本地 sqlite 中】

- 使用 sqlite 数据库记录 TCP 服务端的网络事件日志
- 设计网络事件表结构，字段包含序号（自增）、时间、对端 IP、对端端口、本地 IP、本地端口、事件和内容
 - 事件包括：侦听、连接、发送、接收、关闭等
 - 内容，针对发送或接收的具体内容
- 在宿主机用 Telnet 连接到目标板的服务端，输入自己的学号和名字，查看 LED 显示结果
- 打开 SQLite 数据库，查看网络事件表的内容

三、实验步骤和结果记录

【编写服务端程序 server2.c】

使用 C 语言 socket 编程实现 TCP 通信，使用 SQLite C/C++ API 嵌入 SQL 命令，以建立数据库和网络事件表，并在通信事件发生后向数据库中插入记录。

引用头文件，定义服务端 IP 和口号，声明数据库操作需要的全局变量：

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4. #include <unistd.h>
5. #include <sys/types.h>
6. #include <sys/stat.h>
7. #include <fcntl.h>
8. #include <netinet/in.h>
9. #include <arpa/inet.h>
10. #include <sys/socket.h>
```

```
11. #include <sqlite3.h>
12. #include <time.h>
13.
14. #define PORT 8080
15. #define ADDR "0.0.0.0"
16. #define QUEUE 20
17.
18. #define BUFF_SIZE 2048
19.
20. sqlite3 *db;
21. char *zErrMsg = 0;
22. int rc;
23. char *sql;
24.
25. time_t t;
```

定义初始化日志的函数，该函数在主函数最开始被调用，作用是新建一个本地数据库 test.db，并新建一张表 log，表的字段包含 id, time, client_ip, client_port, server_ip, server_port, event, detail:

```
1. int init_log()
2. {
3.     rc = sqlite3_open("test.db", &db);
4.     if (rc)
5.     {
6.         printf("Can't open database: %s\n", sqlite3_errmsg(db));
7.         return -1;
8.     }
9.
10.    printf("Opened database successfully\n");
11.
12.    char *sql = "CREATE TABLE LOG( "
13.            "ID INTEGER PRIMARY KEY AUTOINCREMENT, "
14.            "TIME      TEXT      NOT NULL, "
15.            "CLIENT_IP  TEXT, "
16.            "CLIENT_PORT   INT, "
17.            "SERVER_IP  TEXT, "
18.            "SERVER_PORT   INT, "
19.            "EVENT      TEXT      NOT NULL, "
20.            "DETAIL      TEXT);";
21.
22.    rc = sqlite3_exec(db, sql, callback, 0, &zErrMsg);
23.    if (rc != SQLITE_OK)
```

```
24.     {
25.         printf("SQL error: %s\n", zErrMsg);
26.         sqlite3_free(zErrMsg);
27.     }
28.     else
29.     {
30.         printf("Table created successfully\n");
31.     }
32.
33.     return 0;
34. }
```

定义写入新记录的函数，该函数封装了执行 SQL 命令及处理错误的语句：

```
1. int write_log()
2. {
3.     rc = sqlite3_exec(db, sql, callback, 0, &zErrMsg);
4.     if (rc != SQLITE_OK)
5.     {
6.         printf("SQL error: %s\n", zErrMsg);
7.         sqlite3_free(zErrMsg);
8.         return -1;
9.     }
10.    return 0;
11. }
```

上面的 `sqlite3_exec()` 就是执行 SQL 命令的接口，它接收的参数包含一个回调函数，该函数定义如下：

```
1. static int callback(void *NotUsed, int argc, char **argv, char **azColName)
2. {
3.     int i;
4.     for (i = 0; i < argc; i++)
5.     {
6.         printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
7.     }
8.     printf("\n");
9.     return 0;
10. }
```

主函数按照 TCP 的流程与客户机通信，此外还分别在监听端口、建立连接、接收数据和关闭连接的时候向日志中插入新的记录：

```
1. int main()
2. {
3.     if(init_log() == -1)
4.     {
5.         return -1;
6.     }
7.
8.     int fd = open("/dev/matrix", O_WRONLY | O_NONBLOCK);
9.     if (fd < 0)
10.    {
11.        printf("ERROE: Could not open matrix device!\n");
12.        return -1;
13.    }
14.
15.    // establish server
16.    int server = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
17.
18.    struct sockaddr_in serverAddr;
19.    serverAddr.sin_family = AF_INET;
20.    serverAddr.sin_addr.s_addr = inet_addr(ADDR);
21.    serverAddr.sin_port = htons(PORT);
22.
23.    // bind ip and port
24.    if (bind(server, (struct sockaddr*)&serverAddr, sizeof(serverAddr)) == -
25.        1)
26.    {
27.        printf("ERROR: Could not bind %s:%d\n", ADDR, PORT);
28.        return -1;
29.    }
30.
31.    if (listen(server, QUEUE) == -1)
32.    {
33.        printf("ERROR: Listening failed!\n");
34.        return -1;
35.    }
36.
37.    printf("Server running at %s:%d\n", ADDR, PORT);
38.
39.    char *tmp = "INSERT INTO LOG (TIME, SERVER_IP, SERVER_PORT, EVENT)" \
40.               "VALUES ('%s', '%s', %d, 'LISTEN');";
41.    sql = (char *)malloc(sizeof(char) * BUFF_SIZE);
42.    sprintf(sql, tmp, ctime(&t), ADDR, PORT);
43.    if (write_log() == -1)
```

```
44.    {
45.        return -1;
46.    }
47.
48.    while(1)
49.    {
50.        struct sockaddr_in clientAddr;
51.        socklen_t length = sizeof(clientAddr);
52.
53.        // handle connection
54.        int conn = accept(server, (struct sockaddr*)&clientAddr, &length);
55.        if (conn < 0)
56.        {
57.            printf("ERROR: Connection failed!\n");
58.            return -1;
59.        }
60.
61.        char *client_ip = inet_ntoa(clientAddr.sin_addr);
62.        int client_port = clientAddr.sin_port;
63.
64.        printf("A new connection from %s:%d\n", client_ip, client_port);
65.
66.        time(&t);
67.        tmp = "INSERT INTO LOG (TIME, CLIENT_IP, CLIENT_PORT, SERVER_IP, SERVER_PORT, EVENT) " \
68.              "VALUES ('%s', '%s', %d, '%s', %d, 'CONNECT');";
69.        sprintf(sql, tmp, ctime(&t), client_ip, client_port, ADDR, PORT);
70.        if (write_log() == -1)
71.        {
72.            return -1;
73.        }
74.
75.        // handle character
76.        while(1)
77.        {
78.            char receiveBuf[BUFF_SIZE];
79.            int cnt;
80.            memset(receiveBuf, 0, sizeof(receiveBuf));
81.
82.            // receive
83.            cnt = recv(conn, receiveBuf, sizeof(receiveBuf), 0);
84.
85.            time(&t);
```

```

86.             tmp = "INSERT INTO LOG (TIME, CLIENT_IP, CLIENT_PORT, SERVER_IP,
87.                         SERVER_PORT, EVENT, DETAIL)" \
88.                         "VALUES (\\"%s\\", \\"%s\\", %d, \\"%s\\", %d, \\"RECEIVE\\", \\"%
89.                         \");";
90.             sprintf(sql, tmp, ctime(&t), client_ip, client_port, ADDR, PORT,
91.                     receiveBuf);
92.             if (write_log() == -1)
93.             {
94.                 // leave
95.                 if (cnt == 0)
96.                 {
97.                     close(conn);
98.                     break;
99.                 }
100.
101.                // display
102.                write(fd, receiveBuf, cnt);
103.                printf("%d characters written\n", cnt);
104.            }
105.
106.            time(&t);
107.            tmp = "INSERT INTO LOG (TIME, CLIENT_IP, CLIENT_PORT, SERVER_IP
108.                          , SERVER_PORT, EVENT)" \
109.                          "VALUES (\\"%s\\", \\"%s\\", %d, \\"%s\\", %d, \\"CLOSE\\");";
110.            sprintf(sql, tmp, ctime(&t), client_ip, client_port, ADDR, PORT);
111.            if (write_log() == -1)
112.            {
113.                return -1;
114.            }
115.
116.            close(server);
117.            return 0;
118.        }

```

四、实验结果分析

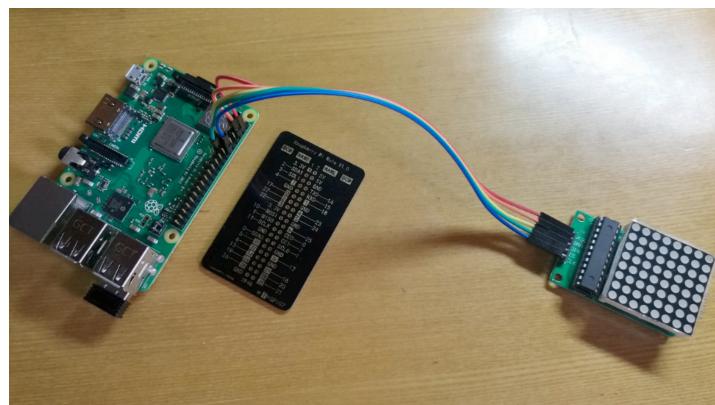
本次实验连线说明：

树莓派

MAX7219 驱动的点阵

5V	VCC
GND	GND
GPIO 0 (BCM 17)	DIN
GPIO 2 (BCM 27)	CS
GPIO 1 (BCM 18)	CLK

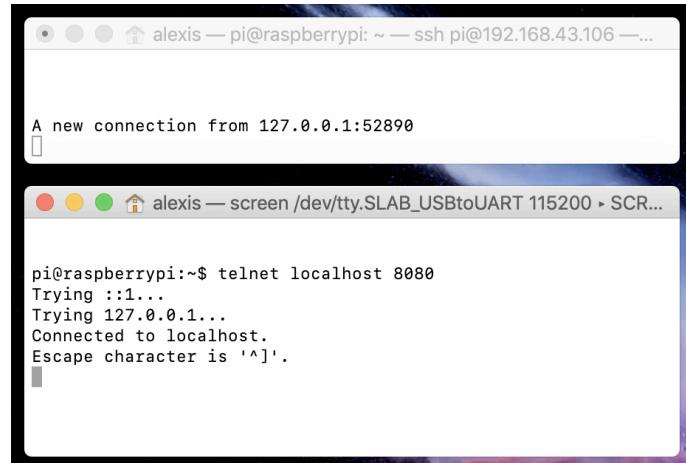
实物连线图如下：



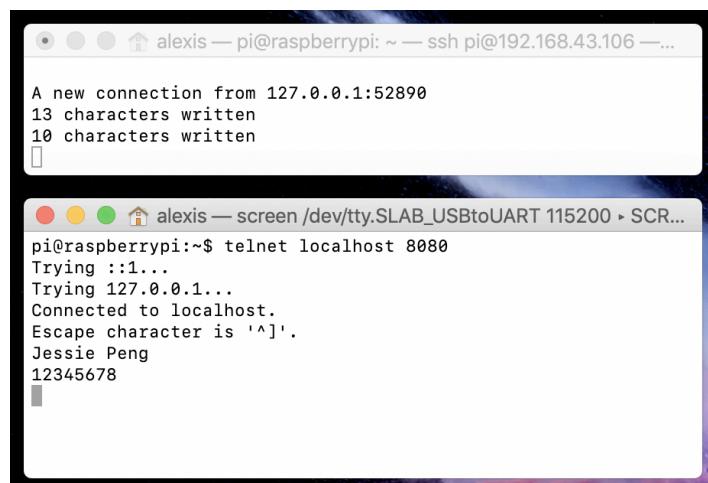
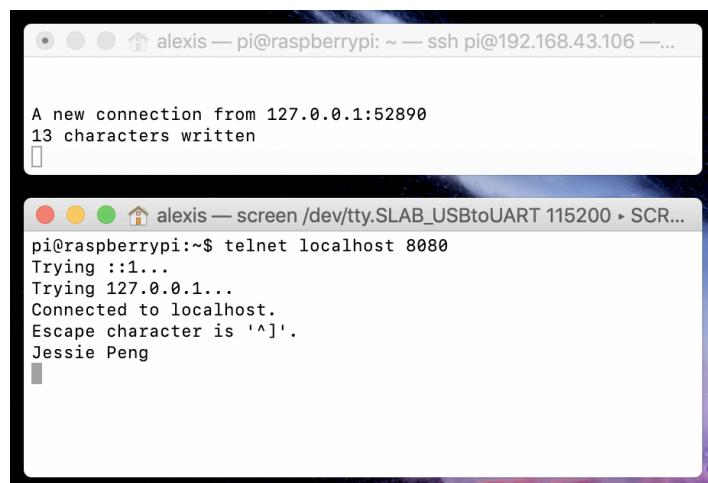
安装上次实验编译的内核模块，生成/dev/matrix 驱动设备。用 gcc server2.c -o server2 -lsqlite3 语句编译服务端程序，运行：

```
alexis — pi@raspberrypi: ~ — ssh pi@192.168.43.106 —...
[pi@raspberrypi:~ $ ./server2
Opened database successfully
Table created successfully
Server running at 0.0.0.0:8080]
```

客户端与服务器建立连接：



客户端发送字符串：



关闭连接后，查看日志：



```
pi@raspberrypi: ~ - ssh pi@192.168.43.106 - 70x19
[pi@raspberrypi: ~ - $ sqlite3 test.db
SQLite version 3.16.2 2017-01-06 16:32:41
Enter ".help" for usage hints.
sqlite> select * from log;
1|Tue Jun 25 15:26:50 2019
|||0.0.0|8080|LISTEN|
2|Tue Jun 25 15:27:38 2019
|127.0.0.1|52890|0.0.0.0|8080|CONNECT|
3|Tue Jun 25 15:31:49 2019
|127.0.0.1|52890|0.0.0.0|8080|RECEIVE|Jessie Peng

4|Tue Jun 25 15:32:34 2019
|127.0.0.1|52890|0.0.0.0|8080|RECEIVE|12345678

5|Tue Jun 25 15:33:24 2019
|127.0.0.1|52890|0.0.0.0|8080|RECEIVE|
6|Tue Jun 25 15:33:24 2019
|127.0.0.1|52890|0.0.0.0|8080|CLOSE|
sqlite> ]
```

五、讨论与心得

本次实验的点阵驱动沿用了上一次实验的内核模块，通过 `write()` 向 `/dev/matrix` 文件写入数据即驱动点阵显示指定的字符串。

网络通信和日志记录主要涉及到计算机网络和数据库，通过编写简单的单线程服务端程序，我学习了 TCP 协议和 socket 编程等基本知识，通过操作 SQLite3，也顺便复习了数据库的基本操作。