嵌入式实验报告

目录

[嵌入式实验报告 1](#_Toc169615627)

[1.概述 3](#_Toc169615628)

[1.1 设计概述 3](#_Toc169615629)

[2.需求分析 4](#_Toc169615630)

[2.1 功能需求 4](#_Toc169615631)

[2.2 技术指标 5](#_Toc169615632)

[2.3 开发环境 6](#_Toc169615633)

[3.设计思想 6](#_Toc169615634)

[4.准备工作 7](#_Toc169615635)

[4.1 连接实验箱 7](#_Toc169615636)

[4.2 配置交叉编译器和编译内核 8](#_Toc169615637)

[4.3 使用U盘更新 8](#_Toc169615638)

[4.4 交叉编译器的使用和程序的运行 8](#_Toc169615639)

[5.技术实现 10](#_Toc169615640)

[5.1 进入功能实现 10](#_Toc169615641)

[密码进入实现 10](#_Toc169615642)

[NFC功能实现 13](#_Toc169615643)

[主线程 14](#_Toc169615644)

[5.2 点阵实现（以输入密码界面为例） 15](#_Toc169615645)

[5.3 欢迎界面实现 17](#_Toc169615646)

[5.4 时间设置实现 20](#_Toc169615647)

[接收键盘信号 20](#_Toc169615648)

[七段数码管显示 22](#_Toc169615649)

[5.5 主函数框架 23](#_Toc169615650)

[5.6 计时功能实现 32](#_Toc169615651)

[5.7 LED和蜂鸣器实现 33](#_Toc169615652)

[5.8 矩阵键盘实现 34](#_Toc169615653)

[6.系统测试 36](#_Toc169615654)

[6.1 进入系统 36](#_Toc169615655)

[6.2 欢迎界面 37](#_Toc169615656)

[6.3 设置时间 38](#_Toc169615657)

[6.4 功能使用 39](#_Toc169615658)

[6.5 暂停与退出 41](#_Toc169615659)

[7.总结 41](#_Toc169615660)

1.概述

1.1 设计概述

本实验设计的智能医疗系统主要用于模拟医院中的挂水和病人监护过程，通过一系列的电子元器件和嵌入式编程实现对挂水全过程的监控与操作。系统以龙芯2K500为核心控制单元，通过刷卡或输入密码进入操作界面，主要功能和特点如下：

1. 欢迎界面：系统启动后，首先进入欢迎界面。点阵显示屏显示“欢迎使用智能医疗系统”，用户可以在该界面设置挂水的默认时间（初始设置为20秒）以及对系统的各项功能进行测试。欢迎界面提供了按键退出功能，便于用户进入正式操作界面。

2. 挂水过程模拟：

- 绿灯状态：挂水过程开始时，绿灯亮起，持续10秒（初始设置时间的一半）。在此期间，用户需要执行一系列操作。

- 黄灯状态：绿灯状态结束后，系统进入黄灯状态，再持续10秒。如果用户在这20秒内没有完成操作，系统将启动红灯和蜂鸣器报警，提示用户需要进行操作。

- 红灯和蜂鸣器报警：当用户在指定时间内未完成操作时，红灯亮起并伴随蜂鸣器报警，直到用户完成相应的动作。

3. 用户操作：用户需要按照顺序完成以下操作：

- 电机准备：启动电机进行准备。

- 电机正转：通过按键控制电机正转，模拟拆除吊瓶的过程。

- 电机反转：通过按键控制电机反转，模拟挂上新的吊瓶。

4. 操作完成反馈：用户完成以上操作后，系统返回绿灯状态，表示挂水过程正常进行。

5. 呼叫功能：在挂水过程中，用户可以通过按键启动蜂鸣器，模拟呼叫功能，以便在需要时提醒医护人员。

6. 环境参数显示：系统能够实时显示当前的温度和湿度，便于用户了解当前环境状态。

7. 暂停和设置功能：在挂水过程中，用户可以通过按键暂停系统，进入欢迎界面修改挂水时间设置，提供了灵活的操作空间。

本设计通过轮询检测用户的动作是否完成，确保挂水过程的每一步都得到正确执行。整个系统的设计旨在模拟医院中挂水的实际操作，帮助用户更好地理解和掌握挂水过程中的各项操作要求，提高智能医疗系统的实用性和操作性。

2.需求分析

2.1 功能需求

智能医疗系统的功能需求主要围绕挂水过程的模拟操作和用户交互界面的设计展开。系统应满足以下具体功能需求：

1. 用户身份验证：

- 支持用户通过刷卡或输入密码的方式进入系统操作界面，确保操作的安全性和用户身份的合法性。

2. 欢迎界面：

- 启动系统后，显示欢迎界面，点阵显示屏显示“欢迎使用智能医疗系统”。

- 在欢迎界面，用户可以设置挂水的默认时间（初始设置为20秒）。

- 用户可以通过按键退出欢迎界面，进入正式操作界面。

3. 挂水过程的指示灯显示：

- 绿灯状态：挂水开始时绿灯亮起，持续10秒，提示用户准备操作。

- 黄灯状态：绿灯状态结束后进入黄灯状态，再持续10秒，提示用户继续操作。

- 红灯报警：如果用户在绿灯和黄灯状态期间没有完成操作，系统启动红灯和蜂鸣器报警，直到用户完成相应的动作。

4. 用户操作控制：

- 电机准备：系统启动电机进行准备工作。

- 电机正转：用户通过按键控制电机正转，模拟拆除吊瓶。

- 电机反转：用户通过按键控制电机反转，模拟挂上新的吊瓶。

5. 操作完成指示：

- 用户完成所有操作后，系统重新进入绿灯状态，表示挂水过程正常进行。

6. 呼叫功能：

- 用户在挂水过程中可以通过按键启动蜂鸣器，模拟呼叫功能，以便在需要时提醒医护人员。

7. 环境参数显示：

- 系统能够实时检测并显示当前的温度和湿度，方便用户了解当前环境状态。

8. 系统暂停和设置功能：

- 用户可以在挂水过程中通过按键暂停系统，进入欢迎界面修改挂水时间设置。

- 用户可以对系统的各项功能进行测试，以确保系统正常运行。

9. 系统轮询检测：

- 系统通过轮询检测用户的操作状态，确保挂水过程中的每一步都得到正确执行。

通过满足上述功能需求，智能医疗系统能够模拟实际的挂水过程，提供用户友好的交互界面和多样的功能选择，增强用户对挂水过程的理解和操作熟练度，同时提高系统的实用性和可靠性。

2.2 技术指标

智能医疗系统的技术指标要求系统在硬件和软件方面都具有较高的可靠性、稳定性和可操作性，以确保过程的准确模拟和用户友好的交互体验。具体技术指标如下：

1. 硬件指标：

- 控制单元：采用龙芯2K500芯片作为核心控制单元，确保系统运行的稳定性和高效性。

- 显示：使用点阵显示屏显示欢迎界面，使用七段数码管显示环境参数和时间。

- LED：绿灯、黄灯、红灯：分别用于指示不同的挂水状态。

- 蜂鸣器：能够有效提醒用户进行操作或模拟呼叫功能。

- 电机：准备、正转、反转功能。

- 传感器：温度传感器和湿度传感器：能够实时检测当前环境参数。

- 输入设备：NFC和矩阵键盘。

2. 软件指标：

- 系统初始化：系统启动时完成所有硬件设备的初始化，进入欢迎界面。

- 用户身份验证：提供NFC和密码两种验证方式。

- 挂水过程控制：绿灯、黄灯、红灯的状态切换时间精确。用户操作的检测和反馈及时。

- 系统轮询：轮询周期设置合理，确保用户操作的实时检测，避免漏检和误检。

3. 系统性能指标：

- 实时性：挂水过程中的各项操作和状态切换在规定时间内完成。

- 易用性：用户操作简单直观，能够快速上手使用各项功能。

2.3 开发环境

Ubuntu18.04

3.设计思想

智能医疗系统的设计思想基于模块化、易用性和可靠性原则，以确保系统在模拟挂水过程中的高效性和用户友好的交互体验。

1. 用户身份验证模块：设计刷卡和密码两种验证方式。

2. 欢迎模块：显示欢迎字符，可以设置时间。

3. 时间设置模块：设置时间。

4. 挂水过程控制模块：通过控制绿灯、黄灯、红灯的状态切换表示剩余时间。电机控制实现吊瓶的拆除和更换。

为了提高系统的易用性，设计过程中重点考虑了用户交互的简便性。用户通过刷卡或输入密码即可进入系统，操作流程简便快捷。按键操作响应灵敏，用户可以迅速控制电机和蜂鸣器，实现吊瓶的更换和呼叫功能。

系统设计过程中注重可靠性，主要体现在：

1. 实时检测：系统通过轮询机制实时检测用户操作状态，确保挂水过程的每一步都得到正确执行。

2. 对执行顺序的要求。

综上所述，智能医疗系统的设计思想立足于模块化、易用性、可靠性，通过科学的设计和合理的选型，确保系统能够高效、稳定地模拟过程，为用户提供良好的操作体验。

4.准备工作

4.1 连接实验箱

安装VMware虚拟机，在虚拟机中安装Ubuntu18.04.

用UART-USB 线将实验箱与电脑连接，分别接在实验箱的USB DEBUG接口和电脑的USB接口。

在Linux上安装minicom：

sudo apt-get install minicom

将实验箱与电脑连接：在shell中使用下面命令连接试验箱，通常试验箱的设备名为ttyUSB0。

minicom -D /dev/ttyUSB0

启动minicom之后，按下开发板上PWR ON 按钮开机。

4.2 配置交叉编译器和编译内核

准备FAT格式U盘，并在U盘中创建名为update的文件夹。

将资料包中的toolchain-loongarch64-linux-gnugcc8-host-x86\_64-2022-07-18.tar.xz 交叉编译工具链压缩包解压到/opt目录下。

接着进入linux-5.10-2k500-cbd-src目录下，执行如下操作：

source ./set\_env.sh #设置交叉工具链等环境

make loongson\_2k500\_defconfig #指定板载卡配置

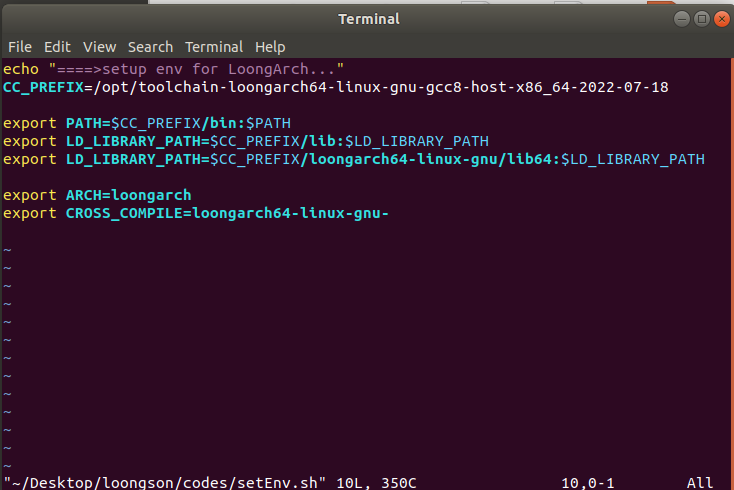
make #开始编译

4.3 使用U盘更新

使用已经给出的文件系统、内核和固件，即u-boot-with-spl.bin、uImage、rootfs-ubifs-ze.img，放入U盘的update目录中，插入到开发板上。用minicom把主机与开发板进行连接，在启动开发板时按住M键，进入u-boot设置菜单，选择Update All，使用U盘更新（by usb），更新完成后，使用reset命令重启开发板，内核、固件与文件系统的升级完成。

4.4 交叉编译器的使用和程序的运行

配置交叉编译器的环境变量。在任意文件位置处，创建一个setEnv.sh文件：



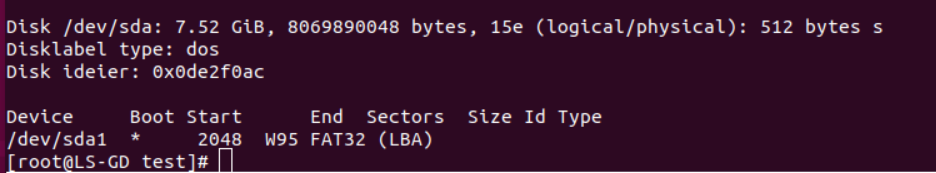
编译程序前，需要先运行前面的setEnv.sh文件设置环境变量，再使用运行make完成编译：

source ./setEnv.sh

make

完成编译之后，将生成的可执行文件通过U盘传输到开发板上。

文件存入FAT32格式U盘中，插在开发板的USB接口上。使用 fdisk -l 指令，在最下方的设备表格中可以找到U盘，如下图：



sda1即为该U盘。然后将其挂载。这里将他挂载到/mnt/usb目录下。

descript

接着进入/mnt/usb目录下，找到可执行文件，运行。

拔出U盘前，需要使用如下命令卸载：

umount /dev/sda1

5.技术实现

5.1 进入功能实现

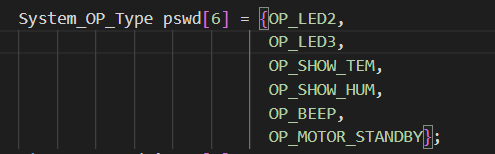
密码进入实现

1. **void** \*password\_keybord(**void** \*arg)
2. {
3. System\_OP\_Type pswd[6] = {OP\_LED2,
4. OP\_LED3,
5. OP\_SHOW\_TEM,
6. OP\_SHOW\_HUM,
7. OP\_BEEP,
8. OP\_MOTOR\_STANDBY};
9. uint16\_t pswd\_input[6];
10. uint16\_t key;
11. **int** ps = 0;
12. //int ret = pthread\_create(&logo\_pth, NULL, logo\_display\_inputpswd, NULL);
13. // if (ret)
14. // {
15. //  printf("logo初始化失败！\n");
16. //  return;
17. // }
18. **while** (ps != 1)
19. {
20. **int** i=0;
21. **while**(ps!=1){
22. key = key\_scan();
23. sleep(0.1);
24. **if** (key != 0)
25. {
26. // System\_OP\_Type op = key\_decode(key);
27. printf("key:%u\n", key);
28. **if** (key & KEY16(15))
29. {
30. system\_finish=1;
31. **return**;
32. }
33. **else** **if** (key & KEY16(14))
34. {
35. i = 0;
36. }
37. **else**
38. {
39. pswd\_input[i] = key;
40. i++;
41. **if**(i==6) **break**;
42. }
43. }
44. }
45. **for** (**int** i = 0; i <= 6; i++)
46. {
47. **if** (i == 6)
48. {
49. //beep\_on();
50. led\_on(0);
51. msleep(1000);
52. //beep\_off();
53. led\_off(2);
54. ps = 1;
55. key\_finish=1;
56. printf("welcome!\n");
57. **break**;
58. }
59. **if** (key\_decode(pswd\_input[i]) != pswd[i])
60. {
61. ps = 0;
62. beep\_on();
63. led\_on(2);
64. msleep(1000);
65. beep\_off();
66. led\_off(2);
67. printf("retype the password!\n");
68. **break**;
69. }
70. }
71. }
72. }

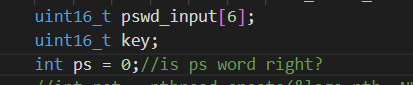
全局变量key\_finish，用于主线程检测用户是否通过输入密码进入：



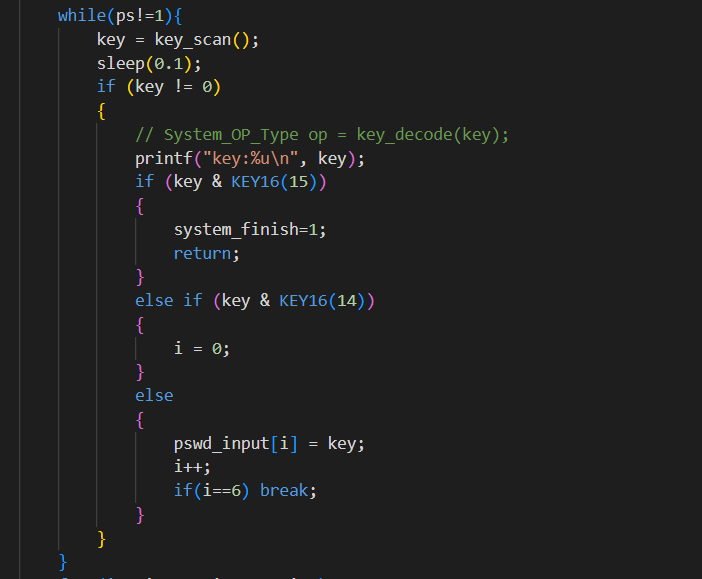
正确密码以数组形式储存，储存为对键盘信号解码后的结果：



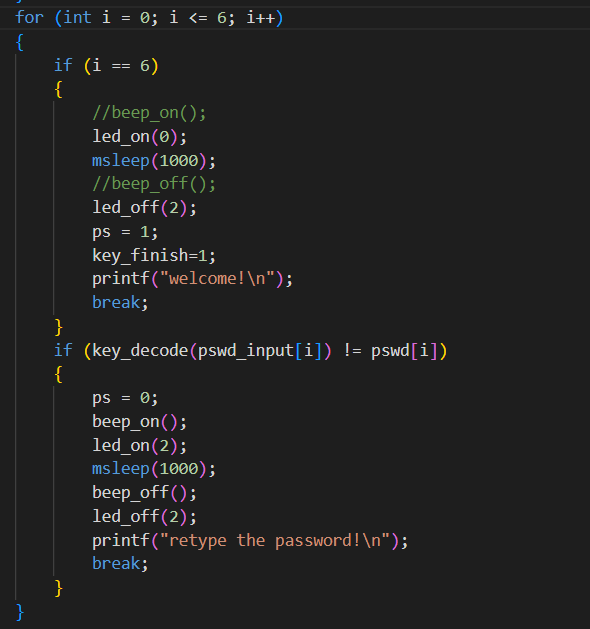
用uint16\_t的键盘信号输入形式存储用户输入的密码：



接收密码，按键14是重新输入，按键15是退出系统，接收到6个输入后跳出循环：



验证密码是否正确，对储存的输入每一位解码后与前面存储的进行比对，如果有错就打开蜂鸣器，亮红灯，并持续1s，进入到输入密码的循环开始，重新输入密码；如果没错，将ps置1，key\_finish置1，结束输入密码线程：

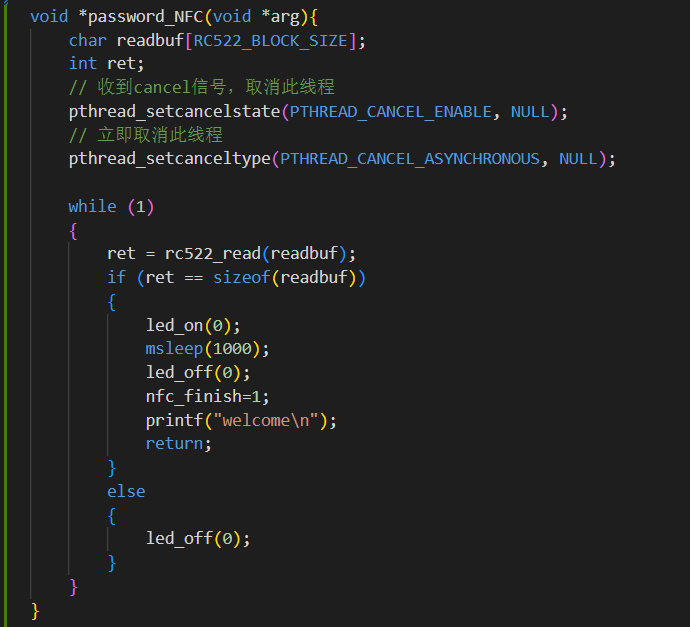


NFC功能实现

全局变量：

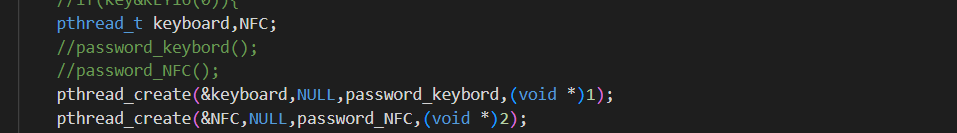


当检测到NFC信号，LED绿灯亮起并持续1s，全局变量nfc\_finish置1，线程结束：

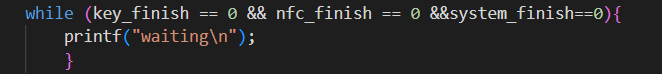


主线程

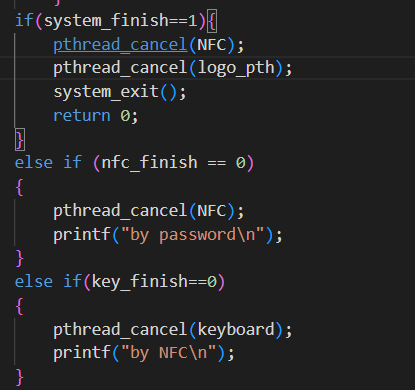
创建密码进入、NFC进入的线程：



等待某一种方式成功进入后将全局变量置1：



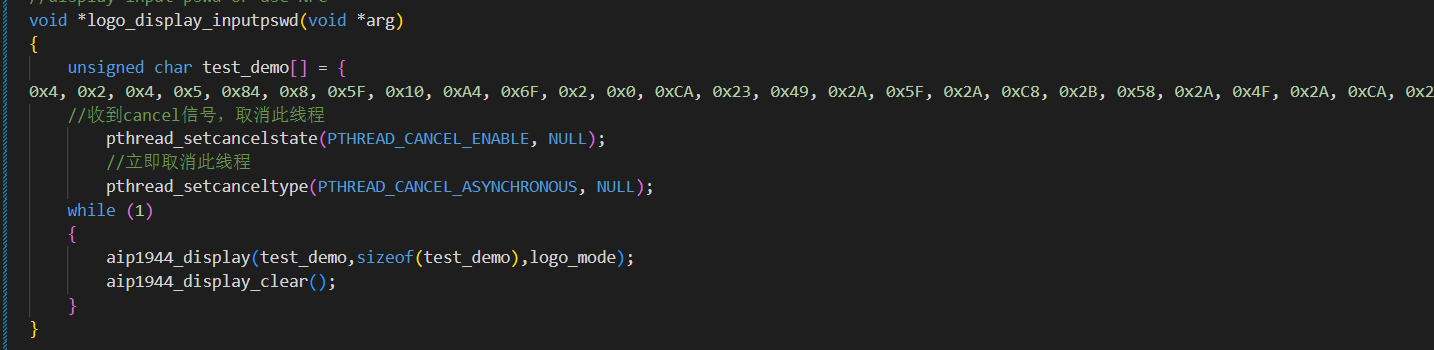
如果是输入密码线程接收到按键15信息，即系统退出，就退出系统；如果是输入密码成功进入（nfc\_finish==0），就中止NFC线程，如果是通过NFC进入，就中止输入密码线程：



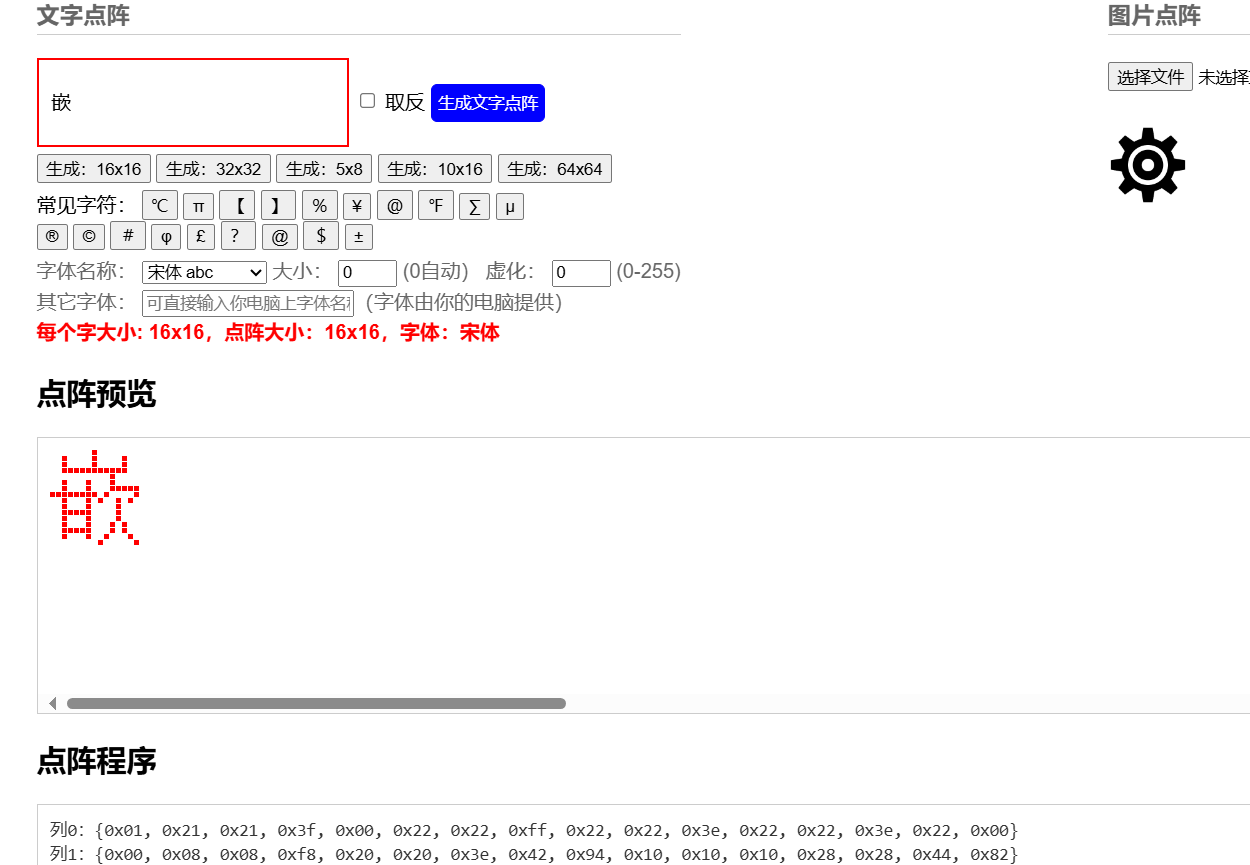
5.2 点阵实现（以输入密码界面为例）

当前的线程创建一个用于点阵显示的线程：



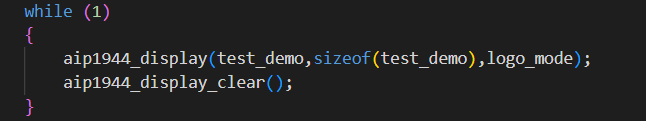


点阵的文字显示使用类似于活字印刷的方式，在提供的网站上贴上想要显示的单个文字，将其转化为16x16点阵的形式，再用aip1944文件夹中提供的工具，将前面生成的点阵格式转化为适合龙芯2k500的点阵格式：

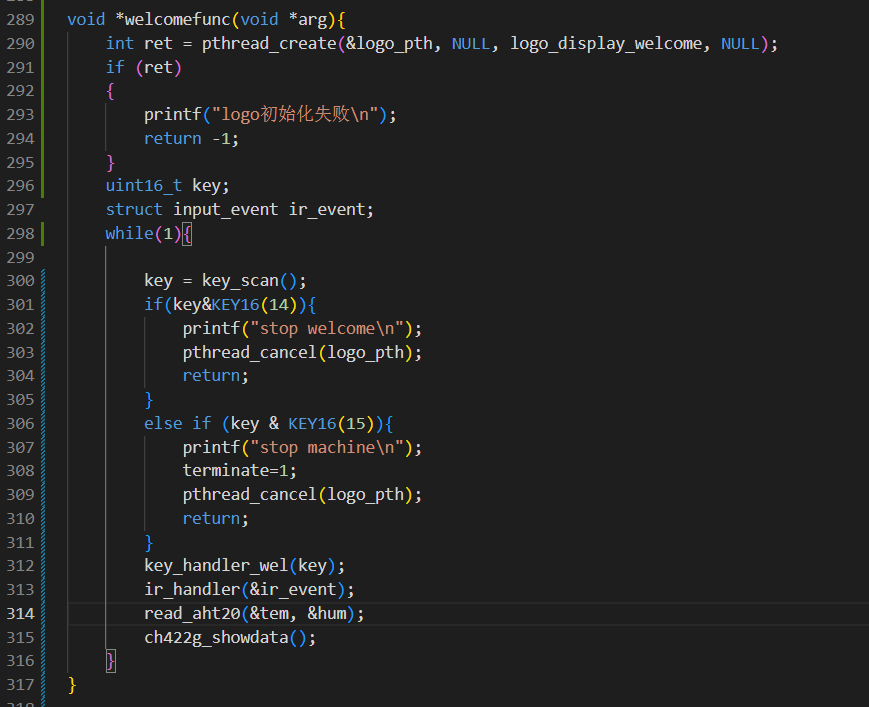




将最终生成的文字按顺序贴入数组中，调用aip1944\_display，如果执行程序，此时文字会以logo\_mode的形式在点阵上显示：

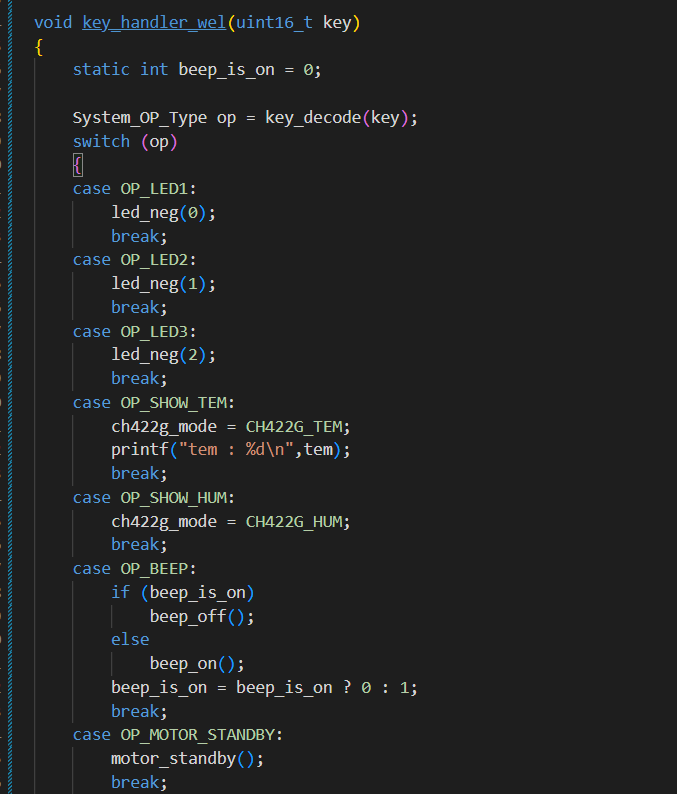


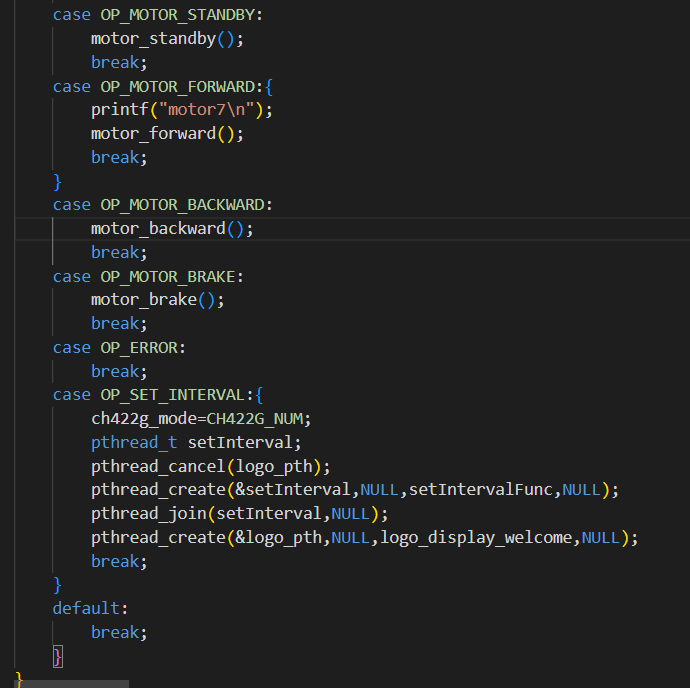
5.3 欢迎界面实现



创建线程logo\_display\_welcome，具体实现和上面的点阵模块相同，把点阵输出的字符换成“欢迎使用智能医疗系统”。

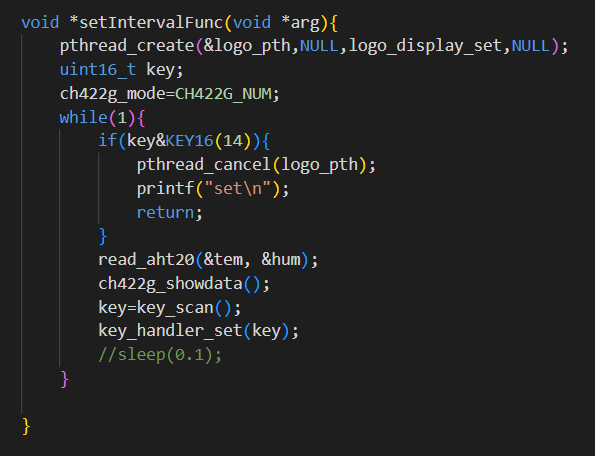
key\_scan()接收矩阵键盘信息，对不同按键做出相应处理，在欢迎界面，可以对各种功能进行测试，使用按键11可以进入时间设置界面。





当按下按键11，进入case OP\_SET\_INTERVAL进行处理，欢迎界面线程创建设置时间线程，并且等待设置时间线程执行结束。

5.4 时间设置实现

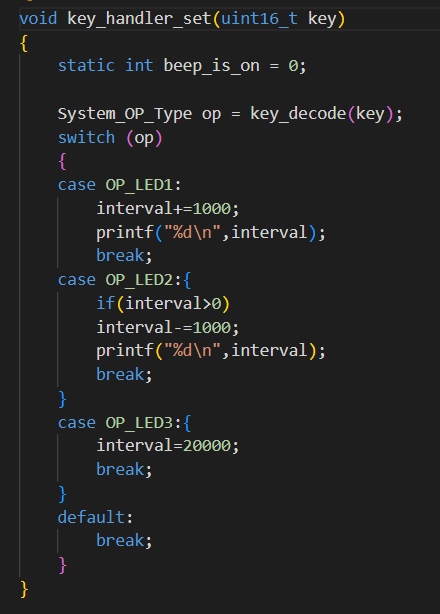


点阵显示“设置时间”，实现方法同上。

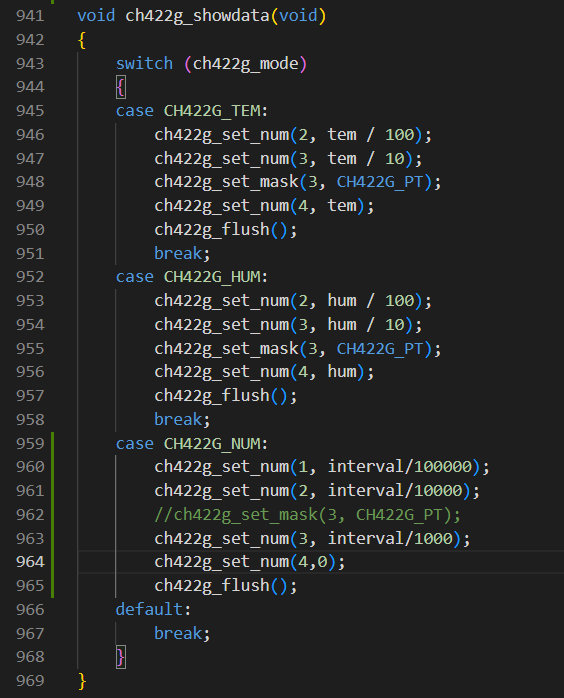
7段数码管更改显示模式为显示此时的间隔时间，默认是20s，即20000ms（int interval = 20000）。进入while循环，接收矩阵键盘信号：

接收键盘信号

按键0为增加1000ms，按键1为减少1000ms，并保证间隔时间大于0，按键2为恢复为默认的20000ms。



七段数码管显示



三种情况：温度、湿度、时间

温度，湿度传感器输出的数字都是不带小数点的三位数，比如现在的温度是33.6摄氏度，传感器会传回336，因此需要手动加小数点，即ch422g\_set\_mask(3, CH422G\_PT); 实现数码管显示正确的、带小数点的温度。即33.6。

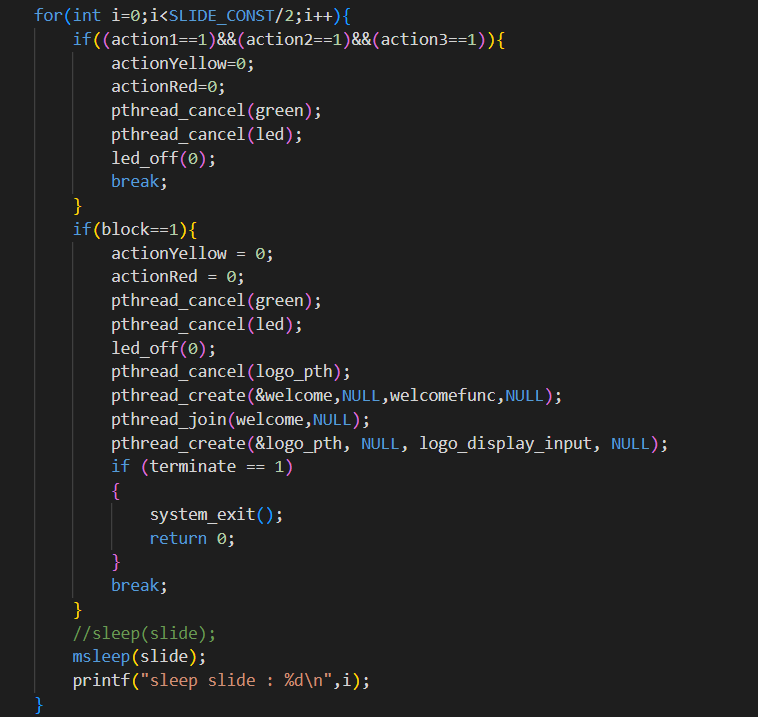
在显示时间的情况下（以默认时间20000ms为例）：第一位/100000，显示0，第二位/10000，显示2，第三位/1000，显示0，第四位显示0。

最后刷新数码管。

5.5 主函数框架

1. **while** (1)
2. {
3. actionYellow=1;
4. actionRed=1;
5. action1=0;
6. action2=0;
7. action3=0;
8. block=0;
9. terminate=0;
10. led\_off(0);
11. led\_off(1);
12. led\_off(2);
13. pthread\_create(&green,NULL,greenFunc,NULL);
14. printf("create green:\n");
15. pthread\_create(&led,NULL,ledFunc,(**void** \*)0);
16. slide = interval / SLIDE\_CONST;
17. printf("slide : %d\n",slide);
18. //polling
19. //when action 1&2&3 done, cancel thread
20. **for**(**int** i=0;i<SLIDE\_CONST/2;i++){
21. **if**((action1==1)&&(action2==1)&&(action3==1)){
22. actionYellow=0;
23. actionRed=0;
24. pthread\_cancel(green);
25. pthread\_cancel(led);
26. led\_off(0);
27. **break**;
28. }
29. **if**(block==1){
30. actionYellow = 0;
31. actionRed = 0;
32. pthread\_cancel(green);
33. pthread\_cancel(led);
34. led\_off(0);
35. pthread\_cancel(logo\_pth);
36. pthread\_create(&welcome,NULL,welcomefunc,NULL);
37. pthread\_join(welcome,NULL);
38. pthread\_create(&logo\_pth, NULL, logo\_display\_input, NULL);
39. **if** (terminate == 1)
40. {
41. system\_exit();
42. **return** 0;
43. }
44. **break**;
45. }
46. //sleep(slide);
47. msleep(slide);
48. printf("sleep slide : %d\n",i);
49. }
50. **if**(actionYellow==1){
51. pthread\_cancel(green);
52. pthread\_cancel(led);
53. led\_off(0);
54. pthread\_create(&yellow,NULL,yellowFunc,NULL);
55. printf("create yellow :\n");
56. pthread\_create(&led,NULL,ledFunc,(**void** \*)1);
57. **for**(**int** i=0;i<SLIDE\_CONST/2;i++){
58. **if**(action1==1&&action2==1&&action3==1){
59. actionRed = 0;
60. pthread\_cancel(yellow);
61. pthread\_cancel(led);
62. led\_off(1);
63. **break**;
64. }
65. **if**(block==1){
66. actionRed = 0;
67. pthread\_cancel(yellow);
68. pthread\_cancel(led);
69. led\_off(1);
70. pthread\_cancel(logo\_pth);
71. pthread\_create(&welcome, NULL, welcomefunc, NULL);
72. pthread\_join(welcome, NULL);
73. pthread\_create(&logo\_pth, NULL, logo\_display\_input, NULL);
74. **if**(terminate==1){
75. system\_exit();
76. **return** 0;
77. }
78. **break**;
79. }
80. msleep(slide);
81. printf("sleep slide : %d\n",i);
82. //sleep(slide);
83. }
84. **if**(actionRed==1){
85. pthread\_cancel(yellow);
86. pthread\_cancel(led);
87. led\_off(1);
88. pthread\_create(&red,NULL,redFunc,NULL);
89. pthread\_create(&led,NULL,ledFunc,(**void** \*)2);
90. pthread\_create(&beep,NULL,beepFunc,NULL);
91. printf("create red\n");
92. pthread\_join(red,NULL);
93. pthread\_cancel(beep);
94. pthread\_cancel(led);
95. led\_off(2);
96. beep\_off();
97. **if**(block==1){
98. pthread\_cancel(logo\_pth);
99. pthread\_create(&welcome, NULL, welcomefunc, NULL);
100. pthread\_join(welcome, NULL);
101. pthread\_create(&logo\_pth, NULL, logo\_display\_input, NULL);
102. **if** (terminate == 1)
103. {
104. system\_exit();
105. **return** 0;
106. }
107. }
108. }
109. }
110. // key = key\_scan();
111. // if (key & KEY16(15))
112. //  break;
114. // key\_handler(key);
115. // ir\_handler(&ir\_event);
116. // read\_aht20(&tem, &hum);
117. // // read\_stk8ba\_xyz(xyz);
118. // //  printf("x:%d, y:%d, z:%d\n", xyz[0], xyz[1], xyz[2]);
119. // ch422g\_showdata();
120. }

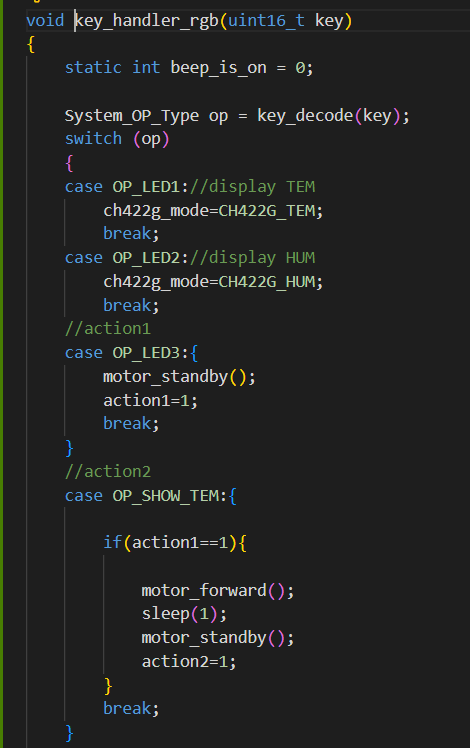
绿灯状态，创建绿灯线程，在此期间检测用户是否按顺序完成三个动作，如果完成，则不需要进入黄灯和红灯状态，终止绿灯线程，回到while循环开始。



绿灯线程（黄灯线程与绿灯线程相同）：  


接收键盘和温湿度传感器信号：

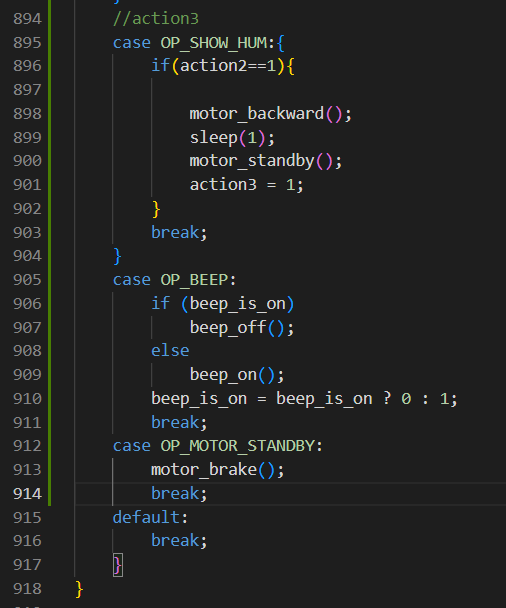
0号按键显示温度，1号键显示湿度，2号键为第一个动作。按下2号键后，电机准备，并将表示第一个动作完成的全局变量action1置为1。3号键为第二个动作，表示拆下旧的吊瓶，按下后，判断第一个动作是否完成，如果完成第一个动作，电机正转一秒后停止，将表示第二个动作完成的全局变量置为1。



4号键为第三个动作，即装入新的吊瓶。按下4号键，判断第二个动作是否完成，如果完成，电机反转一秒后停止，将表示第三个动作完成的action3置为1，至此，所有动作全部完成。

按键5为蜂鸣器响，表示呼叫医护人员。

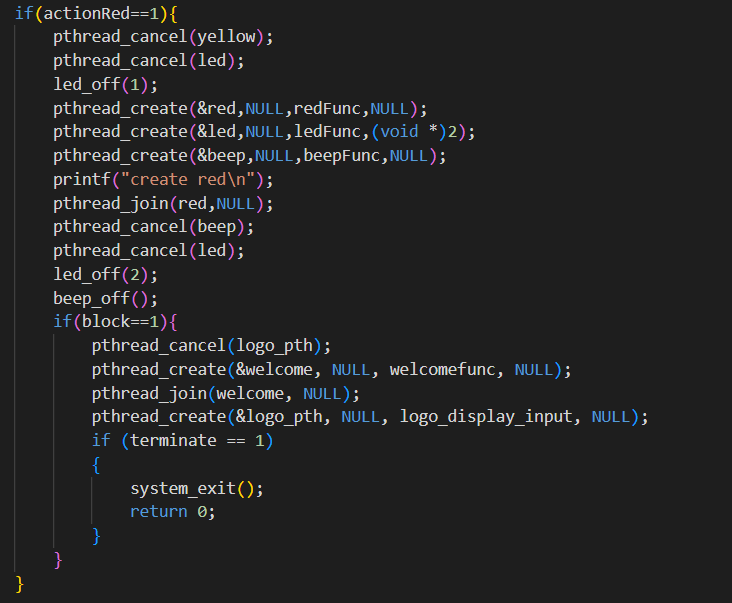
按键6为电机停止。



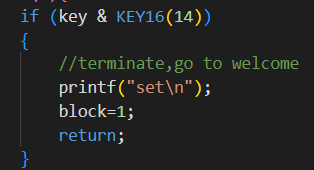
如果绿灯期间没有完成三个动作，进入黄灯状态：



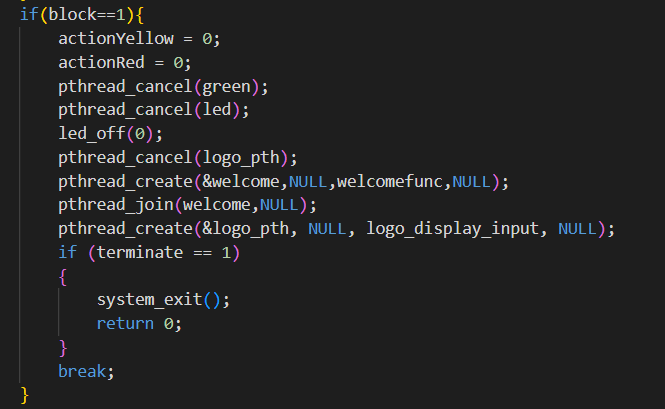
如果黄灯期间没有完成三个动作，进入红灯状态，主线程等待红灯线程执行结束，即三个动作完成或暂停、中止系统：



如果上述期间检测到按键14按下：



系统暂停，进入欢迎界面，主线程等待欢迎线程执行结束，结束后从while循环的开头开始继续执行：



5.6 计时功能实现

定义一个全局变量int interval=20000；并设置时间片（100个）：



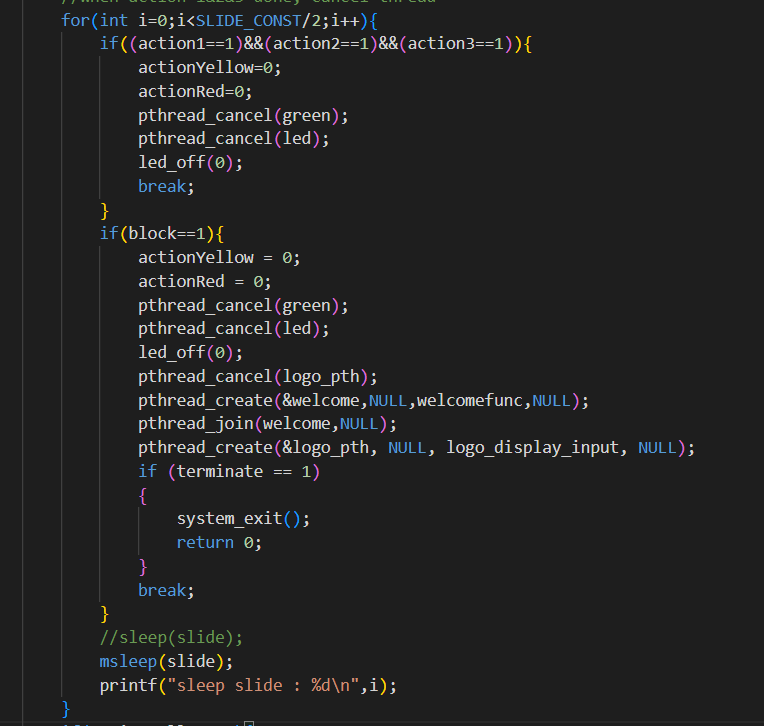


每个时间片大小为slide：



以绿灯状态为例，给绿灯线程分配了所有时间片数量的一半，主线程每隔一个时间片，检测三个动作是否完成，如果没有完成，继续之前的动作，并使用msleep（slide）等待一个时间片，如果完成了，终止绿灯和绿灯相关线程。

如果绿灯的时间片用完且三个动作并未全部按顺序完成，for循环结束，进入给黄灯状态分配的时间片，即剩下所有的时间片。

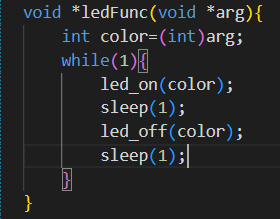


5.7 LED和蜂鸣器实现

系统运行过程中LED灯闪烁的实现（以绿灯为例）：

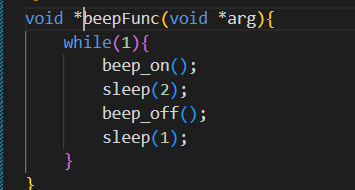
创建一个新的线程led用于控制led灯的亮灭，执行ledFunc，传入参数0，即LED绿灯的参数：





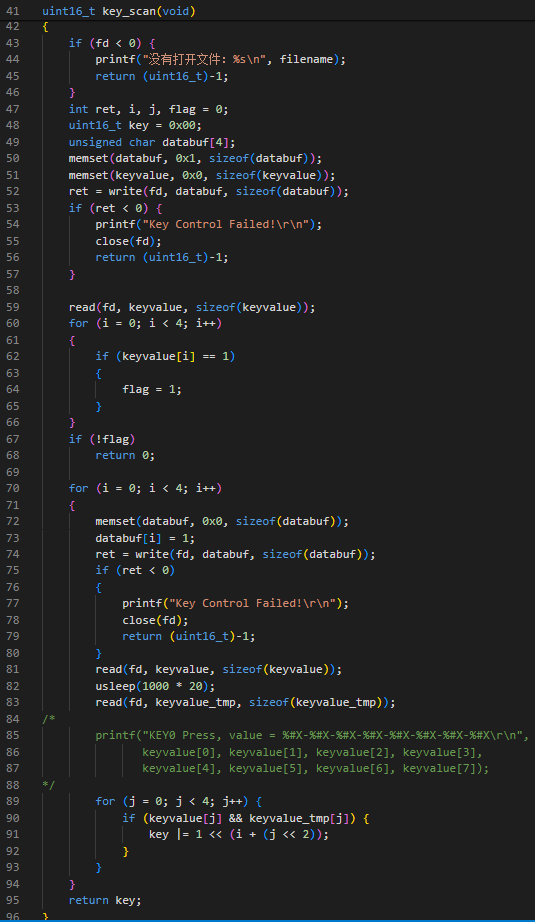
蜂鸣器：



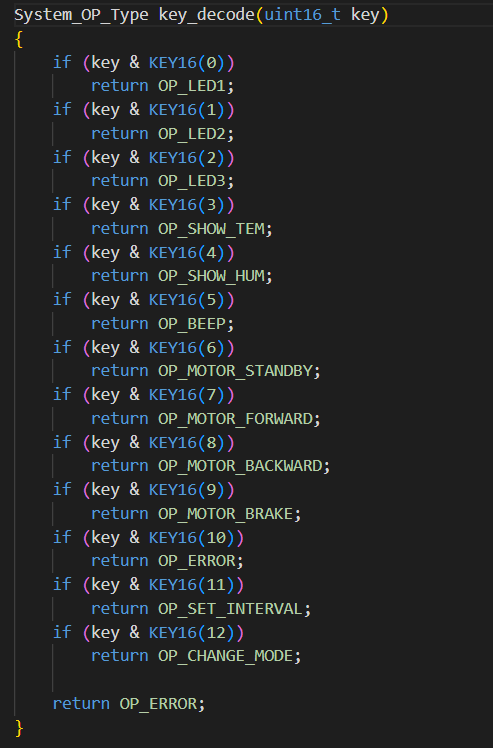


5.8 矩阵键盘实现

接收键盘信号：



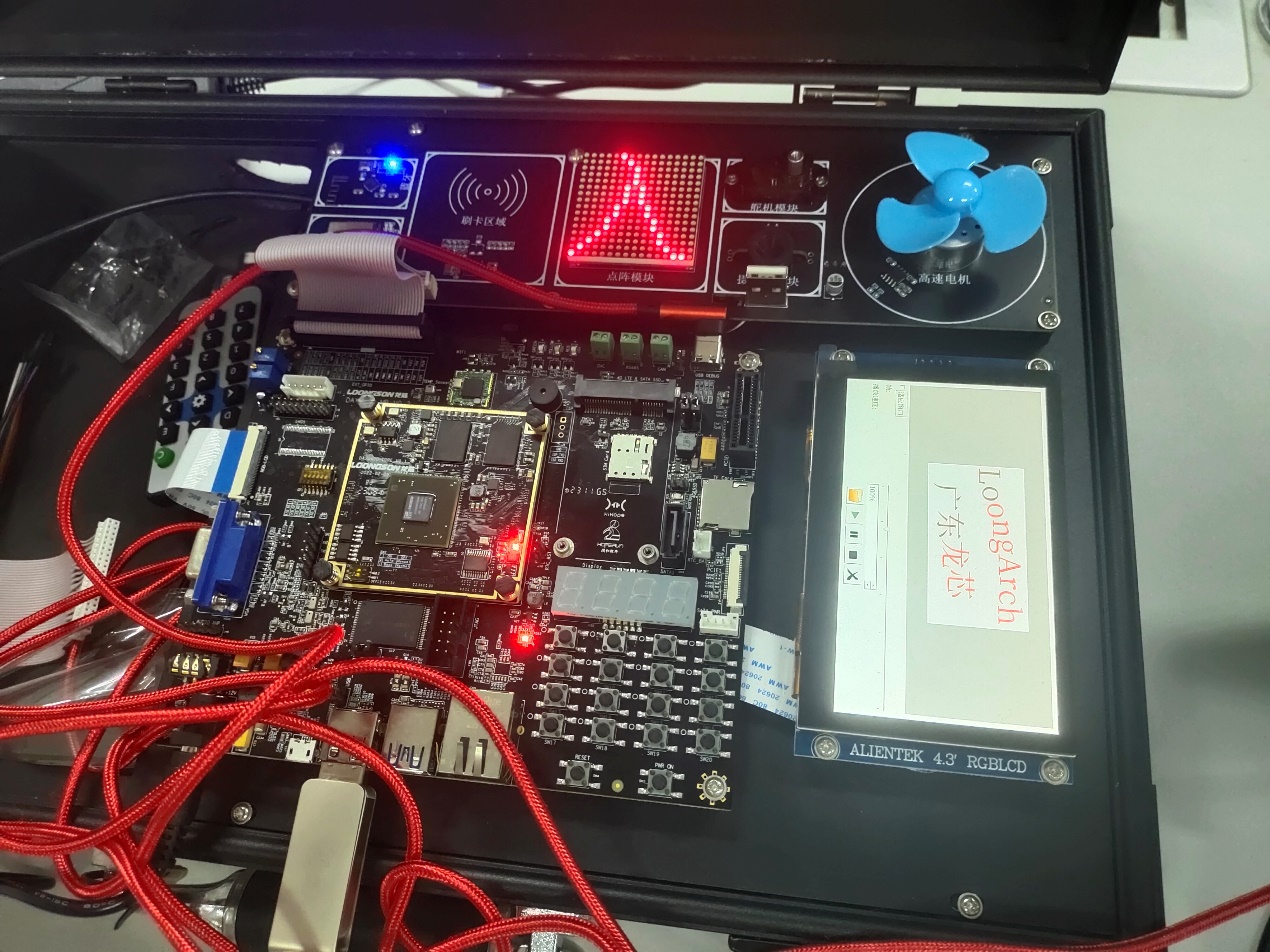
键盘信号解码：



6.系统测试

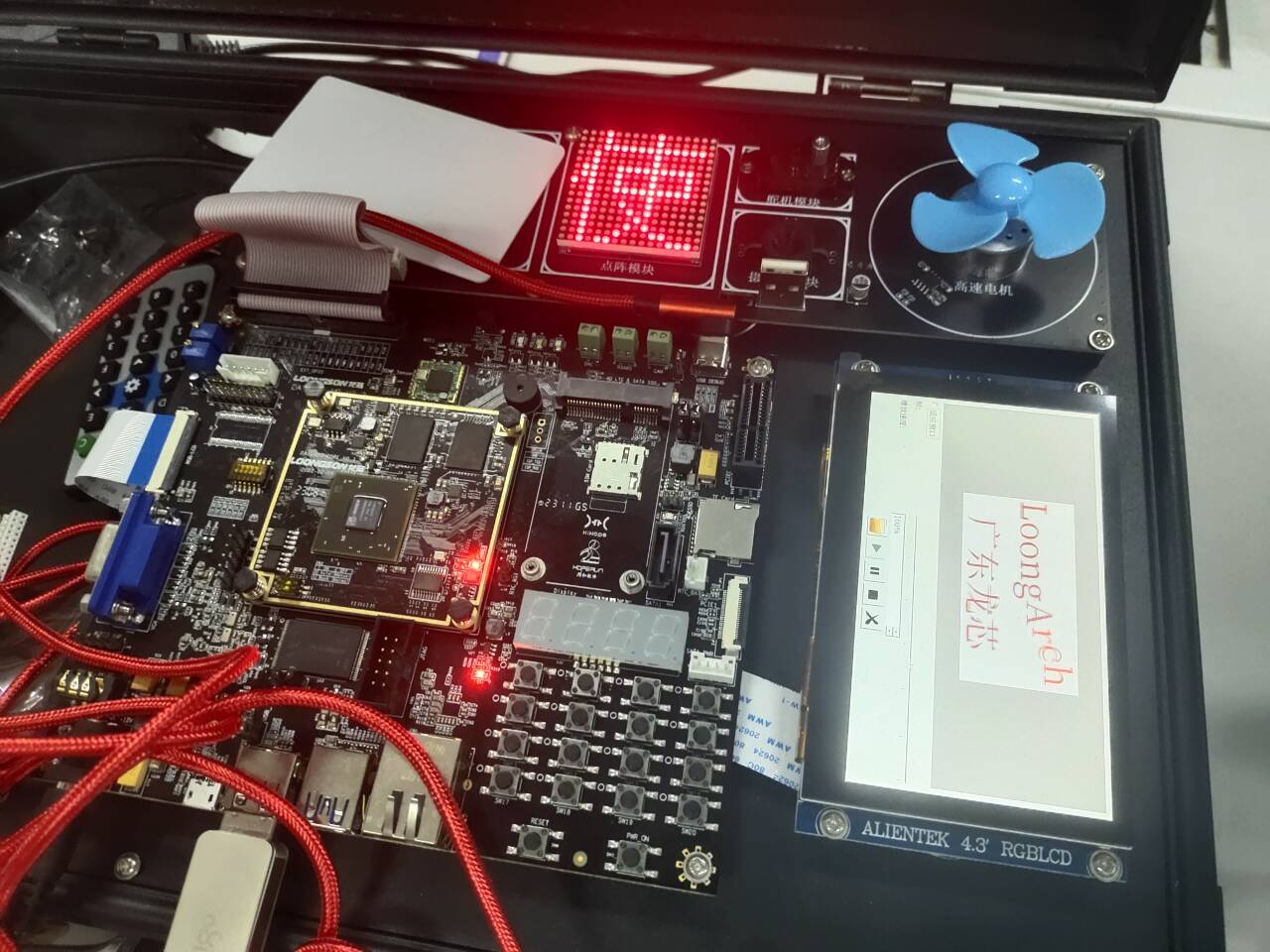
6.1 进入系统

点阵显示“输入密码或刷卡”



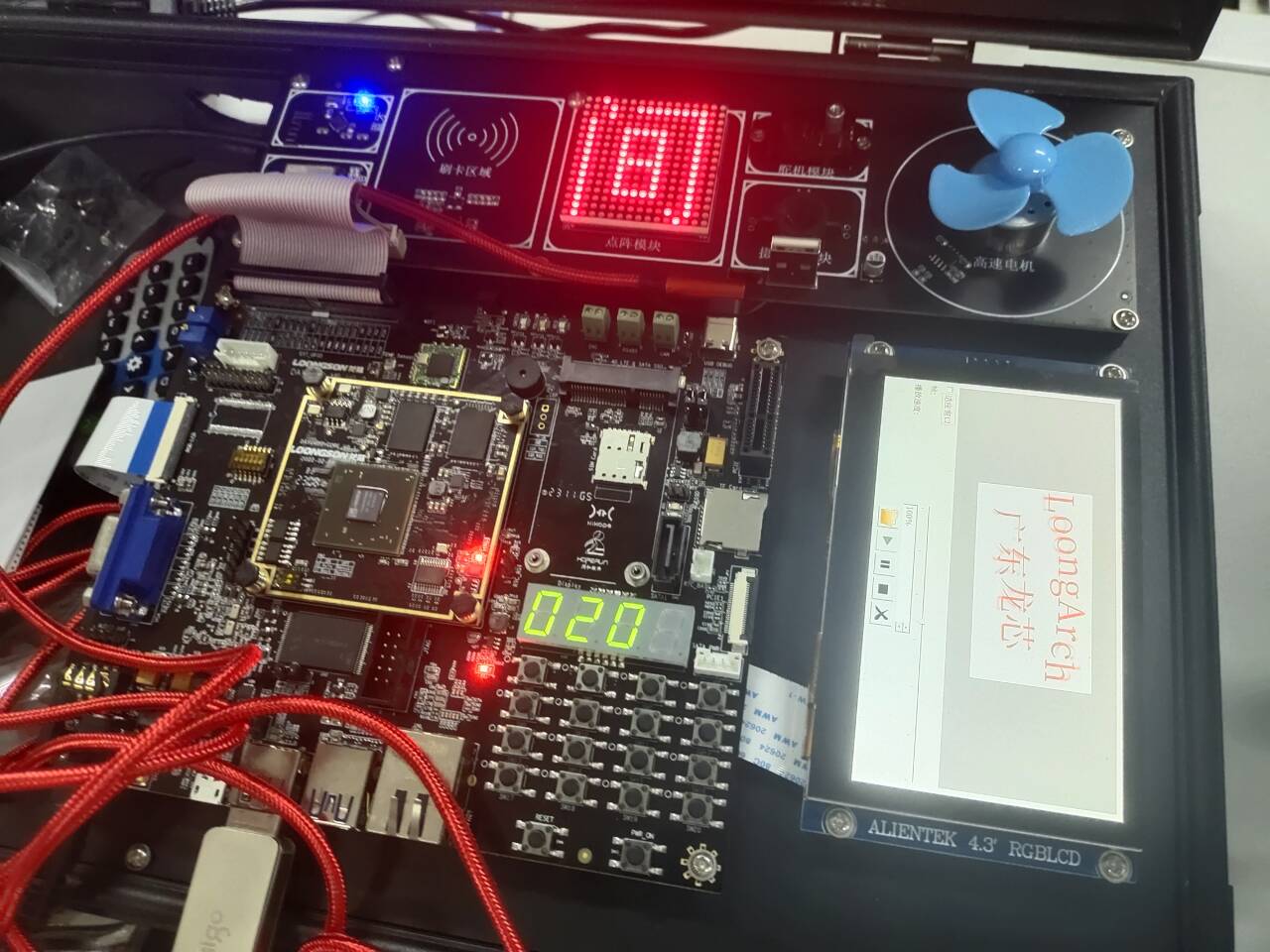
6.2 欢迎界面

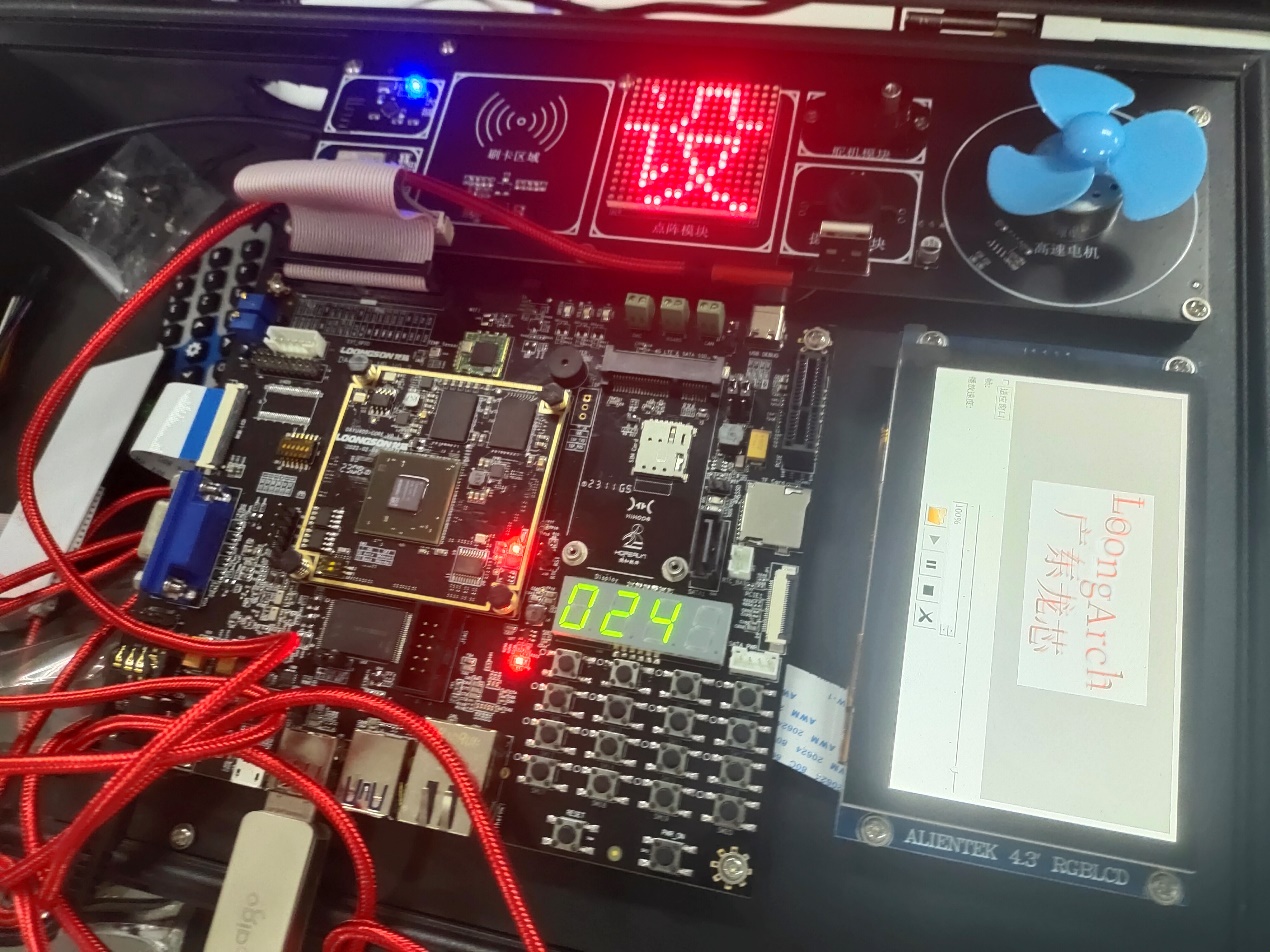
点阵显示“欢迎使用智能医疗系统”

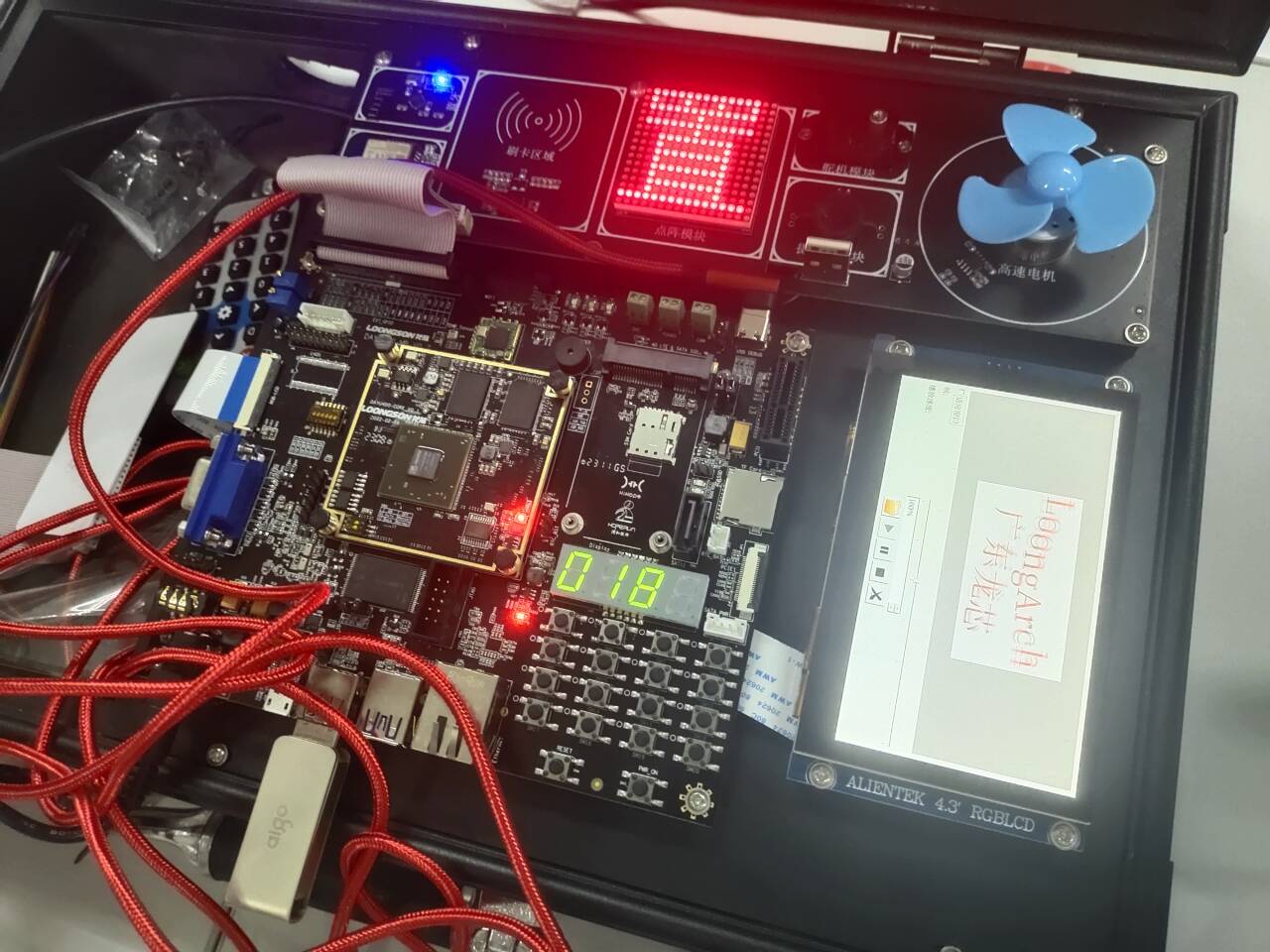


6.3 设置时间

七段数码管显示设置的时间（秒），点阵显示“设置时间“

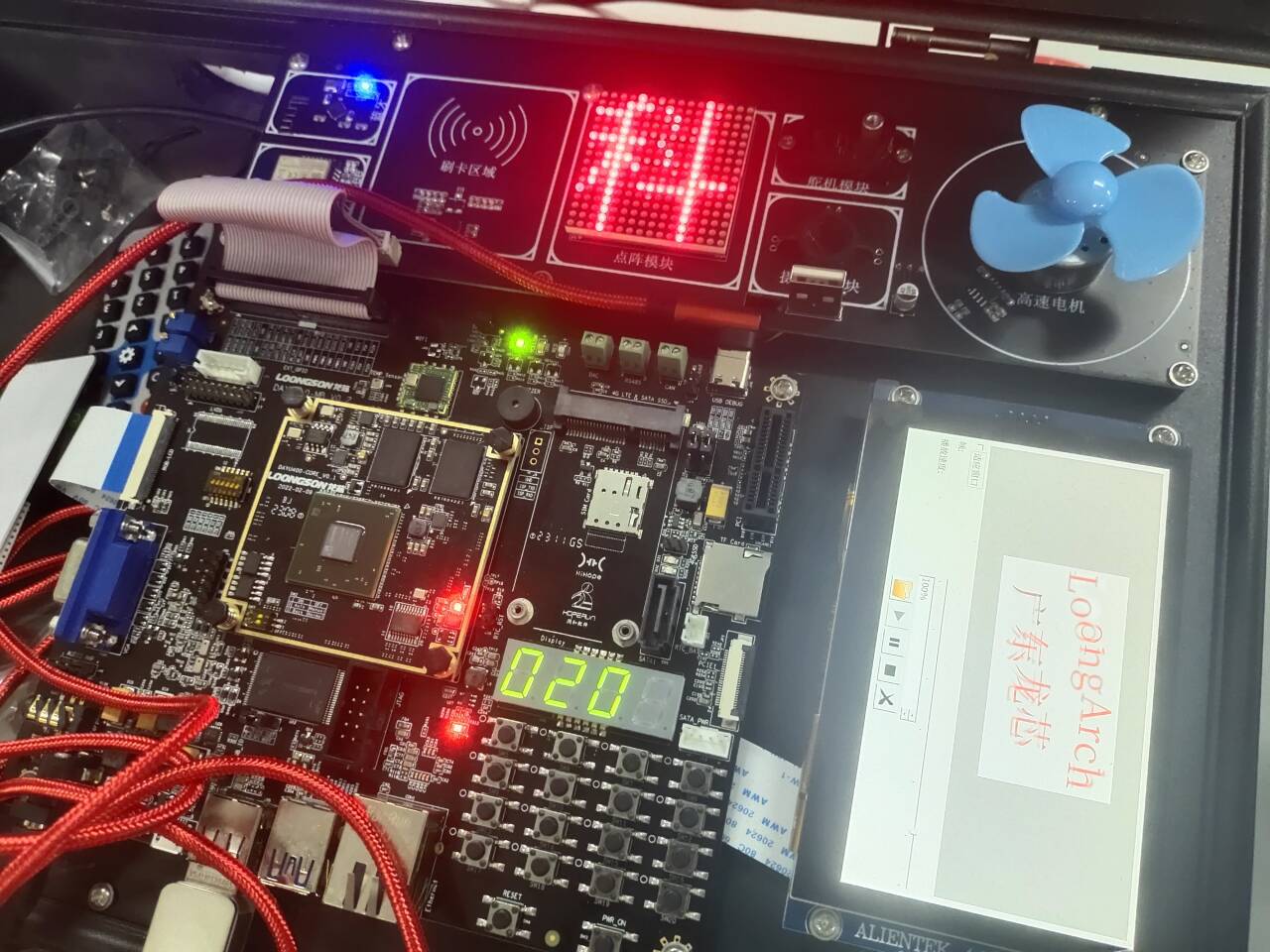




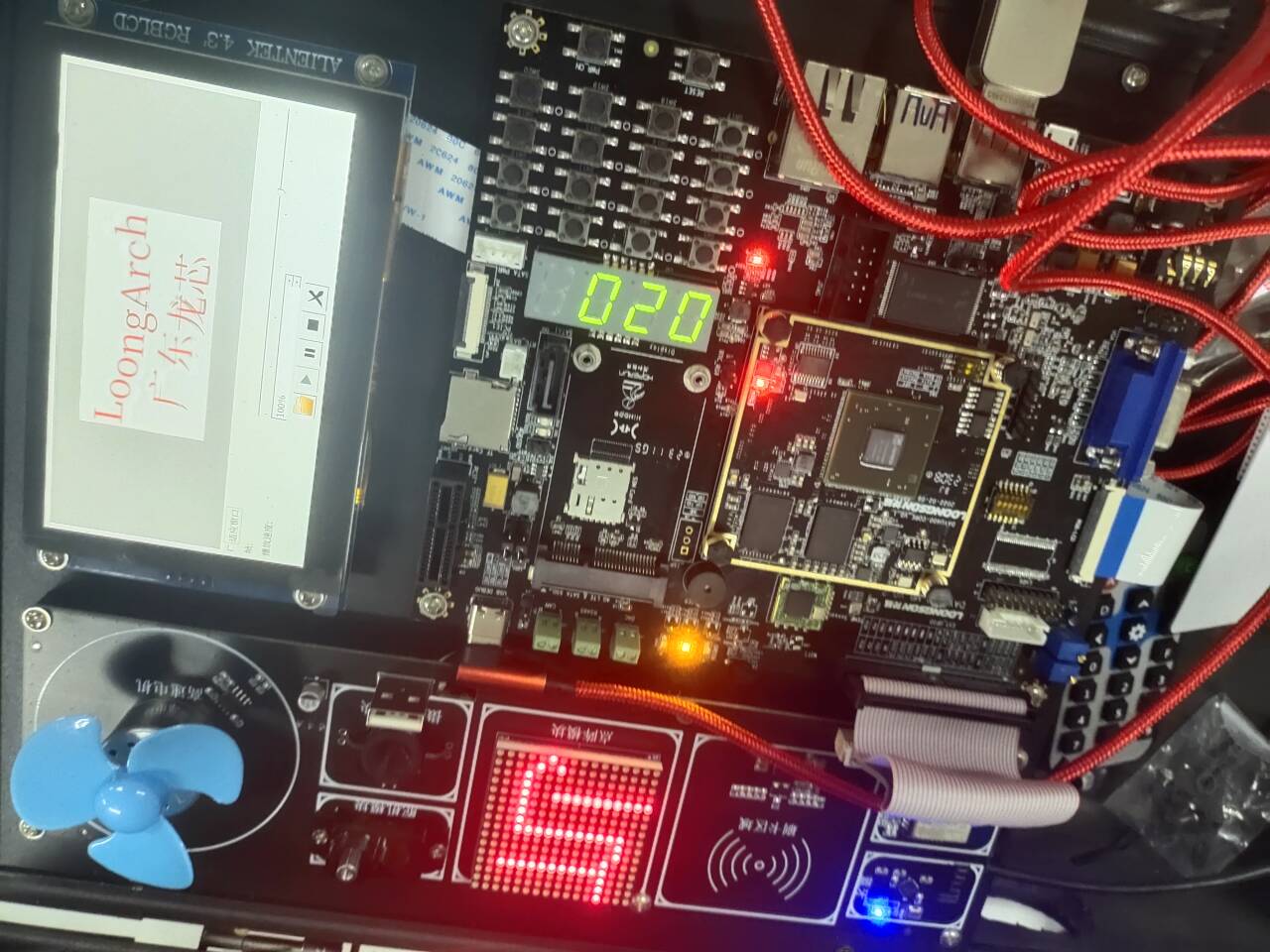


6.4 功能使用

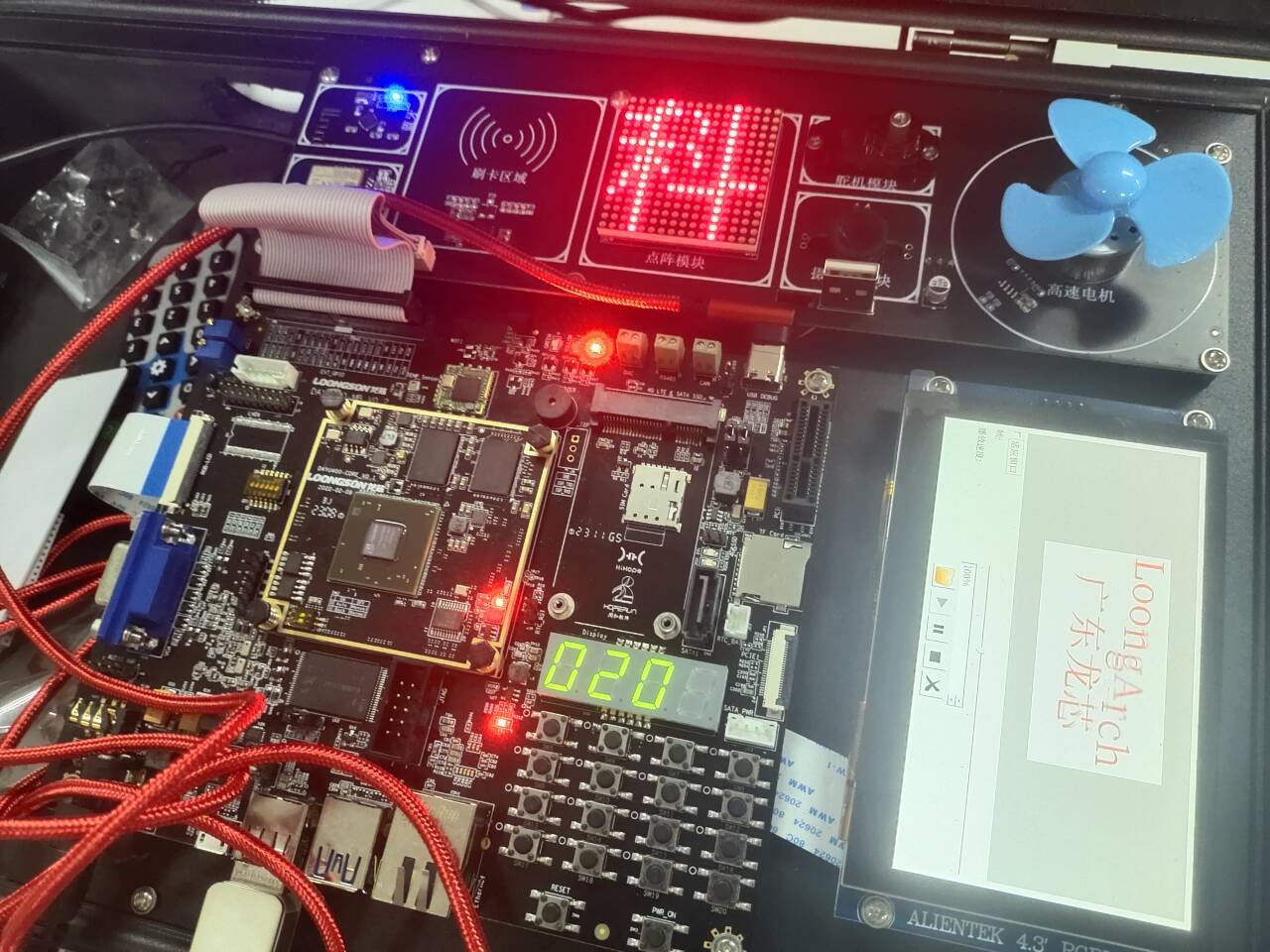
绿灯闪烁：



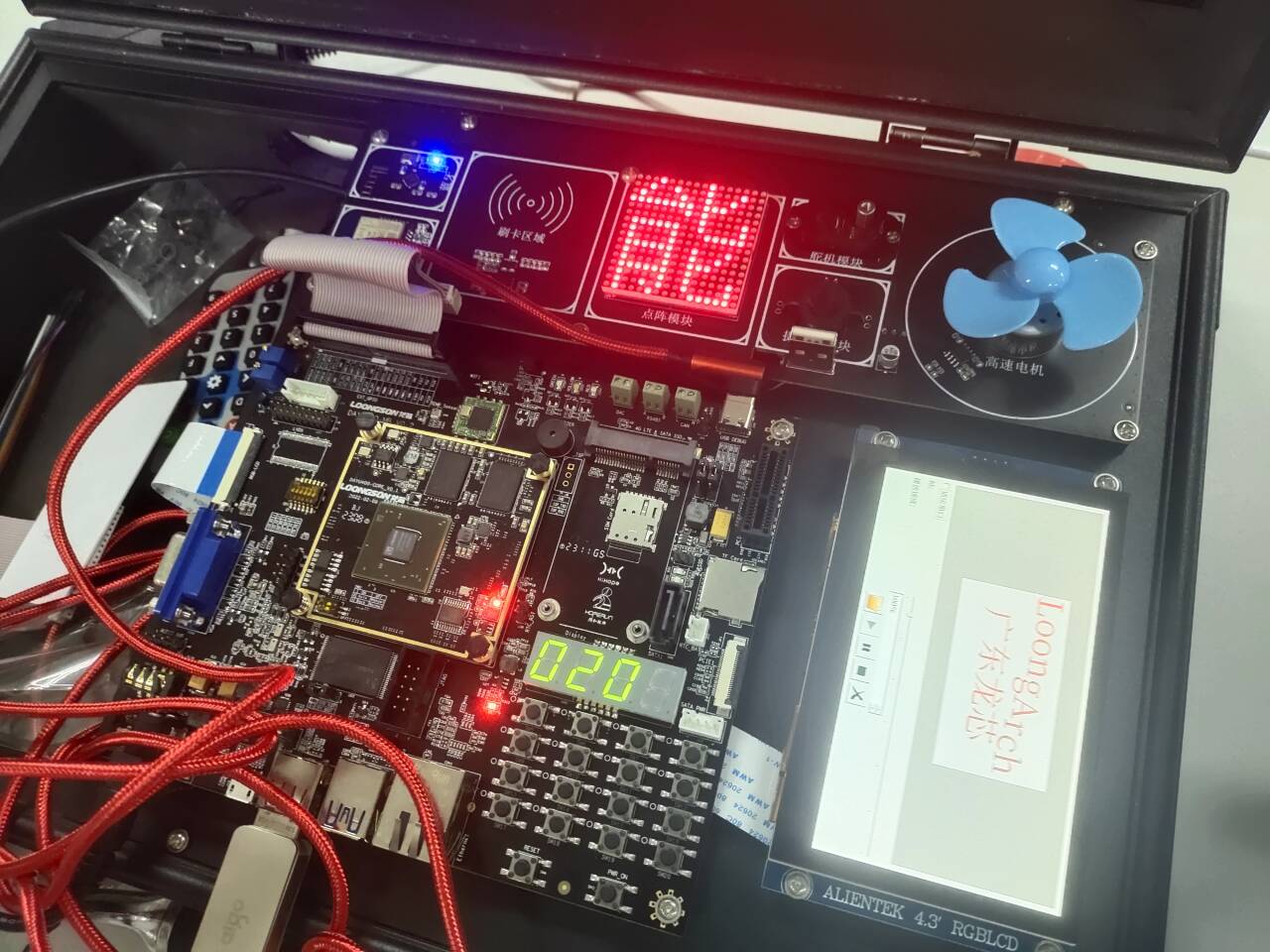
黄灯闪烁：



红灯闪烁：



6.5 暂停与退出



7.总结

与龙芯2k500打了一个学期的交道，经过为期16周的艰难探索，从编译内核和刷入系统开始，就遇到了很多困难，出现了不少莫名其妙的报错和文件缺少，在助教学长的帮助下，与同组同学和其他龙芯组同学共同努力，成功刷入系统。然而，在后续编译模块时又出现了问题，模块编译不出来，或者编译出来的模块无法安装在嵌入式平台上，这时候发现内核有问题，又换了个新的内核刷入系统。

在开始动手写最终成品的前几周，又把编译出来的模块挨个试了一遍，发现能用的模块实在不多，只有键盘、NFC、点阵、LED、蜂鸣器、电机、七段数码管、温湿度传感器。这个时候已经是第12、13周了，陀螺仪模块安装后系统就会卡死，红外模块根本接收不到任何红外信号（应该是在input文件夹下的某个event，event0 or event2？），一开始以为是遥控器没电了，后来对接收红外模块的组件进行cat没有任何输出信息，然而如果cat屏幕对应的event，当触摸屏幕时会有接收到的信息，并且手机的红外模块是能接收到遥控器的红外信号。其他组的同学在最后发现了陀螺仪要在某一个模块后安装，当时也没有时间再去加陀螺仪的功能了。至于声卡和摄像头，刷入的系统根本不支持这些，如果刷入支持声卡和摄像头的系统，前面的模块无法正常使用。

最终，呈现出来的成品就是这篇实验报告所展示的，虽然在逻辑上说得过去，但比较粗糙，由于时间紧迫，也没有余力再去打磨一下这个系统和添加一些新的功能，或者使操作更加人性化、更合理。虽然困难重重，也确实有很多收获：

1.学会了点阵显示屏、电机控制、传感器读取等外设的应用方法，提升了硬件开发能力。

2.通过模块化设计，理解了如何将复杂系统分解为多个功能模块，并实现模块之间的协同工作。

3.在调试过程中，学会了使用调试工具和方法，解决了多项技术难题，增强了问题解决能力。

4.在实验过程中，与同学们分工合作，进行了有效的沟通与协调，提升了团队协作能力。

5.在设计系统功能时，结合实际需求和技术条件，提出了多项创新设计方案，并通过实践验证了其可行性。