# 单片机大作业报告

17343031 辜宇然

小组成员

17343031 辜宇然
17343050 黄昱珲
17343075 刘皓铧
17343092 潘鹏程

一．大作业完成情况简述

对于本课程的大作业，我们完成了所有的必做（非拓展内容），其中实时时钟部分由于我们的 STC 单片机板上没有 RTC 时钟，故我们采用说明的方式完成（因为我们在课堂上已完成 RTC 时钟实验的检查，并且同时完成 RTC 时钟的实验报告，所以实际上我们已经完全掌握了该部分内容）。

二．大作业完成内容及其要求

1、对通道进出权限的管理

进出通道的方式就是对可以进出该通道的人进行进出方式的授权，进出方式通常有密码、读卡、其它身份识别几种方式。基本功能需要实现密码开门，持卡人必须输入密码正确才能开门

4、异常报警功能

在异常情况下可以实现报警，如：非法侵入、门超时未关、温湿度超过报警界限（扩展）等

6.有实时时钟功能

7.有键盘、显示屏

9.有网络功能：蓝牙、WIFI

三．项目实现

1）实现器材

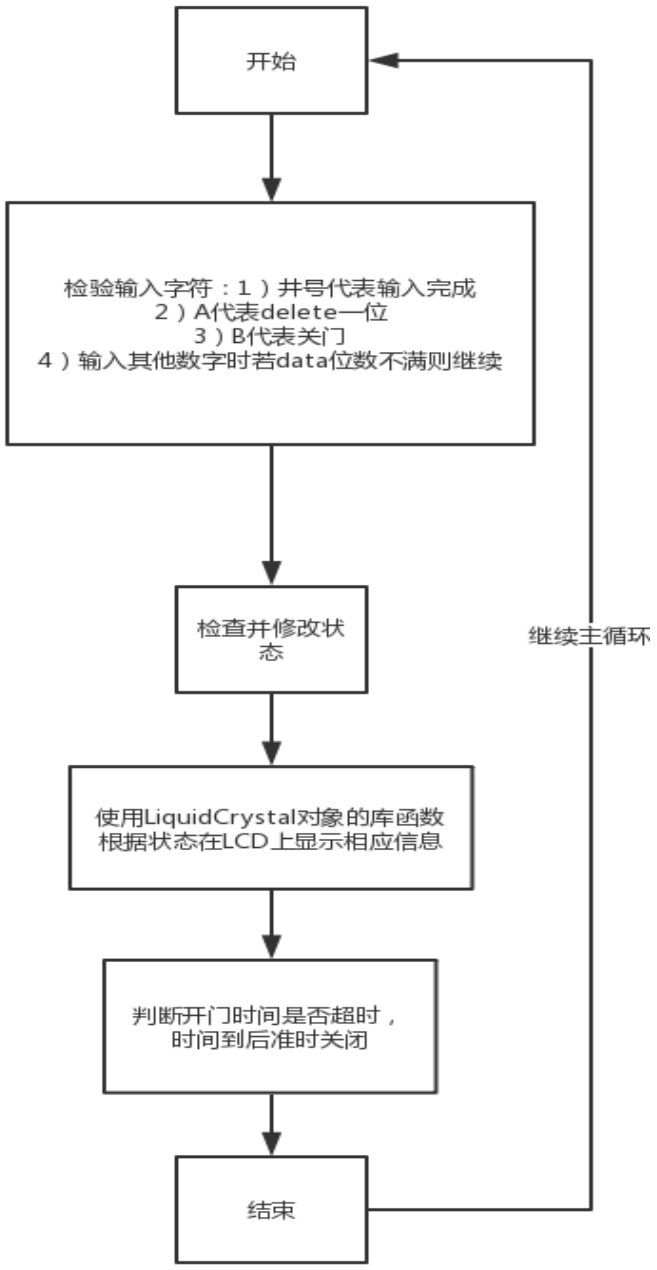
为了熟悉单片机领域的各种设备，针对不同功能我们使用了 Arduino 板和 stc51 单片机两种设备分功能实现。

2)具体实现

1.功能一和四由 Arduino 实现，其中 LCD 屏显示的控制需要借助头文件<LiquidCrystal.h>，通过 Serial 可以调用 Arduino 的串口库函数，在这里简单介绍以便理解代码：

```
Serial.begin(); //开启串行通信接口并设置通信波特率
Serial.end();    //关闭通信串口
Serial.available();//判断串口缓冲器是否有数据装入
Serial.read();    //读取串口数据
Serial.peek();    //返回下一字节(字符)输入数据，但不删除它
Serial.flush();    //清空串口缓存
Serial.print();    //写入字符串数据到串口
Serial.println();   //写入字符串数据+换行到串口
Serial.write();     //写入二进制数据到串口
Serial.SerialEvent();//read 时触发的事件函数
Serial.readBytes(buffer,length);//读取固定长度的二进制流
Serial.println(incomingByte, DEC);//打印接到数据十进制表示的 ascii 码。
HEX 十六进制表示
```

由于每一时刻输入密码的状态和门锁的状态唯一，故使用枚举变量 PasswordState 和 DoorState，这两个变量代表所有的情况，针对不同的情况我们在主循环函数 loop 中进行检验和做出相应的功能，其中 loop 函数中的各功能函数完成的功能如下流程图：




2. 由前所述，功能六，即实时时钟功能，由于器材的局限，使用说明的方式进行完成。
原理：

在 STC 开发板上有时钟芯片 RTC，通过 keyboard 输入通过 I2C 总线读入数据，通过中断设置芯片时间。

读取 RTC 函数

```
void  ReadRTC(void)
{
      u8    tmp[3];
```

```
	ReadNbyte(2, tmp, 3);
	second = ((tmp[0] >> 4) & 0x07) * 10 + (tmp[0] & 0x0f);
	minute = ((tmp[1] >> 4) & 0x07) * 10 + (tmp[1] & 0x0f);
	hour   = ((tmp[2] >> 4) & 0x03) * 10 + (tmp[2] & 0x0f);
}
```

写入 RTC 函数

```
void  WriteRTC(void)
{
	u8    tmp[3];

	tmp[0] = ((second / 10) << 4) + (second % 10);
	tmp[1] = ((minute / 10) << 4) + (minute % 10);
	tmp[2] = ((hour / 10) << 4) + (hour % 10);
	WriteNbyte(2, tmp, 3);
}
```

解释：

单片机通过这两个函数控制 RTC 芯片中时间变量

tmp[3]表示时分秒三个变量位

Hour minute second 分别表示三个 RTC 变量

这里需要说明，tmp 是一个专门服务于 RTC 芯片时间显示的寄存器组，一共有 7 个：

表 1　日历、时间寄存器及其控制字

| 寄存器名称 | 命令字 写操作 | 命令字 读操作 | 取值范围 | 各位内容 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 秒寄存器 | 80H | 81H | 00～59 | CH | | 10SEC | | | SEC | | |
| 分寄存器 | 82H | 83H | 00～59 | 0 | | 10MIN | | | MIN | | |
| 时寄存器 | 84H | 85H | 01～12 或 00～23 | 12/24 | 0 | 10 HR | | | HR | | |
| 日寄存器 | 86H | 87H | 01～28,29,30,31 | 0 | 0 | 10DATE | | | DATE | | |
| 月寄存器 | 88H | 89H | 01～12 | 0 | 0 | 0 | 10M | | MONTH | | |
| 周寄存器 | 8AH | 8BH | 01～07 | 0 | 0 | 0 | 0 | 0 | DAY | | |
| 年寄存器 | 8CH | 8DH | 00～99 | 10YEAR | | | | | YEAR | | |

Tmp[0]控制秒，由于 second 变量前四位是秒钟十位，后四位是秒钟个位，由于 tmp 遵循高低四位设置方式，故写为 tmp[0] = ((second / 10) << 4) + (second % 10) ，下同 tmp[1]控制分，tmp[2]控制时，tmp[3]控制日，tmp[4]控制月，tmp[6]控制年，这些都与上述一致，但 tmp[5]是周寄存器，专门显示星期，我们此次不用它，故置为 0

再解释一下，由于 second 最大值为 60，所以高四位最多使用位数为三位，故 second = ((tmp[0] >> 4) & 0x07) * 10 + (tmp[0] & 0x0f);

所以为了达到预置年月日的效果：首先在本地变量区定义 RTC 变量
u8 year,month,day

修改上述函数为：

```
void  ReadRTC(void)
{
      u8    tmp[7];

      ReadNbyte(2, tmp, 7);

      second = ((tmp[0] >> 4) & 0x07) * 10 + (tmp[0] & 0x0f);
      minute = ((tmp[1] >> 4) & 0x07) * 10 + (tmp[1] & 0x0f);
      hour   = ((tmp[2] >> 4) & 0x03) * 10 + (tmp[2] & 0x0f);
day = ((tmp[3] >> 4) & 0x07） * 10 + (tmp[3] & 0x0f);
month = ((tmp[4] >> 4) & 0x01) * 10 + (tmp[4] & 0x0f);
year = ((tmp[6] >> 4) & 0x0f) * 10 + (tmp[6] & 0x0f);
}
```

写入 RTC 函数

```
void  WriteRTC(void)
{
      u8    tmp[7];

      tmp[0] = ((second / 10) << 4) + (second % 10);
      tmp[1] = ((minute / 10) << 4) + (minute % 10);
      tmp[2] = ((hour / 10) << 4) + (hour % 10);
tmp[3] = ((day / 10) << 4) + (day % 10)
tmp[4] = 0x80 || ((month / 10) << 4) + (month % 10);
tmp[5] = 0;
tmp[6] = ((year / 10) << 4) + (year % 10);
      WriteNbyte(2, tmp, 7);
}
```

此处将 tmp[4]最高位置 1 后可自行设置 rtc 芯片世纪位

3．功能 7：有键盘、显示屏

显示屏功能已在功能 1 和 4 中实现，故现在实现键盘功能即可。键盘的使用原理很简单，不断检验所有的行列的电平判断是否有键按下，若有键按下则对应的行列位置 1，则每个特殊的键盘位对应一个特殊的数字，根据数字解析出对应的键盘编码即可。

4．功能 9：蓝牙、WIFI

对于 WIFI 模块，通过 uart.c 库函数的调用实现功能，其中原理为发送数据帧->接收数据帧->保存至 buffer->buffer 满或到达指定位数则处理信息->关闭串口中断->清空 buffer->打开串口接收中断。

在 main.c 中，我们在初始化串口后，通过调用 uart.c 的 Hand()函数进行连接及其确认，通过 U1SendString 和 U2SendString 进行数据的发送，从而达到 WIFI 控制单片机 WIFI 模块的效果。

检验：

通过网络调试助手启动，具体验证已经在期中项目中发过 ESP8266 控制 L298N 进而控制发动机的部分拍录视频演示过，由于此次未携带 STC51 板归家故不再演示，不过原理和代码是几乎完全一致的。

我们使用的是 TCP 协议连接，故只需要创建 tcp client，ESP8266 作为 server，我们的手机此时作为 server，在 192.168.4.1:5000 接收 server 的状态信息并向其发送指令。


3）功能展示






四．总结

通过一学期的学习，我们学会了单片机的原理，学会了如何使用单片机进行各种模块的控制，见识到了单片机的魅力。这次大作业就是我们在一学期的单片机课程学习后的一次小小的总结，同时我们也尝试进行一些小小的拓展，通过使用不同的设备和模块来丰富我们对单片机乃至对嵌入式软件的认识，不得不感谢老师和相伴的同学所共同创造的这个学习氛围。

五. 代码

1.key.c #键盘代码

```
#include "key.h"

/*********************************************************************************
*********
 * 描  述 : 按键扫描函数
 * 入  参 : 无
 * 返回值 : 哪个按键按下的对应值

*********************************************************************************
********/
uint8 KeyScan(void)
{
      uint8 X_temp,Y_temp,temp;

       X_temp=0xF0;          //列值赋初值
       Y_temp=0x0F;          //行值赋初值

       P2M1 &= 0x3F;     P2M0 |= 0xC0;        //设置 P2.6~P2.7 为强推挽输出
       P4M1 &= 0x0F;     P4M0 |= 0xF0;        //设置 P4.4~P4.7 为强推挽输出
       P5M1 &= 0xF3;     P5M0 |= 0x0C;        //设置 P5.2~P5.3 为强推挽输出


    ROW1=1;ROW2=1;ROW3=1;ROW4=1;  //行置高
        COL1=0;COL2=0;COL3=0;COL4=0;  //列置低

            //所用到行 IO 口配置为输入，进行检测
        DelayMS(10);
        P4M1 &= 0x0F;    P4M0 &= 0x0F;        //设置 P4.4~P4.7 为准双向口
    DelayMS(10);

            if(ROW1 == 0)          //检测行 1 电平是否为低电平
            {
                  DelayMS(10);
                  if(ROW1 == 0)
                  Y_temp &= 0x0E;
            }
            if(ROW2 == 0)          //检测行 2 电平是否为低电平
            {
```

```
            DelayMS(10);
            if(ROW2 == 0)
            Y_temp &= 0x0D;
        }
        if(ROW3 == 0)          //检测行 3 电平是否为低电平
        {
            DelayMS(10);
            if(ROW3 == 0)
            Y_temp &= 0x0B;
        }
        if(ROW4 == 0)          //检测行 4 电平是否为低电平
        {
            DelayMS(10);
            if(ROW4 == 0)
            Y_temp &= 0x07;
        }

    P2M1 &= 0x3F;   P2M0 |= 0xC0;        //设置 P2.6~P2.7 为强推挽输出
    P4M1 &= 0x0F;   P4M0 |= 0xF0;        //设置 P4.4~P4.7 为强推挽输出
    P5M1 &= 0xF3;   P5M0 |= 0x0C;        //设置 P5.2~P5.3 为强推挽输出

        ROW1=0;ROW2=0;ROW3=0;ROW4=0;  //行置低
    COL1=1;COL2=1;COL3=1;COL4=1;  //列置高

        //所用到列 IO 口配置为输入，进行检测
        DelayMS(10);
    P2M1 &= 0x3F;   P2M0 &= 0x3F;        //设置 P2.6~P2.7 为准双向口
    P5M1 &= 0xF3;   P5M0 &= 0xF3;        //设置 P5.2~P5.3 为准双向口
DelayMS(10);

        if(COL1 == 0)          //检测列 1 电平是否为低电平
        {
            DelayMS(10);
            if(COL1 == 0)
            X_temp &= 0xE0;
        }
        if(COL2 == 0)          //检测列 2 电平是否为低电平
        {
            DelayMS(10);
            if(COL2 == 0)
            X_temp &= 0xD0;
```

```
        }
        if(COL3 == 0)          //检测列 3 电平是否为低电平
        {
              DelayMS(10);
              if(COL3 == 0)
              X_temp &= 0xB0;
        }
        if(COL4 == 0)          //检测列 4 电平是否为低电平
        {
              DelayMS(10);
              if(COL4 == 0)
              X_temp &= 0x70;
        }

        //将行值和列值合并，得到按键对应的编码值，该值与 16 个按键一一对应
        temp = X_temp|Y_temp;
    temp = ~temp;

        //将按键检测的原始编码值解析对应按键值信息
  switch (temp)
  {
              case 0x11:return 1;     //1
              case 0x21:return 2;     //2
              case 0x41:return 3;     //3
              case 0x81:return 4;     //4
              case 0x12:return 5;     //5
              case 0x22:return 6;     //6
              case 0x42:return 7;     //7
              case 0x82:return 8;     //8
              case 0x14:return 9;     //9
              case 0x24:return 10;    //0
              case 0x44:return 11;    //a
              case 0x84:return 12;    //b
              case 0x18:return 13;    //c
              case 0x28:return 14;    //d
              case 0x48:return 15;  //e
              case 0x88:return 16;    //f
              default:  return 0;
  }
}
```

2.uart.c #串口代码

```
#include "key.h"

/*********************************************************************************
*********
 * 描  述 : 按键扫描函数
 * 入  参 : 无
 * 返回值 : 哪个按键按下的对应值

*********************************************************************************
********/
uint8 KeyScan(void)
{
      uint8 X_temp,Y_temp,temp;

       X_temp=0xF0;          //列值赋初值
       Y_temp=0x0F;          //行值赋初值

       P2M1 &= 0x3F;     P2M0 |= 0xC0;        //设置 P2.6~P2.7 为强推挽输出
       P4M1 &= 0x0F;     P4M0 |= 0xF0;        //设置 P4.4~P4.7 为强推挽输出
       P5M1 &= 0xF3;     P5M0 |= 0x0C;        //设置 P5.2~P5.3 为强推挽输出


    ROW1=1;ROW2=1;ROW3=1;ROW4=1;  //行置高
        COL1=0;COL2=0;COL3=0;COL4=0;  //列置低

            //所用到行 IO 口配置为输入，进行检测
        DelayMS(10);
        P4M1 &= 0x0F;   P4M0 &= 0x0F;        //设置 P4.4~P4.7 为准双向口
    DelayMS(10);

            if(ROW1 == 0)          //检测行 1 电平是否为低电平
            {
                  DelayMS(10);
                  if(ROW1 == 0)
                  Y_temp &= 0x0E;
            }
            if(ROW2 == 0)          //检测行 2 电平是否为低电平
            {
                  DelayMS(10);
                  if(ROW2 == 0)
```

```
            Y_temp &= 0x0D;
        }
        if(ROW3 == 0)          //检测行 3 电平是否为低电平
        {
            DelayMS(10);
            if(ROW3 == 0)
            Y_temp &= 0x0B;
        }
        if(ROW4 == 0)          //检测行 4 电平是否为低电平
        {
            DelayMS(10);
            if(ROW4 == 0)
            Y_temp &= 0x07;
        }

    P2M1 &= 0x3F;   P2M0 |= 0xC0;        //设置 P2.6~P2.7 为强推挽输出
    P4M1 &= 0x0F;   P4M0 |= 0xF0;        //设置 P4.4~P4.7 为强推挽输出
    P5M1 &= 0xF3;   P5M0 |= 0x0C;        //设置 P5.2~P5.3 为强推挽输出

        ROW1=0;ROW2=0;ROW3=0;ROW4=0;  //行置低
    COL1=1;COL2=1;COL3=1;COL4=1;  //列置高

        //所用到列 IO 口配置为输入，进行检测
        DelayMS(10);
    P2M1 &= 0x3F;   P2M0 &= 0x3F;        //设置 P2.6~P2.7 为准双向口
    P5M1 &= 0xF3;   P5M0 &= 0xF3;        //设置 P5.2~P5.3 为准双向口
DelayMS(10);

        if(COL1 == 0)          //检测列 1 电平是否为低电平
        {
            DelayMS(10);
            if(COL1 == 0)
            X_temp &= 0xE0;
        }
        if(COL2 == 0)          //检测列 2 电平是否为低电平
        {
            DelayMS(10);
            if(COL2 == 0)
            X_temp &= 0xD0;
        }
        if(COL3 == 0)          //检测列 3 电平是否为低电平
```

```
        {
            DelayMS(10);
            if(COL3 == 0)
            X_temp &= 0xB0;
        }
        if(COL4 == 0)          //检测列 4 电平是否为低电平
        {
            DelayMS(10);
            if(COL4 == 0)
            X_temp &= 0x70;
        }

        //将行值和列值合并，得到按键对应的编码值，该值与 16 个按键一一对应
        temp = X_temp|Y_temp;
    temp = ~temp;

        //将按键检测的原始编码值解析对应按键值信息
    switch (temp)
    {
            case 0x11:return 1;     //1
            case 0x21:return 2;     //2
            case 0x41:return 3;     //3
            case 0x81:return 4;     //4
            case 0x12:return 5;     //5
            case 0x22:return 6;     //6
            case 0x42:return 7;     //7
            case 0x82:return 8;     //8
            case 0x14:return 9;     //9
            case 0x24:return 10;    //0
            case 0x44:return 11;    //a
            case 0x84:return 12;    //b
            case 0x18:return 13;    //c
            case 0x28:return 14;    //d
            case 0x48:return 15;  //e
            case 0x88:return 16;    //f
            default:  return 0;
    }
}
```

3.LCD.cpp #液晶显示屏

```
#include <LiquidCrystal.h>
```

```
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
String userInput;
void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  Serial.begin(9600);
}

int door_open_time = 0;
int password_wrong_time = 0;
enum PasswordState {
  ENTER, CORRECT, WRONG
};
enum DoorState {
  CLOSE, OPEN, TIMEOUT
};
PasswordState password_state = ENTER;
DoorState door_state = CLOSE;
const int MAX_DOOR_OPEN_TIME = 5000; // 最大开门时间 5 秒
const int PROMPT_DELAY = 5000;
String data;
String password = "17343075"; // 开门的密码
const char* empty_line = "                ";
const char* mask = "***************";
void loop() {
  passwordVerify();
  checkPrompt();
  displayMessage();
  checkDoor();
}

void checkPrompt() {
  if (password_state == WRONG) {
    if (millis() - password_wrong_time > PROMPT_DELAY) {
      password_state = ENTER;
    }
  }
}

void lcdPrintLine(String msg, int line) {
  lcd.setCursor(0, line);
```

```
  lcd.print(msg);
  if (msg.length() < 16)
    lcd.print(empty_line + msg.length());
}

void displayMessage() {
  switch (door_state) {
    case CLOSE: {
        switch (password_state) {
          case WRONG:
            lcdPrintLine("Wrong password!", 0);
            lcdPrintLine("Try again", 1);
            break;
          case ENTER:
          default:
            lcdPrintLine("Enter password:", 0);
            lcdPrintLine(mask + (16 - data.length()), 1);
            break;
        }
      }
      break;
    case OPEN:
      lcdPrintLine("Door is open", 0);
      lcdPrintLine("", 1);
      break;
    case TIMEOUT:
      lcdPrintLine("Open timeout!", 0);
      lcdPrintLine("Close door now!", 1);
      break;
    default:
      break;
  }
}

void timeoutDoor() {
  door_state = TIMEOUT;
  Serial.write("timeout");
}

void passwordWrong() {
  closeDoor();
```

```
  password_state = WRONG;
  password_wrong_time = millis();
  delay(10);
  Serial.write("fail");
}

void passwordCorrect() {
  openDoor();
  password_state = CORRECT;
  Serial.write("success");
}

void openDoor() {
  door_state = OPEN;
  door_open_time = millis(); // 记录开门时刻
  delay(10);
  Serial.write("open");
}

void closeDoor() {
  door_state = CLOSE;
  Serial.write("close");
}

void checkDoor() {
  if (door_state == OPEN) {
    if (millis() - door_open_time > MAX_DOOR_OPEN_TIME) {
      // 开门时间超时
      timeoutDoor();
    }
  }
}

void passwordVerify() {
  if (Serial.available()) {
    char ch = (char) Serial.read();
    switch (ch) {
      case '#':
        if (door_state == CLOSE) {
          if (data == password) {
            passwordCorrect();
```

```
          } else {
            passwordWrong();
          }
          data = "";
        }
        break;
      case 'A':
        data = data.substring(0, data.length() - 1);
        break;
      case 'B':
        closeDoor(); // 模拟关门
        break;
      default:
        if (ch >= '0' && ch <= '9' && door_state == CLOSE) {
          password_state = ENTER;
          String new_data = String(ch);
          data += new_data;
        }
        break;
    }
  }
}
```

4.main.c #主函数

```
#include "uart.h"                    // 串行通信函数头文件
#include "delay.h"
#include "key.h"
/**********************
引脚别名定义
**********************/
sbit RST = P2^0;     //复位引脚用 IO 口
sbit LED_B=P0^6;         //蓝色 LED 用 IO 口 P06,定义为 LED1
sbit LED_R=P0^7;     //红色 LED 用 IO 口 P07,定义为 LED2

/*****************************************************************
AT 指令集
*****************************************************************/
char code str1[]="AT\r\n";                                          //
联机指令，返回"OK"
char code str2[]="AT+CWMODE=1\r\n";
//  设置 ESP8266 的工作模式，返回"OK"或者"no change"
```

```
char code str3[]="AT+CWJAP=\"ChinaNet\",\"aaaaaaaa\"\r\n";
//  连接到 WiFi 热点，ChinaNet aaaaaaaa
char code str4[]="AT+CIFSR\r\n";                                        //
      本机 IP 地址查询指令
char code str5[]="AT+CIPSTART=\"TCP\",\"192.168.3.230\",5000\r\n";    //  连接
到 TCP 服务器，返回“Linked”
char code str6[]="AT+CIPSEND=6\r\n";
                                                    //  发送数据指令
char code str7[]="hello!\r\n";
char code str8[]="AT+CIPSERVER=1,5000\r\n";
                                    //  建立 TCP 服务器，开放端口 5000

char code str9[]="AT+CIPMUX=1\r\n";
                                                      //    打开多连接
char code str10[]="AT+RST\r\n";
                                                                //  软件复位
char code str11[]="AT+CIPSEND=0,15\r\n";
                                                    //  发送数据指令,基于多路连接
模式
char code str12[]="Command Executed!\r\n";
                                              //  数据内容
char tmp1[]="x\r\n";
uint8 keymap[16] = {'1', '2', '3', 'A', '4', '5', '6', 'B', '7', '8', '9',
'C', '*', '0', '#', 'D'};
/*******************************************************************************
 * 描  述 : 主函数
 * 入  参 : 无
 * 返回值 : 无
 *******************************************************************************/
int main()
{
      ////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
////////////////////////////////////////////
      uint8 key, last_key = 0, ch;
      P0M1 &= 0x3F;     P0M0 &= 0x3F;                   //设置 P0.6~P0.7 为准双向
口
```

```
    P1M1 &= 0xFC;     P1M0 &= 0xFC;                 //设置 P1.0~P1.1 为准双向
口(串口 2)
    P2M1 &= 0xFE;     P2M0 |= 0x01;                 //设置 P2.0 为推挽输出口
    P3M1 &= 0xFC;     P3M0 &= 0xFC;                 //设置 P3.0~P3.1 为准双向
口(串口 1)

    RST = 1;                                        //复位引脚置高
    UartInit();
    // 初始化串口 1 串口 2
    ES = 1;                                         // 串口 1 中断打开
    IE2 |= 0x01;                                    // 串口 2 中断打开
    EA = 1;                                         // 总中断打开
    DelayMS(1000);
// 延时一段时间，让 ESP8266 启动
    DelayUS(100);

    CLR_Buf();                                          //清除串口 2 缓存内
容

    while(!Hand("OK"))                              //判断是否握手成功,如果不
成功延时一会,再发送 AT 握手指令
    {
        U2SendString(str1);                         //通过串口 2 发送联机指
令
        DelayMS(500);                               //延时不可少
    }
    CLR_Buf();                                      //清除串口 2 缓存内容
    LED_R = 0;                                      //点亮开发板红色指示灯

    while(!(Hand("OK")|Hand("no change")))          //判断是否设置成功，如不成
功，延时后再次发送
    {
        U2SendString(str2);                         //发送设置 ESP8266 工作
模式指令
        DelayMS(500);                                       //延时不可少
    }
    CLR_Buf();                                      //清除串口 2 缓存内容

    while(!Hand("OK"))
            //判断是否连接 WiFi 路由器，如不成功，延时后再次发送
    {
```

```
            U2SendString(str3);                                  //连接到 WiFi 热点，
lces 为热点名称，88518851 为密码；连接成功返回“OK”
            DelayMS(2000);                                       //延时不可少
      }
      LED_B = 0;                                          //点亮开发板蓝色指示灯
      CLR_Buf();                                          //清除串口 2 缓存内容

      while(!Hand("OK"))                                  //判断是否连接 TCP sever，
如不成功，延时后再次发送
      {
            U2SendString(str5);                                  //连接到 TCP 服务器，返
回“OK”
            DelayMS(3000);                                       //延时不可少
      }
      CLR_Buf();                                          //清除串口 2 缓存内容

      LED_R = 1;                                          //熄灭开发板红色指示灯
      LED_B = 1;                                          //熄灭开发板蓝色指示灯

      while(!Hand("SEND OK"))
      //判断是否发送数据成功，如不成功，延时后再次发送
      {

            U2SendString(str6);                                  //数据发送指令
            DelayMS(100);                                              //延时不可少
            U2SendString(str7);                                  //数据内容
      "hello!"
            DelayMS(500);                                        //延时不可少
      }
      CLR_Buf();                                          //清除串口 2 缓存内容

      while (1) {
            key = KeyScan();
            if (UARTHand("success")) {
                  while(!Hand("SEND OK"))
                  //判断是否发送数据成功，如不成功，延时后再次发送
                  {
                        U2SendString("AT+CIPSEND=7\r\n");
//数据发送指令
                        DelayMS(100);                                          //
延时不可少
```

```
                U2SendString("success\r\n");
                DelayMS(500);                                   //延时不可
少
            }
            CLR_Buf();
        } else if (UARTHand("fail")) {
            while(!Hand("SEND OK"))
            //判断是否发送数据成功，如不成功，延时后再次发送
            {

                U2SendString("AT+CIPSEND=4\r\n");
//数据发送指令
                DelayMS(100);                                       //
延时不可少
                U2SendString("fail\r\n");
                DelayMS(500);                                   //延时不可
少
            }
            CLR_Buf();
        } else if (UARTHand("close")) {
            while(!Hand("SEND OK"))
            //判断是否发送数据成功，如不成功，延时后再次发送
            {

                U2SendString("AT+CIPSEND=5\r\n");
//数据发送指令
                DelayMS(100);                                       //
延时不可少
                U2SendString("close\r\n");
                DelayMS(500);                                   //延时不可
少
            }
            CLR_Buf();
        } else if (UARTHand("timeout")) {
            while(!Hand("SEND OK"))
            //判断是否发送数据成功，如不成功，延时后再次发送
            {

                U2SendString("AT+CIPSEND=7\r\n");
//数据发送指令
```

```
                        DelayMS(100);                                   //
延时不可少
                        U2SendString("timeout\r\n");
                        DelayMS(500);                                   //延时不可
少
                  }
                  CLR_Buf();
            }

            if (last_key != 0 && last_key == key) continue;
            last_key = key;
            ch = keymap[key - 1];
            tmpl[0] = ch;
            if (key)
                  UlSendData(ch);
      }
}
```