单片机期中项目报告

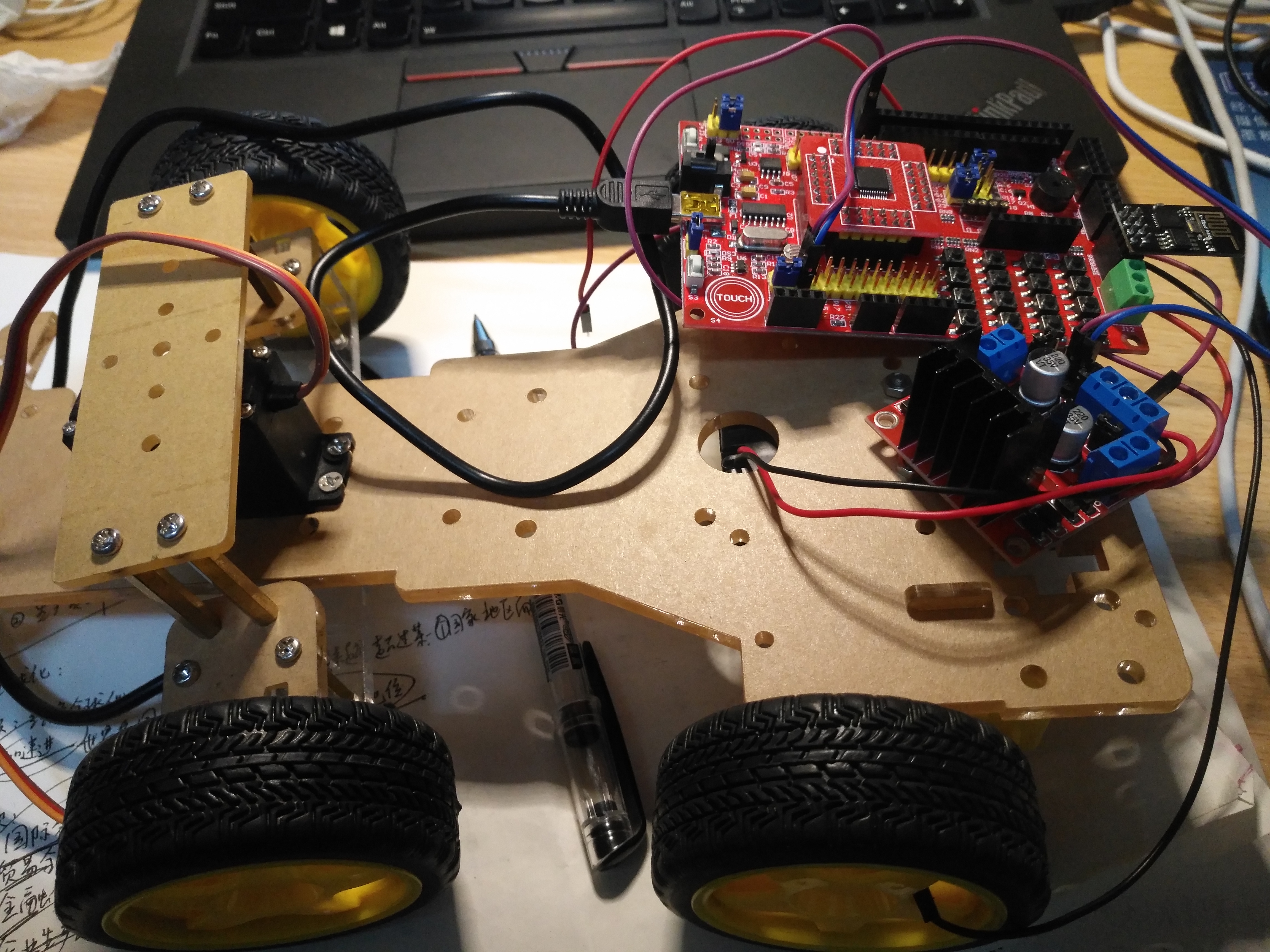
17343031 辜宇然

17343050 黄昱珲

17343075 刘皓铧

17343092 潘鹏程

一．简述



二．模块

①WIFI模块Esp8266

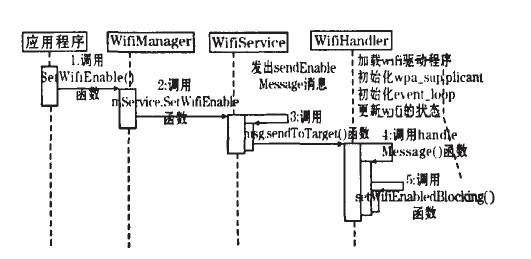
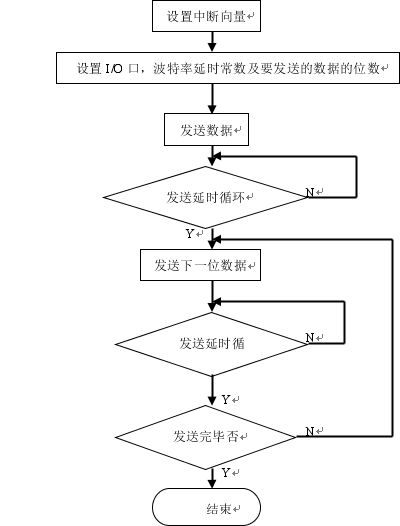
涉及原理：通信

原理简介：

根据我们的理解，通信是指两台设备通过一定方式连接后，进行数据传输。我们学习的是串口通信，实际上wifi模块的通信与其他通信的基本工作原理是一样的，以UART为例，大致流程为

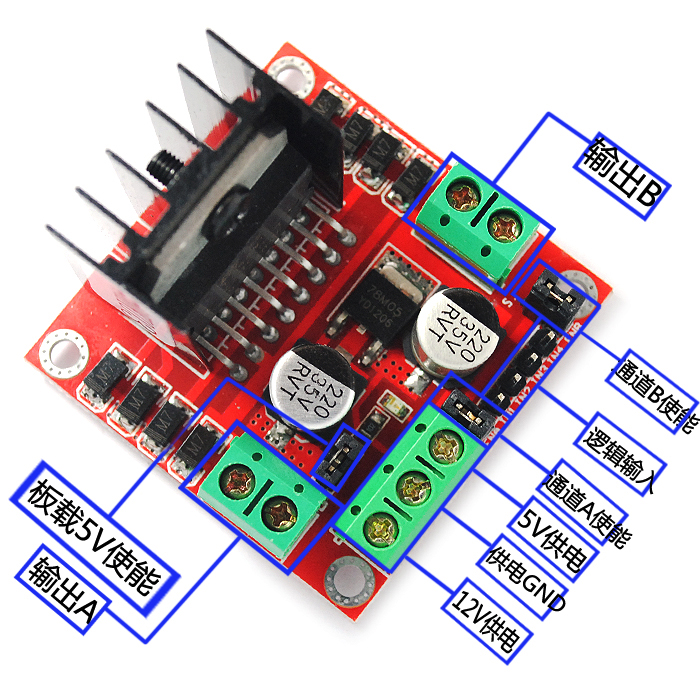
1.首先进行初始化，2.其次进行数据的发送，3.SBUF有数据时引发UART中断，4.接收者响应中断，同时自身中断处理（注意在自身中断时需要首先将TI/RI手动清零，不然会陷入死循环），5.接着发送下一位数据并由接收者处理...。

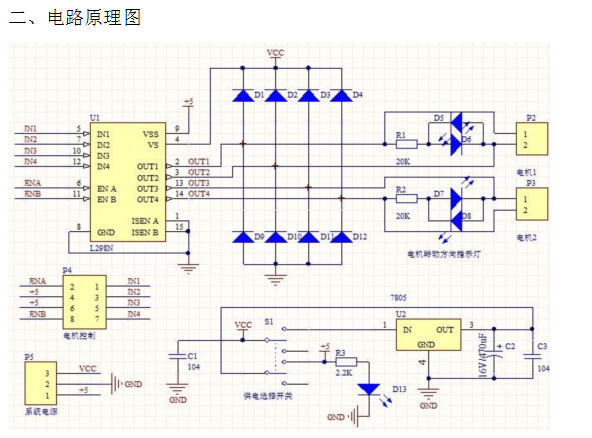
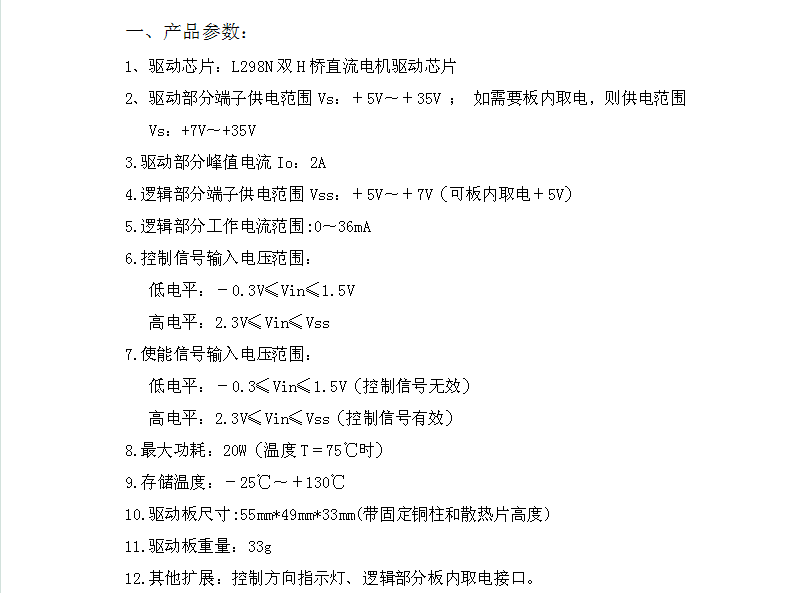
左图为串口通信模块，右图为wifi模块工作流程图。可以发现基本流程是一样的，我们只需要根据流程图进行程序编写。



②降压模块L298N

由于我们使用的马达依靠单片机本身电压无法驱动，因此我们选择两节+3.7v的ultraFire电池，并使用L298N电源控制模块进行降压。



由文档知，该直流电机的驱动板可驱动2路直流电机，使能端ENA\ENB为高电平时有效，控制方式及直流电机状态表如下：

|  |  |  |  |
| --- | --- | --- | --- |
| ENA | IN1 | IN2 | 直流电机状态 |
| 0 | x | x | 停止 |
| 1 | 0 | 0 | 制动 |
| 1 | 0 | 1 | 正转 |
| 1 | 1 | 0 | 反转 |
| 1 | 1 | 1 | 制动 |

测试程序（测试功能循环：电机正转2s，反转2s，自动加速正转，自动减速反转）

接口 P1.0 - IN1/P1.1 - IN2/P1.2 - ENA

电机两端分别接OUT1/OUT2

#include <stc51.h>

sbit IN1 = P1^0;

sbit IN2 = P1^1;

sbit ENA = P1^2;

void delay(unsigned int z);

void main() {

while(1) {

unsigned int i, cycle = 0, T=2048;

IN1 = 1;//正转

IN2 = 0;

for(i=0;i<200;i++) {

delay(10);//pwm占空比为50%，修改延时调整PWM脉冲

ENA=~ENA;

}

IN1 = 0；//反转

IN2 = 1;

for(i=0;i<100;i++){

delay（20);

ENA=~ENA;

}

IN1 = 1;//自动加速正转

IN2 = 0;

while(cycle != T){

ENA = 1;

delay\_us(cycle++);

ENA = 0;

delay\_us(T-cycle);

}

IN1 = 0;

IN2 = 1;

while(cycle != T) {

ENA = 1;

delay\_us(cycle++);

ENA = 0;

delay\_us(T-cycle);

}

}

}

//秒延时

void delay(unsigned int z) {

unsigned int x,y;

for(x = z;x > 0;x--)

for(y=110;y > 0;y --);

}

//微秒延时

void delay\_us(unsigned int aa) {

while (aa--);

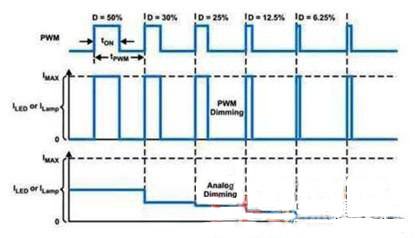
}

在使用时我们曾经遇到一些问题，后来经过研究发现是电平不一致的问题。实际上在将单片机与L298N连接的时候为了确保正常工作不仅应该将L298N的VCC和GND与电源正负极相连，还应该将单片机与L298N共地。

③pwm控制舵机

\*本模块因为只有舵机而无舵机模块，故在此详述原理。原计划为期中后实现，现已因项目改变为门禁系统而未硬件实现。

PWM:Pulse Width Modulation(脉冲宽度调制)。它通过对一系列脉冲的宽度进行调制，等效出所需要的波形（包含形状以及幅值），对模拟信号电平进行数字编码，也就是说**通过调节占空比的变化来调节信号、能量等的变化**，占空比就是指在一个周期内，信号处于高电平的时间占据整个信号周期的百分比，例如方波的占空比就是50%.

IMG_256

简单代码测试PWM

说明：

舵机原理

控制信号由接收机的通道进入信号调制芯片，获得直流偏置电压。它内部有一个基准电路，产生周期为20ms，宽度为1.5ms的基准信号，将获得的直流偏置电压与电位器的电压比较，获得电压差输出。最后，电压差的正负输出到电机驱动芯片决定电机的正反转。当电机转速一定时，通过级联减速齿轮带动电位器旋转，使得电压差为0，电机停止转动。

舵机的控制一般需要一个20ms左右的时基脉冲，该脉冲的高电平部分一般为0.5ms-2.5ms范围内的角度控制脉冲部分，总间隔为2ms。以180度角度伺服为例，那么对应的控制关系是这样的

   0.5ms--------------0度；

   1.0ms------------45度；

   1.5ms------------90度；

   2.0ms-----------135度；

   2.5ms-----------180度；

（1）舵机的追随特性

  假设现在舵机稳定在A点，这时候CPU发出一个PWM信号，舵机全速由A点转向B点，在这个过程中需要一段时间，舵机才能运动到B点。

保持时间为Tｗ

        当Tｗ≥△T时，舵机能够到达目标，并有剩余时间；

当Tｗ≤△T时，舵机不能到达目标；

理论上：当Tｗ=△T时，系统最连贯，而且舵机运动的最快。

实际过程中ｗ不尽相同，连贯运动时的极限△T比较难以计算出来。

假如我们的舵机1DIV =8us，当PWM信号以最小变化量即（1DIV=8us）依次变化时，舵机的分辨率最高，但是速度会减慢[1]。

使用示例

#include <reg51.h>

#define uchar unsigned char

#define uint unsigned int

sbit p10=P1^0;

uint aa;

void delay (uint z)

{

unsigned int x,a;

for(x=z;x>0;x--)

for(a=250;a>0;a--);

}

void timer0()interrupt 1 //定时器1为0.1ms一个周期

{

TH0=(65536-100)/256;

TL0=(65536-100)%256;

if(aa<15) //产生周期为20ms，高电平为1.5ms，舵机会转到90度

 p10=1;//如果想反转方向，就需把第一个p10=0，第二个p10=-1

else

 p10=0;

 aa++;

if(aa>200)aa=0;

}

void init()

{

  aa=0;

a=0;

p10=0;

TMOD=0x11;

TH0=(65536-100)/256;

TL0=(65536-100)%256;

EA=1;

ET0=1;

}

void main()

{

init();

TR0=1;

p10=1;

while(1)

{

}

}

三．代码

Esp8266控制L298N代码

①uart.c

#include "uart.h"

#define Buf\_Max 50

uint8 xdata Rec\_Buf[Buf\_Max]; //接收串口2缓存数组

uint8 i = 0;

void SendData(uint8 ch);

void SendString(uint8 \*s);

void UartInit(void);

void U2SendData(uint8 ch);

void U2SendString(uint8 \*s);

void CLR\_Buf(void);

bit Hand(uint8 \*a);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

功能描述：串口初始化(串口1和串口2)

入口参数：无

返回值：无

备注：晶振选择18.432MHz 串口1波特率9600bps 串口2波特率115200bps

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UartInit(void)

{

PCON &= 0x3f; //波特率不倍速，串行口工作方式由SM0、SM1决定

SCON = 0x50; //8位数据，可变波特率

AUXR |= 0x40; //定时器1时钟1为FOSC,即1T

AUXR &= 0xFE; //串口1选择定时器1为波特率发生器

TMOD &= 0x0F; //设定定时器1为16位自动重装方式

TL1 = 0x20; //设定定时初值

TH1 = 0xFE; //设定定时初值

ET1 = 0; //禁止定时器1中断

TR1 = 1; //启动定时器1

S2CON = 0x50; //8位数据，可变波特率

AUXR |= 0x04; //定时器2时钟1为FOSC,即1T

T2L = 0xD8; //设定初始值

T2H = 0xFF; //设定初始值

AUXR |= 0x10; //启动定时器2

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

功能描述：串口1发送数据

入口参数：uint8 ch

返回值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void U1SendData(uint8 ch)

{

SBUF = ch; //写数据到UART数据寄存器

while(TI == 0); //在停止位没有发送时，TI为0即一直等待

TI = 0; //清除TI位（该位必须软件清零）

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

功能描述：串口1发送字符串

入口参数：uint8 \*s

返回值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void U1SendString(uint8 \*s)

{

while (\*s) //检测字符串结束标志

{

U1SendData(\*s++); //发送当前字符

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

功能描述：串口2发送数据

入口参数：uint8 ch

返回值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void U2SendData(uint8 ch)

{

S2BUF = ch; //写数据到UART数据寄存器

while(!(S2CON&S2TI)); //在停止位没有发送时，S2TI为0即一直等待

S2CON&=~S2TI; //清除S2CON寄存器对应S2TI位（该位必须软件清零）

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

功能描述：串口2发送字符串

入口参数：uint8 \*s

返回值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void U2SendString(uint8 \*s)

{

IE2 &= 0xFE; // 串口2中断关闭

while (\*s) //检测字符串结束标志

{

U2SendData(\*s++); //发送当前字符

}

IE2 |= 0x01; // 串口2中断打开

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

功能描述：握手成功与否函数

入口参数：uint8 \*a

返回值：位

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bit Hand(uint8 \*a)

{

if(strstr(Rec\_Buf,a)!=NULL) //判断字符串a是否是字符串Rec\_Buf的子串

return 1; //如果字符串a是字符串Rec\_Buf的子串

else

return 0; //如果字符串a不是字符串Rec\_Buf的子串

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

功能描述：清除缓存内容函数

入口参数：无

返回值：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CLR\_Buf(void)

{

uint8 k;

for(k=0;k<Buf\_Max;k++) //将串口2缓存数组的值都清为零

{

Rec\_Buf[k] = 0;

}

i = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 描 述 : 串口1中断服务函数

\* 入 参 : 无

\* 返回值 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Uart1() interrupt UART1\_VECTOR using 1

{

ES = 0; // 串口1中断关闭

if (RI) //串行接收到停止位的中间时刻时，该位置1

{

RI = 0; //清除RI位 （该位必须软件清零）

}

if (TI) //在停止位开始发送时，该位置1

{

TI = 0; //清除TI位（该位必须软件清零）

}

ES = 1; // 串口1中断打开

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 描 述 : 串口2中断服务函数

\* 入 参 : 无

\* 返回值 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Uart2() interrupt UART2\_VECTOR using 1

{

IE2 &= 0xFE; // 串口2中断关闭

if (S2CON & S2RI) //串行接收到停止位的中间时刻时，该位置1

{

S2CON &= ~S2RI; //清除S2CON寄存器对应S2RI位（该位必须软件清零）

Rec\_Buf[i] = S2BUF; //把串口2缓存SBUF寄存器数据依次存放到数组Rec\_Buf2中

i++;

if(i>Buf\_Max) //接收数大于定义接收数组最大个数时，覆盖接收数组之前值

{

i = 0;

}

}

if (S2CON & S2TI) //在停止位开始发送时，该位置1

{

S2CON &= ~S2TI; //清除S2CON寄存器对应S2TI位（该位必须软件清零）

}

IE2 |= 0x01; // 串口2中断打开

}

②main.c

#include "delay.h"

#include "uart.h"

sbit RST = P2^0;

sbit IN1 = P1^0;

sbit IN2 = P1^1;

sbit IN3 = P1^2;

sbit IN4 = P1^3;

char code str1[]="AT\r\n";

char code str2[]="AT+CWMODE=3\r\n";

char code str3[]="AT+CWJAP=\"FiYu\",\"55815581\"\r\n";

char code str4[]="AT+CIFSR\r\n";

char code str5[]="AT+CIPSTART=\"TCP\",\"192.168.191.1\",5000\r\n";

char code str6[]="AT+CIPSEND=6\r\n";

char code str7[]="hello!\r\n";

char code str8[]="AT+CIPSERVER=1,5000\r\n";

char code str9[]="AT+CIPMUX=1\r\n";

char code str10[]="AT+RST\r\n";

char code str11[]="AT+CIPSEND=0,15\r\n";

char code str12[]="Command Executed!\r\n";

int main()

{

P0M1 &= 0x3F; P0M0 &= 0x3F;

P1M1 &= 0xFC; P1M0 &= 0xFC;

P2M1 &= 0xFE; P2M0 |= 0x01;

P3M1 &= 0xFC; P3M0 &= 0xFC;

RST = 1;

UartInit();

ES = 1;

IE2 |= 0x01;

EA = 1;

DelayMS(1000);

DelayUS(100);

U1SendString("Please wait while we are getting the device ready\r\n");

CLR\_Buf();

while(!Hand("OK"))

{

U2SendString(str1);

DelayMS(500);

}

CLR\_Buf();

U1SendString("OK,Succeed Establish connection with ESP8266\r\n");

while(!(Hand("OK")|Hand("no change")))

{

U2SendString(str2);

DelayMS(500);

}

if(Hand("OK"))

{

CLR\_Buf();

U2SendString(str10);

DelayMS(500);

}

CLR\_Buf();

U1SendString("OK,ESP8266 has been set as AP+Station Mode\r\n");

while(!Hand("OK"))

{

U2SendString(str9);

DelayMS(500);

}

CLR\_Buf();

while(!Hand("OK"))

{

U2SendString(str8);

DelayMS(500);

}

CLR\_Buf();

while(!Hand("OK"))

{

U2SendString(str4);

DelayMS(500);

}

U1SendString("Congratulations, Everything is set up! TCP sever:192.168.4.1, Port: 5000\r\n");

CLR\_Buf();

while (1)

{

if(Hand("FORWARD"))

{

ES = 0;

IE2 &= 0xFE;

//L298N roll in direction 1

IN1 = 1;

IN2 = 0;

IN3 = 1;

IN4 = 0;

CLR\_Buf(); U1SendString("Command: Turn on LED1, Executed!\r\n");

ES = 1;

IE2 |= 0x01;

}

else if(Hand("BACK"))

{

ES = 0;

IE2 &= 0xFE;

//L298N roll in direction 2

IN1 = 0;

IN2 = 1;

IN3 = 0;

IN4 = 1;

CLR\_Buf();

U1SendString("Command: Turn off LED1, Executed!\r\n");

ES = 1;

IE2 |= 0x01;

}

else if(Hand("STOP"))

{

ES = 0;

IE2 &= 0xFE; //stop L298N

IN1 = 0;

IN2 = 0;

IN3 = 0;

IN4 = 0;

CLR\_Buf(); U1SendString("Command: Turn on LED2, Executed!\r\n");

ES = 1;

IE2 |= 0x01;

}

}

}

参考资料：

1. : <https://blog.csdn.net/Wekic/article/details/51761525>
2. UART: <https://www.jianshu.com/p/df2ed057b4a0>