串口使用与测量报告

班级：计科 2104 班　学号：202108010426 姓名：冷长佼

一.实验目的

1.学习 linux 系统的基本使用，常见命令。

2.使用示波器观察 STC 单片机 UART 串口输出信号。

3.学习使用 Linux 下 io 函数 read、write 和 epoll 函数，实现串口数据通信。

4.熟练 RS485 串口的信号特点

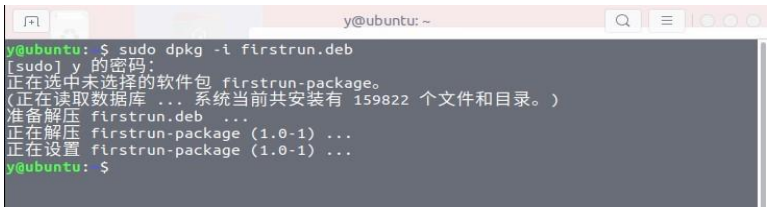
5.熟练处理流式通信数据

6.理解 485 总线的冲突问题

二.实验过程

1.linux 操作系统平台

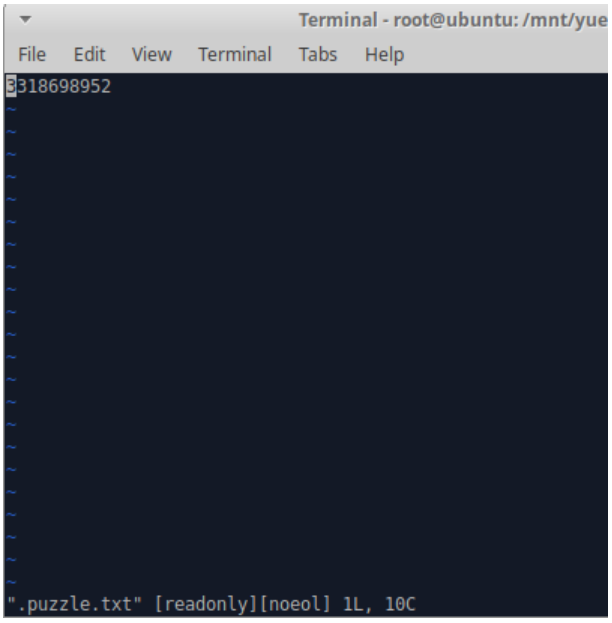
安装 firstrun.deb 包　使用指令 sudo dpkg -i firstrun.deb 进行安装



运行根目录下的/gettips

运行指令切换到 2 中提供的目录，提示权限不够，需要修改权限，修改后仍不能切换，直接切换到 root 用户，进入该目录

ls 查看并使用 vim 打开文件



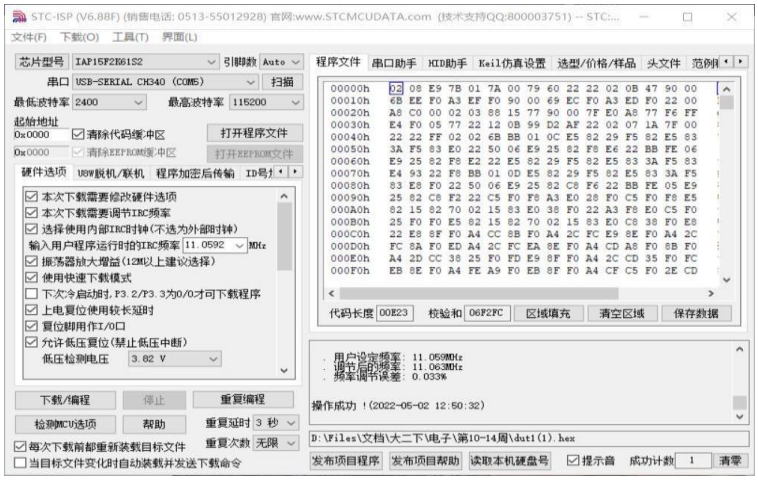
使用如下命令提交

curl"132.232.98.70:6363/check?id=202108010426&v=3318698952" 提交后返回 OK，再次提交返回 DUP

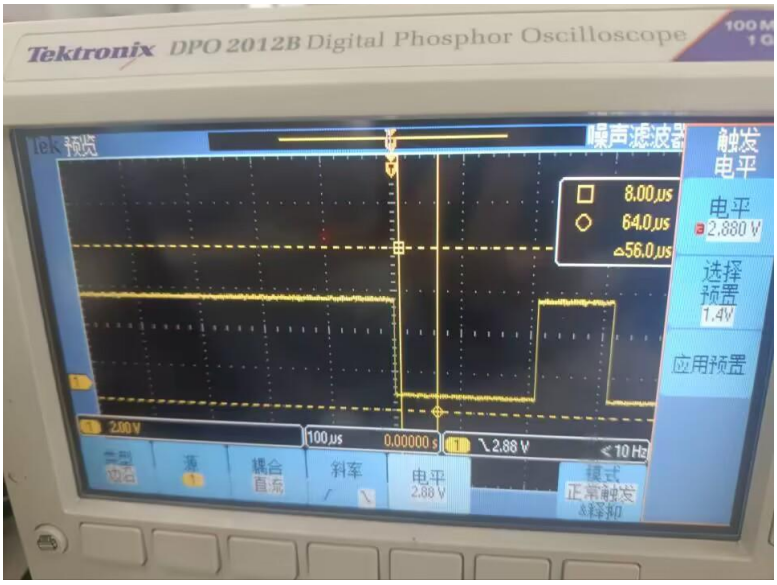
```
root@ubuntu:/mnt/yuelu# curl "132.232.98.70:6363/check?id=202108010426&v=3318698952"
OK
```

2. Linux 平台串口数据接收

向 STC 单片计算机板下载程序



使用示波器观察 STC 单片机 UART 串口输出信号，识别单片机发送数据所使用的波特率：



下降沿与上升沿间隔 200 微秒，计算出波特率为 4800。

修改 C 语言程序从虚拟机的串口读取数据的波特率为 4800，运行程序，读取单片机的序列号为 a01d54de563b4e4be51a7e

```
witeNum0 = 1
   read_len = 13
 aa 55 a0 1d 54 de 56 3b 4e 4b e5 1a 7e
```

将序列号提交。

```
root@ubuntu:/home/ck/ck# curl "132.232.98.70:6363/checkBaud?id=202108010426&v=a0
1d54de563b4e4be51a7e"
OK
```

3. 计算机串口数据收发与测量

向单片机下载程序，波特率为 4800，可以接收到序列号，接收到的序列号为：

a01d54de563b4e4be51a7e

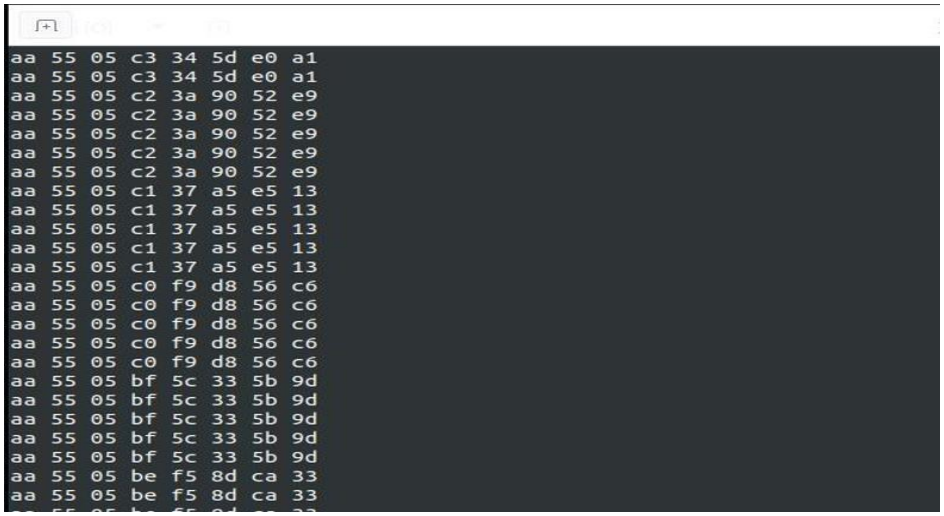
根据实验要求，需要向串口写入学号，读取密码，再写入密码，不断重复至不出现新的密码。编写程序，在读取串口数据实验给出的参考代码中进行修改，使程序能够完成读取发送数据的功能。

向串口写数据，只需要使用 write 函数，传入文件描述符，写入内容的指针，字节数。首先定义一个缓冲区，内容为 AA55+学号，先向串口写入。后续发送密码仍然使用这个缓冲区，由于密码在一串数据中的位置不确定，这里缓冲区多留出了一点空间，共 18 个字节。

```
unsigned char buf[]={0xAA,0x55,0x02,0x00,0x02,0x01,0x00,0x08,0x00,0x01,0x00,0x04,0x02,0x06,0x00,0x00,0x00,0x00};
if(write(fd,buf,14)<=0) printf("write failed!\n");
```

发送学号后，串口将接收到包含密码的数据，数据格式为 AA+55+密码位置 +填充符+密码。为了根据密码位置找到密码，需要对这一

串数据进行计数，在读取数据时，按字节读入数据，并记录前一个字符，前一个字符为 AA，该字符为 55 时，将 count 计数设置为 2，然后下一个字符为密码位置，将密码写入缓冲区中，并向串口写入。读取数据并写入密码的部分代码如下



```
aa 55 05 c3 34 5d e0 a1
aa 55 05 c3 34 5d e0 a1
aa 55 05 c2 3a 90 52 e9
aa 55 05 c2 3a 90 52 e9
aa 55 05 c2 3a 90 52 e9
aa 55 05 c2 3a 90 52 e9
aa 55 05 c2 3a 90 52 e9
aa 55 05 c1 37 a5 e5 13
aa 55 05 c1 37 a5 e5 13
aa 55 05 c1 37 a5 e5 13
aa 55 05 c1 37 a5 e5 13
aa 55 05 c1 37 a5 e5 13
aa 55 05 c0 f9 d8 56 c6
aa 55 05 c0 f9 d8 56 c6
aa 55 05 c0 f9 d8 56 c6
aa 55 05 c0 f9 d8 56 c6
aa 55 05 c0 f9 d8 56 c6
aa 55 05 bf 5c 33 5b 9d
aa 55 05 bf 5c 33 5b 9d
aa 55 05 bf 5c 33 5b 9d
aa 55 05 bf 5c 33 5b 9d
aa 55 05 bf 5c 33 5b 9d
aa 55 05 be f5 8d ca 33
aa 55 05 be f5 8d ca 33
```

：

运行程序，可以得到不断变化的密码。以上是逻辑正确的程序。最初编写的一个程序只识别了 AA 就开始了新的一串计数读取，但因为该单片机没有包含 AA 的密码所以不影响使用，且循环中的处理是读到 AA 后再 while 循环读取，不易理解，这个最初编写的代码也附在工程文件中（main1.c）。

将最终得到的密码提交，得到的是第 180 个密码。

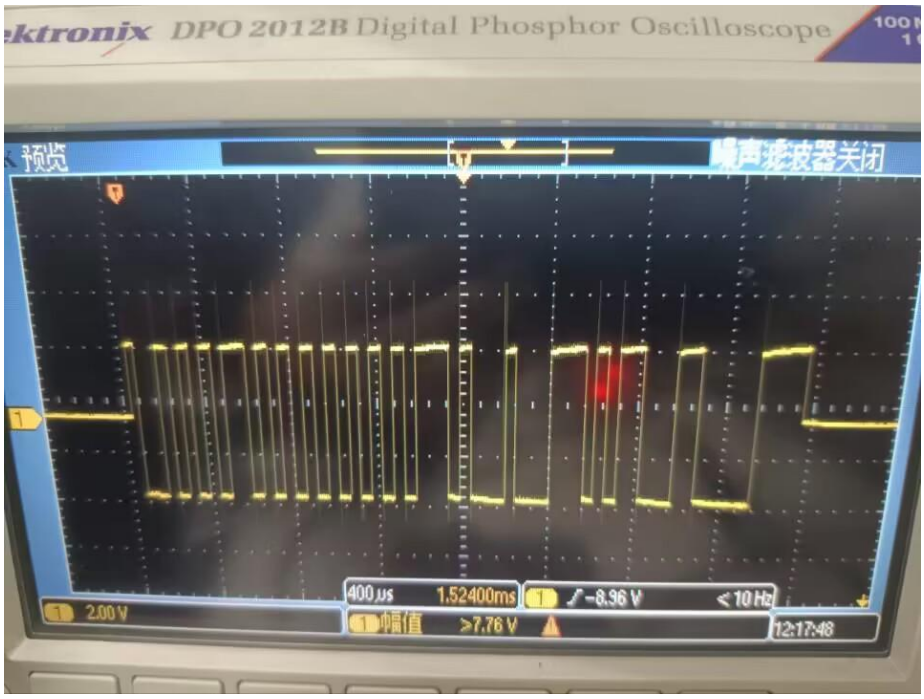
可以看到，每 5 次相同的密码之后才会出现新的密码，这可能与读取数据是单字节读入有关，读到完整的密码需要串口接收 5 串数据才能完成，如果修改为一次读多字节数据，应该可以更快的读到密码并写入数据，得到更多密码。但相应的计数也要处理，因为一次能读取到的字节数可能是不确定的。由于时间有限，没有完成多字节读取的程序编写和尝试。

4.RS485 信号测量

本实验需要将 A 板与 B 板通过 RS485 接口连接并进行通信。需要使用杜邦线连接两单片机板的 RS485 接口，通过计算机向 B 板写入发送的数据（A 板序列号+学号），数据将通过串口发给 B 板，再经 RS485 接口发给 A 板，发送后 A 板将发回密码，B 板将密码通过串口发送到计算机，需要在计算机上读出密码。

1.测量 A 板波特率

将 A 板 RS485 接口接到示波器，测量如下：



波特率大约为 1/50us，故可确认波特率为 19200。将 B 板接示波器，控制摇杆调整至波特率为 19200。

2.测量序列号

算出序列号为 75c48df0

3.读密码

在 linux 平台下的前次实验的读取串口程序中，设置缓冲区，循环中向串口写入 AA55+序列号+学号

```
unsigned char buf[]={0xAA,0x55,0x75,0xc4,0x8d,0xf0,0x02,0x00,0x02,0x01,0x00,0x08,0x00,0x01,0x00,0x04,0x02,0x06};
int wl=write(fd,buf,15);
printf("wl=%d",wl);
```

运行程序，可以收到 A 板发送的密码：

```
jcjc@ubuntu:~/ck$ sudo ./main
set epoll ok!
wl=18
witeNum0 = 1
   read_len = 6
 aa 55 34 5f 4e a7
```

密码为 345f4ea7。

4.提交

将序列号及密码提交。

```
jcjc@ubuntu:~/ck$ ^C
jcjc@ubuntu:~/ck$ curl "132.232.98.70:6363/check485?id=202108010426&v=75C48DF0&s=345f4ea7"
OKjcjc@ubuntu:~/ck$
```

RS485 总线数据收发

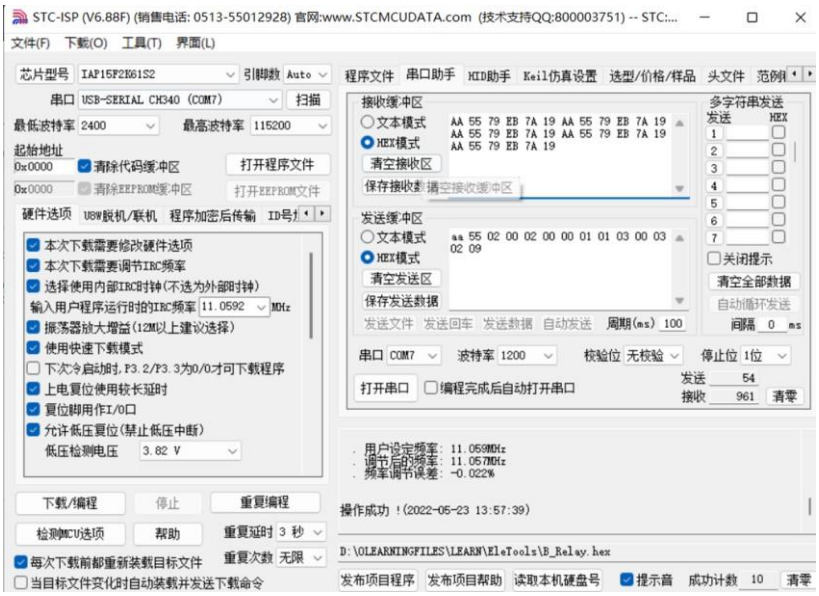
RS485 总线有两条线信号线，能够传输一个逻辑信号。计算机标准的 UART 串口有 RX、TX 收发两条线，因此能够同时进行数据的接收和发送。而 RS485 只有一个逻辑信号，因此同一时刻只能有一个主体进行数据发送（因此叫做半双工通信串口）。

本实验需要连接两单片机板的 RS485 接口，A 板将先发送序列号。使用计算机连接 B 板串口，发送学号，然后 B 板将发送给 A 板，此后 A 板将通过 RS485 接口发送密码，B 板需要接收密码，取出密码发回，然后 A 板将发送新的密码。

重复该过程至 A 板不再发送新的密码。

1.测量 A 板序列号

先使用串口调试助手接收 A 板的序列号，序列号为 79eb7a19



2.接收密码，分析密码格式

先使用串口调试助手发送学号，接收一次 A 板的密码，从而判断 A 板发送密码的格式。得到的密码格式为 aa 55 ff 5b 11 30 95，即 aa 55 后开始四个字节即为密码，中间只有一个字节的填充字符。

3.编写代码

已知密码格式，接下来编写程序完成 B 板的学号发送，密码的处理及密码的发送工作。

编写代码使用 c 语言在第一次实验中的代码基础上进行修改，一个字节一个字节读取数据，然后取出密码发送数据。

C 语言实现

主要部分如下：

```
EpollInit(fd); //初始化终端事件触发函数epoll,设置要监听的事件及相关参数等
unsigned char buf[]={0xAA,0x55,0x02, 0x00, 0x02, 0x00, 0x00, 0x04, 0x00, 0x06, 0x01, 0x04, 0x00, 0x09};
if(write(fd,buf,14)<=0) printf("write failed!\n");
int count=0;
int start=4;                                        //开始的字节
unsigned char temp;
while(1){
        rl = ComRead(tmp,1);                        //读取数据
        printf(" %02x", tmp[0]);
        //识别前导AA 55
        if(count>=2){
                count++;
                if(count>=start) buf[count-start+2]=tmp[0];
                if(count-start==3) {                //读到4个字节
                        write(fd,buf,6);
                        count=0;                    //清0
                }
        }
        if(tmp[0]==0x55&&temp==0xAA)
                count=2;                            //找到aa55，count开始计数
        temp=tmp[0];                                //上一个字节
}
close(epid);
close(fd);
return 0;
```

读取到 aa55 后开始计数，start 设置为 4，即 count=4，第四个字节而开始向缓冲区 buf 写入密码字节，写够 4 字节后向串口写入数据。由于这种实现需要一个字节一个字节读取数据。

三.实验总结

1.涉及知识点

- Linux 操作系统的用户与组
- 文件的权限与权限修改命令
- 目录，文件操作相关命令
- 串口波特率以及使用示波器测量波特率
- 单片机程序的下载以及串口调试方法
- Linux 的 IO 操作函数
- 串口数据收发

2.总结

通过实验进一步熟悉了 linux 操作系统的使用。熟悉了单片机下载程序，调试串口的方法，理解了使用 IO 函数实现的串口数据的收发。编写串口数据收发程序也进一步体会了如何对串口数据进行处理。在编写程序时先编写了一个复杂且存在错误的版本，后重新编写了清晰，正确的版本，加深了理解。在读取串口数据实验观察到读取 13 个字节，但每次总是只读到 1 个字节，因此编写串口收发数据时直接采用了单个字节读取的方式，导致程序性能差，得到的密码少。这里还可以进行重新改写优化。试用 c++或 python 或许也有更好的结果，后续可以继续了解不同语言读取串口的方法。

三.实验总结

通过实验进一步熟悉了串口通信的方法，以及使用 write，read 等函数进行串口数据进行读写。了解了 RS485 接口的使用，通过 RS485 实现了两板的通信和数据传输。认识了 RS485 的半双工通信模式，在程序运行时，有些产生的密码是错误的，即发生了冲突得到了错误数据。对于程序的编写，能够通过 c 语言程序实现功能。