串口使用与测量第一次报告

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。

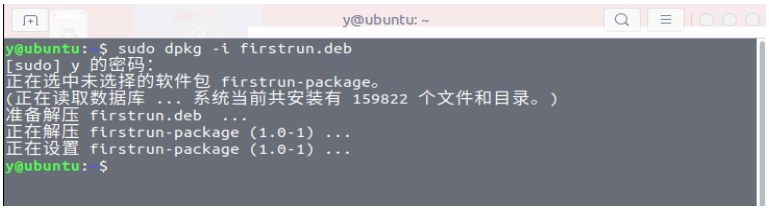
3.学习使用 Linux 下 io 函数 read、write 和 epoll 函数，实现串口数据通信。

二.实验过程

1.linux 操作系统平台

安装 firstrun.deb 包

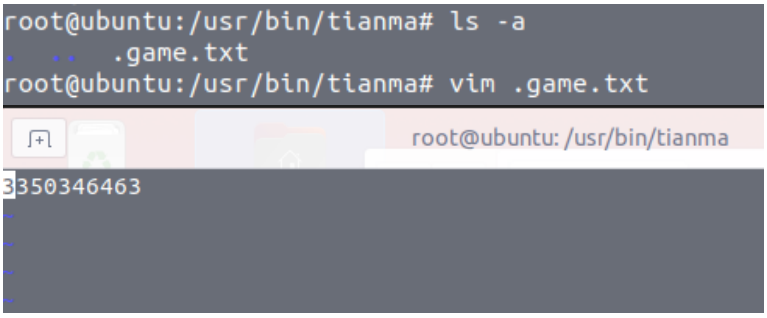
使用指令 sudo dpkg -i firstrun.deb 进行安装



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=202004061409&v=3350346463"

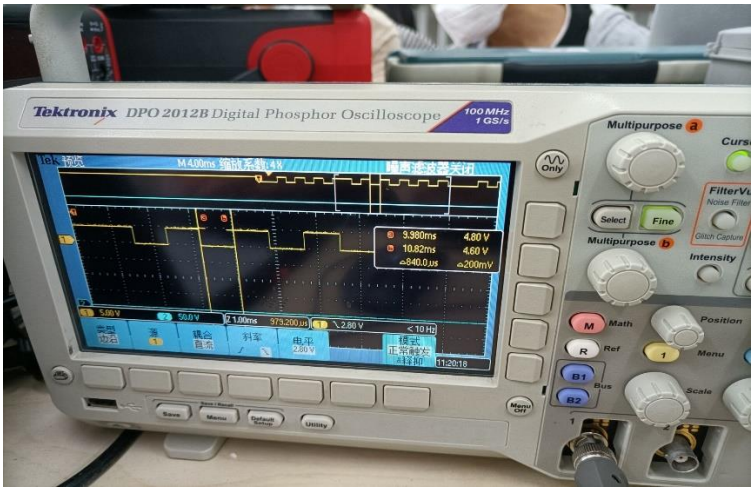
提交后返回 OK，再次提交返回 DUP

```
root@ubuntu:/usr/bin/tianma# curl "132.232.98.70:6363/check?id=202004061409&v=3350346463"
DUP
root@ubuntu:/usr/bin/tianma#
```

2. Linux 平台串口数据接收

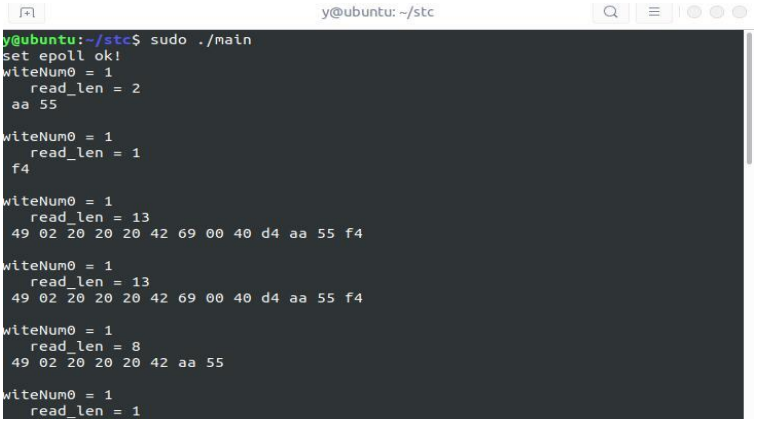
向 STC 单片计算机板下载程序

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：

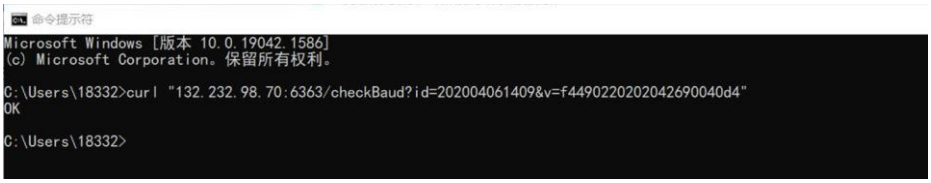


下降沿与上升沿间隔 0.84ms，计算出波特率为 1200。

修改 C 语言程序从虚拟机的串口读取数据的波特率为 1200，运行程序，读取单片机的序列号为 f4490220202042690040d4



将序列号提交。



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 1200，可以接收到序列号，接收到的序列号为：ad1d37ad1a5a796641ca90

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,0x55,0x02,0x00,0x02,0x00,0x00,0x04,0x00,0x06,0x01,0x04,0x00,0x09,0x00,0x00,0x00,0x00};
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：

```
while(1){
        //bzero(tmp,sizeof(tmp));                          //置零
        rl = ComRead(tmp,1);                               //读取数据
        printf(" %02x", tmp[0]);
        //识别前导AA 55
        if(count>=2){
                count++;
                if(count==3) start=(int)tmp[0];            //起始位置
                if(count>=start) buf[count-start+2]=tmp[0];
                if(count-start==3) {
                        write(fd,buf,6);
                        //printf("\n write %2x %2x %2x %2x %2x %2x\n",buf[0],buf[1],buf[2],buf[3],buf[4],buf[5]);
                        count=0;                           //清0
                }
        }
        if(tmp[0]==0x55&&temp==0xAA) {
                count=2;
                printf("\n");
        }
        temp=tmp[0];                                       //上一个字节
}
```

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）。

```
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
```

将最终得到的密码提交，得到的是第 72 个密码。

```
命令提示符
Microsoft Windows [版本 10.0.19042.1586]
(c) Microsoft Corporation。保留所有权利。

C:\Users\18332>curl "132.232.98.70:6363/checkSecret?id=202004061409&v=A01DD7A01A5A796641CA90&s=CDBCC92D"
72
C:\Users\18332>
```

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。

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。