# 1.舵机库冲突

不讲了……

# 2.软串口库，与定时中断和I2C通信冲突

在uno中我们需要将HC08与esp进行串口透传，首先是根据样例使用了软串口库#include

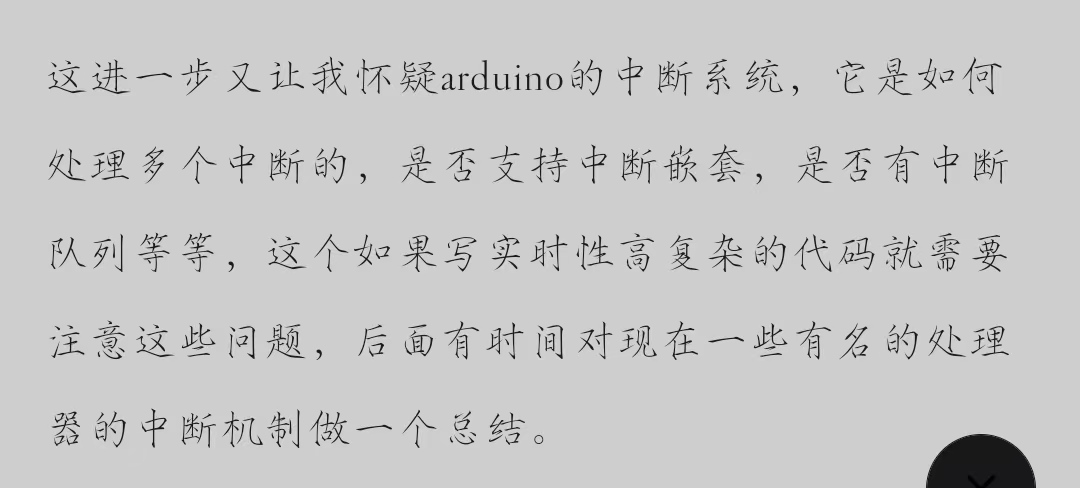
在开启定时器下，include<SoftWareSerial>并执行SW\_serial.begin(115200)后，OLED熄屏，蓝牙通信罢工，键盘扫描异常

@原因：软串口除需要IO口模拟RX，TX针脚下，还需要使用定时器资源模拟波特率。但不是因为定时器选用冲突，而是中断冲突，uno主控芯片为**ATmega328P**，其中断系统异常混乱。

值得一提的是，在我们的项目中，Uno有如下任务：与oled的I2C通信，与HC08的UART通信，定时执行键盘扫描算法。有三个中断源……而uno中断系统很羸弱。

对于儿童玩具Arduino Uno,涉及定时，中断，串口中断后，

网上深入的资料太少，我只找到了和我具有相同疑惑的blog:

* @摘自CSDN：

**Refercence:**[Uno主控芯片手册datasheet](C:\Users\yceachan\Desktop\ICthings\Projects\uno主控芯片手册.ptf.pdf)

缺点：只能挂载一个无线通信设备，本计划再加个HC05模块配对手机，作为产品的管理端和蓝牙开锁端

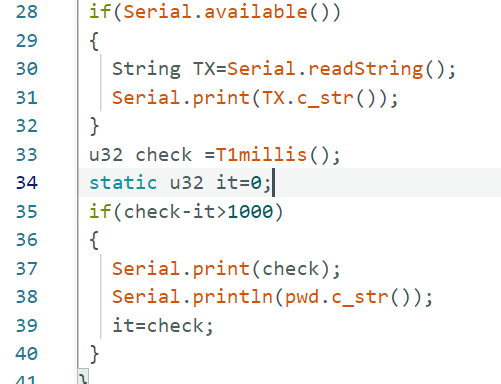
想法：利用esp32的wifi模块，手机在局域网发送指令，由esp32读取，通过BLE下的UART透传发送给UNO

# 3.硬串口的中断冲突，与String类的冲突

//引入Serial.available()的串口RX事件后，会有坑……  
 if(Serial.available())  
 {  
 static String RX=Serial.readString();  
 if(strstr(RX.c\_str(),"CT+PSWD="))  
 {  
 PWD=RX.substring(8,12);  
 }  
 if(strstr(RX.c\_str(),"CT+DATE=")){/\*\*/}  
 }

实验如下，配置了频率为33hz(33ms)的定时器Timer1，绑定中断服务keyscan,封装时间函数T1millis()；在loop回圈中，对RX事件中，将读入字符串原样打印，在TX事件中，定时每秒打印监听开机时间与密码

看个乐呵吧：





结果如图，串口UART通信与定时器计时中断能同时工作，但当按下按键之后（第四行)，串口罢工，板载TX信号灯始终熄灭，向串口发送数据无法触发RX事件，板载RX信号灯无应答

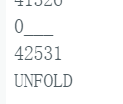
在RX事件中执行Timeroff，向串口发送数据，板载RX信号灯闪烁，串口TX仍罢工

更换时间函数为基于Timer0封装的millis()与将TX事件放入中断服务中，实验现象包括不限于：TXthings中，当pwd数据类型为String时，pwd部分打印空白，打印乱码，甚至打印全局区常量"password",时间部分计时清零，出现乱码…………

1. 将键盘扫描算法的push入队改为cnt++,串口监视cnt，cnt计数并发生Bounce现象，结果符合预期。我写的队列容器显然是没有问题的，在不使用串口时容器数据可正常同步给pwd，猜想是因为加入push入队使中断服务函数过长………
2. 在每个loop回圈打印队列内坐标，结果符合预期，数据正常入队出队，锁定问题出在pwd字符串中。



3.在keyscan中屏蔽入队时间记录，逐字符打印密码，结果符合预期。锁定问题为pwd数据类型-String。 Arduino封装的String容器（注意不是std::string)似乎与串口有些不兼容（至少在额外开启了定时中断后会冲突）。在将pwd类型写为String时，按下按键后串口开始罢工。将pwd的类型写为char[]时，可正常运行



**4.至此硬串口冲突问题解决完成，解决方案如下：**

* 在Timer1的定时中断中，使用sei()打开全局中断，允许任意中断打断此中断（将优先级设为最低）
* 在所有串口相关的事件中使用Timer1off()和Timer1on()（这两个函数是对time1中断使能寄存器的封装）来包裹相关代码，以在串口事件中屏蔽定时中断（设置串口中断优先级高于定时器1中断）
* 不要用Arduino封装的String类（注意它不是std::string)，使用定长字符数组char[]管理密码，

@：用别人造好的轮子面向接口编程有手就行，也不提倡重复造轮子，但如果别人造的轮子有坑呢？

1. **Uno的uart通信程序如下：**

void RXthing()  
{  
 Timeroff();//函数原型：TIMSK1 &= ~(1 << OCIE1A);//将TIMER1使能寄存器TIMSK1下时钟中断使能寄存器OCIE1位0 （即，关闭定时中断）  
 if(Serial.available())  
 {  
 String TX=Serial.readString();  
 Serial.print(TX.c\_str());  
 }  
 Timer1on();//函数原型：TIMSK1 |= (1 << OCIE1A);//置1  
}  
void TXthing()  
{  
 static unsigned long it=0;  
 unsigned time=millis();  
 if(time-it>1000)  
 {  
 Timeroff();  
 Serial.print(pwd);  
 Serial.println(time);  
 it=time;  
 Timer1on();  
 }  
}

6.附,Timer1定时器配置

void Timer1\_setup(int hz)  
{  
 cli();//关闭全局中断  
   
 TCCR1A = 0;//TCCR1为控制寄存器,共AB 16位，A资源：COM1A1,COM1A0,COM1B!,COM1B0,-,-,WGM11,WGM10  
 TCCR1B = 0;// B资源：ICNC1,ICES1,-,WGM13,-,WGM12,CS12,CS11,CS10  
 TIMSK1 = 0;  
  
 TCNT1 = 0; //16位计数器寄存器  
 OCR1A = 16000000/hz/256 -1;// 16位比较寄存器,共AB两个 ocr=晶振/(分频\*目标频率) -1;   
   
 TCCR1B |= (1 << WGM12);//Timer1下控制寄存器WGM1\_位真值：`0100` ，即基于OCR1A的CTC模式  
 TCCR1B |= (1 << CS12);//Timer1下控制寄存器 CS1\_位真值：`100` ，即256倍分频  
  
 // TIMSK1 |= (1 << OCIE1A);//TIMSK使能寄存器。 8位，资源：-,-,ICIE1,-,-,OCIE1B,OCIE1A,TOIE1，此语句使能OCR1A中断  
   
 sei();//恢复全局中断  
}  
ISR(TIMER1\_COMPA\_vect)  
{   
 sei();//使用sei()方法恢复全局中断，相当于把次中断优先级拉为最低任意中断均可打断此中断  
 // clk+=33;//——此后millis能正常计时，无需额外计时  
 //Serial.println("ISRing");  
 if(Timer1\_callback) (\*Timer1\_callback)();   
 // else Serial.println("error");  
}

附：Arduino的串口中断是一种软中断，串口的RX，TX事件发生后，仅将flag位寄存器挂起，把数据留存缓冲区，在loop执行完毕后开始执行中断服务程序





Serial串口（的RX）中断属于伪中断，在代码文件中写下的与串口事件有关的函数经编译后，整合在SerialEvent里。当且仅当同时满足loop执行完毕，硬串口对象Serial被实例化，其RX缓冲区有数据传入，才会执行SerialEvent

**Refercence**:[Arduino的伪中断](https://zhuanlan.zhihu.com/p/44104620)

所以为什么会影响定时中断呢：dont know（dontknow)

在自行配置的定时器1下，定时器0的millis，delay异常秒读（因为中断服务会自动禁用所有其他优先级比它的中断。

**Refercence:**:[Arduino踩坑手册-《中断故障排查指南》-中断、串口、定时器等片内硬件资源之间的冲突](https://blog.csdn.net/u014554395/article/details/89152573)

**Refercence:**[Uno主控芯片手册datasheet](C:\Users\yceachan\Desktop\ICthings\Projects\uno主控芯片手册.ptf.pdf)

* 很喜欢博主的一句话：**（在过度封装而不阅读文档下，）Arduino就是个儿童玩具**

# 4.观察一下uno的中断系统

uno硬串口的波特率直接由系统时钟分频后经Baud Rate Generator@：波特率发生器 一个独立的，不对外开放接口的计数器