# Ardunio开发板间的串口通信

**一、什么是串口通信**

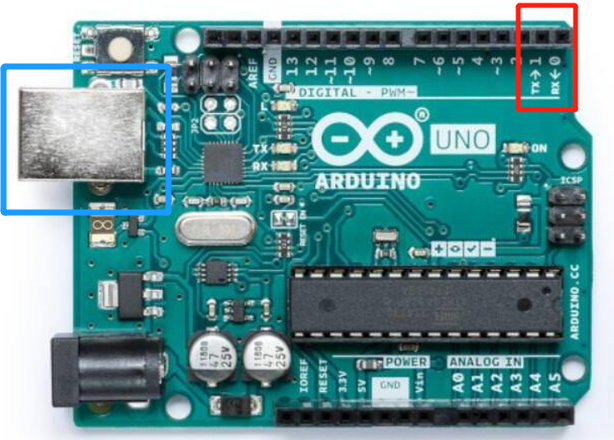
串口通信（Serial Communications）是指串口按位（bit）发送和接收字节。串口用于ASCII码字符的传输，通信使用3根线完成，分别是地线、发送线、接收线。由于串口通信是异步的，端口能够在一根线上发送数据同时在另一根线上接收数据，其他线用于握手，但不是必须的。串口通信最重要的参数是波特率、数据位、停止位和奇偶校验。 对于两个进行通信的端口，这些参数必须要匹配。

通信双方需要使用一致的波特率才能正常通信。 Arduino串口通信通常会使用以下波特率：300、600、1200、2400、4800、9600、14400、19200、28800、38400、57600、115200等，最常用的是9600。波特率越大，说明串口通信的速率越快，但同时传输效果也越不稳定。这边需要注意的是，当通信双方设置的波特率不同的时候，在串口监视器中接收方仍旧可以接收到发送方发出的信息，但是信息可能会呈现出乱码而失去其本身的价值。具体的波特率设置函数将在后续部分给出。

**二、开发板之间的连接方式**

**1.硬串口通信**

在Ardunio的开发板上一般会有标有TX、RX的引脚，可以用杜邦线将两块开发板上RX、TX引脚交叉相连（RX接TX，TX接RX）来完成开发板之间通信的硬件准备。其中RX为接收端，TX为输出端，从两个引脚的功能上看应该也不难理解为什么为实现两块开发板通信的时候这两对引脚要交叉连接了。下图中红色方框标志出来的部分就是上述的TX、RX引脚。

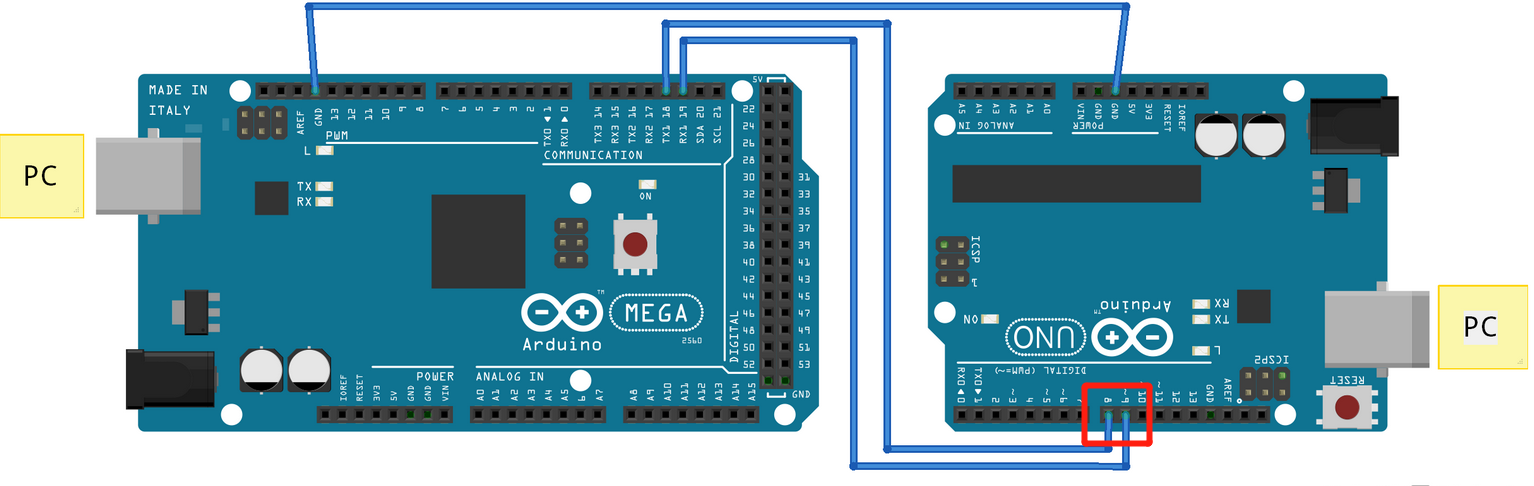


值得注意的是上图蓝色方框标志出的可以连接电脑与电脑通信的USB接口，其通信方式也是串口通信。也就是说一块Ardunio可以通过不同的接口实现多对串口通信，但是通常一个串口只能连接一个设备进行通信，所以在利用先关函数读写的时候最好标注信息来自哪个串口，防止给开发者造成误解或者干扰。

**2.软串口通信**

上文提到，一个串口只能链接一个设备进行通信，但事实上当我们在进行小车的设计的时候我们会发现我们需要通信的设备往往不仅仅只有两块Ardunio开发板。这时候如果Ardunio开发板上自带的TX、RX引脚不够，那我们该怎么办呢？软串口通信给我们提供了解决方案。

通过程序将一些本来不具有串口通信功能的引脚模拟为RX、TX，换而言之，软串口是由程序模拟生成的，虽然使用起来不如硬串口稳定，但与硬串口具有类似的功能与特性。软串口通过AVR芯片的PCINT中断功能来实现，在Arduino UNO 上，所有引脚都支持PCINT中断，因此所有引脚都可设置为软串口的RX接收端。但在其他型号的Arduino上，并不是每个引脚都支持中断功能，所以只有特定的引脚可以设置为RX端。



经过程序设定，将上图红框中的8引脚设置为TX引脚，9引脚设置为RX引脚。具体的软串口相关函数将在后文做出具体介绍。

**三、串口通信的相关函数**

**1.硬串口通信**

**（1）begin()**

描述：开启串口，通常置于setup()函数中。  
语法：Serial.begin(speed)  
参数：speed，波特率，一般取值9600,115200等。  
返回值：无。

**（2）end()**

描述：禁止串口传输。此时串口Rx和Tx可以作为数字IO引脚使用。  
语法：Serial.end()  
参数：无。  
返回值：无。

**（3）print()**

描述：串口输出数据，写入字符数据到串口。  
语法：  
 Serial.print(val)  
 Serial.print(val, format)  
参数：  
 val：打印的值，任意数据类型。  
 format：输出的数据格式。BIN(二进制)、OCT(八进制)、DEC(十进制)、HEX(十六进制)。对 于浮点数，此参数指定要使用的小数位数（默认输出2位）。  
示例：  
 Serial.print(78, BIN) 得到 “1001110”  
 Serial.print(78, OCT) 得到 “116”  
 Serial.print(78, DEC) 得到 “78”  
 Serial.print(78, HEX) 得到 “4E”  
 Serial.print(1.23456, 0) 得到 “1”  
 Serial.print(1.23456, 2) 得到 “1.23”  
 Serial.print(1.23456, 4) 得到 “1.2346”  
 Serial.print(‘N’) 得到 “N”  
 Serial.print(“Hello world.”) 得到 “Hello world.”  
返回值：返回写入的字节数。

**（4）println()**

描述：串口输出数据并换行。  
语法：  
 Serial.println(val)  
 Serial.println(val, format)  
参数：  
 val：打印的值，任意数据类型。  
 format：输出的数据格式。  
返回值：返回写入的字节数。

**（5）available()**

描述：判断串口缓冲区的状态，返回从串口缓冲区读取的字节数。  
语法：Serial.available()  
参数：无。  
返回值：可读取的字节数。

**（6）read()**

描述：读取串口数据，一次读一个字符，读完后删除已读数据。  
语法：Serial.read()  
参数：无。  
返回值：返回串口缓存中第一个可读字节，当没有可读数据时返回-1，整数类型。

**（7）readBytes()**

描述：从串口读取指定长度的字符到缓存数组。  
语法：Serial.readBytes(buffer, length)  
参数：  
 buffer：缓存变量。  
 length：设定的读取长度。  
返回值：返回存入缓存的字符数。

**（8）peek()**

描述：返回1字节的数据，但不会从接收缓冲区删除该数据。与read()函数不同，read()函数读取数据后，会从接收缓冲区删除该数据。

参数：无

返回值：进入接收缓冲区的第1字节的数据；如果没有可读数据，则返回一1。

**（9）write ()**

描述：输出数据到串口。以字节形式输出到串口。

语法：

Serial. write(val)

Serial. write(str)

Serial. write(buf, len)

参数：

val，发送的数据。  
str，String型的数据。.  
buf，数组型的数据。  
len，缓冲区的长度。

返回值：输出的字节数。

**2.软串口通信**

**（1）SoftwareSerial()**

描述：这是SoftwareSerial类的构造函数，通过它可以指定软串口的RX和TX引脚。

语法：

SoftwareSerial mySerial = SoftwareSerial ( rxPin , txPin )

SoftwareSerial mySerial ( rxPin , txPin )

参数：

mySerial，用户自定义软串口对象。  
rxPin，软串口接收引脚。  
txPin，软串口发送引脚。

**（2）listen()**

描述：启软串口监听状态。Arduino在同一时间仅能监听一个软串口，当需要监听某一软串口时，需要该对象调用此函数开启监听功能。

语法：mySerial. listen()

参数：mySerial，用户自定义的软串口对象。

返回值：无

**（3）isListen()**

功能：监测软串口是否正处于监听状态。

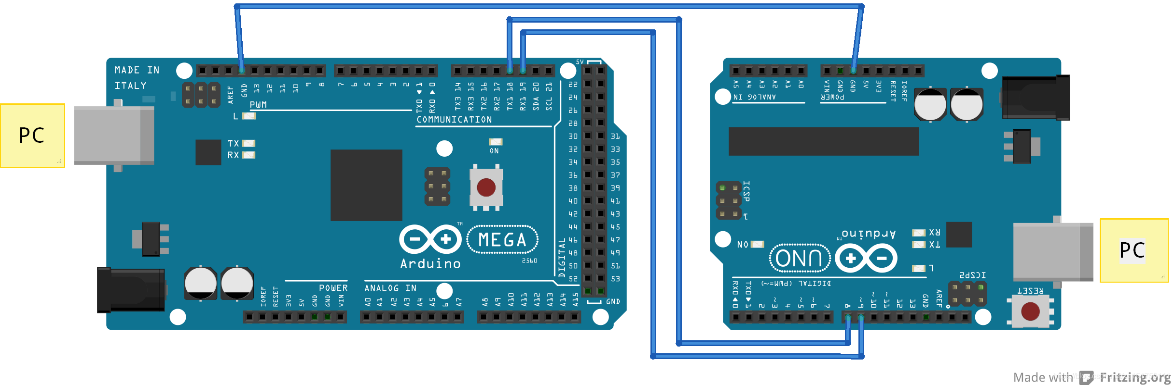
语法：mySerial. isListening()

参数：mySerial，用户自定义的软串口对象。

返回值：boolean型值，为true表示正在监听，为false表示没有监听。

**四、示例程序**

接线示意图：



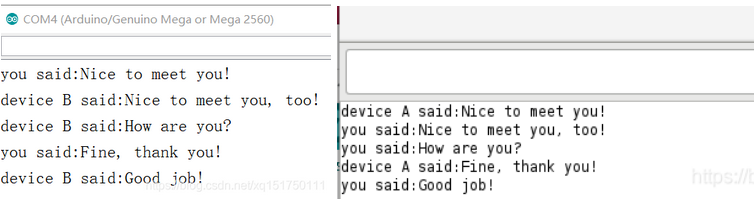
设备A代码：

|  |
| --- |
| void setup(){  Serial.begin(9600); // 初始化serial，该串口用于与计算机连接通信  Serial1.begin(9600); // 初始化serial1，该串口用于与设备B连接通信  }  // 两个字符串分别用于存储A、B两端传来的数据  String device\_A\_String = "";  String device\_B\_String = "";  void loop(){  // 读取从计算机传入的数据，并通过Serial1发送给设备B  if(Serial.available()>0){  if(Serial.peek()!='\n'){  device\_A\_String += char(Serial.read());  }  else{  Serial.read();  Serial.print("you said:");  Serial.println(device\_A\_String);  Serial1.println(device\_A\_String);  device\_A\_String = "";  }  }  // 读取从设备B传入的数据，并在串口监视器中显示  if(Serial1.available()>0){  if(Serial1.peek()!='\n'){  device\_B\_String += char(Serial1.read());  }  else{  Serial1.read();  Serial.print("device B said:");  Serial.println(device\_B\_String);  device\_B\_String = "";  }  }  } |

设备B代码：

|  |
| --- |
| #include <SoftwareSerial.h>  // 新建一个 SoftSerial 对象，RX：9，TX：8  SoftwareSerial softSerial(9,8);  void setup(){  Serial.begin(9600); // 初始化串口通信  softSerial.begin(9600); // 初始化软串口通信  softSerial.listen(); // 监听软串口通信  }  // 两个字符串分别用于存储A、B两端传来的数据  String device\_B\_String = "";  String device\_A\_String = "";  void loop(){  // 读取从计算机传入的数据，并通过Serial1发送给设备A  if(Serial.available()>0){  if(Serial.peek()!='\n'){  device\_B\_String += char(Serial.read());  }  else{  Serial.read();  Serial.print("you said:");  Serial.println(device\_B\_String);  softSerial.println(device\_B\_String);  device\_B\_String = "";  }  }  // 读取从设备A传入的数据，并在串口监视器中显示  if(softSerial.available()>0){  if(softSerial.peek()!='\n'){  device\_A\_String += char(softSerial.read());  }  else{  softSerial.read();  Serial.print("device A said:");  Serial.println(device\_A\_String);  device\_A\_String = "";  }  }  } |

下载程序后，分别打开两个设备的串口监视器，选择各自对应的波特率，在两个串口监视器上随意输人字符，并发送，则会看到如下图所示的效果，这说明串口聊天项目已经成功地运行了。



注：上述代码引用自CSDN博主「长路漫漫2021」的原创文章，遵循CC 4.0 BY-SA版权协议。原文链接：https://blog.csdn.net/xq151750111/article/details/115050441