Arduino 与 Processing 通讯三：Arduion 与 Processing 串行握手

Phodal Huang

October 24, 2017

目录

原文链接：https://www.wandianshenme.com/play/arduino-processing-communication-part-3-shaki

到目前为止，我们已经证明了，Arduino 和 Processing 可以通过串口进行通信，当一个在说话时，另一个在听。我们可以创建一个允许数据流动的链接，以便 Arduino 和 Processing 都可以发送和接收数据？你打赌！在 biz 中，我们称之为串行“握手”，因为双方必须同意何时发送和接收数据。

在本页和下一页中，我们将结合我们前面的两个例子（«从 Processing 中接收 Arduino 的数据»、«从 Arduino 中接收 Processing 的数据»）。这样一来，Processing 可以从 Arduino 接收“Hello，world！”，并将 1 发送回 Arduino 来开头 LED。当然，这也意味着 Arduino 必须能够发送“Hello，world！”，同时从 Processing 处理 1。呼！

步骤 1：握手 - Arduino

让我们先从 Arduino 一方面开始。为了顺利进行，双方都必须知道听什么，对方期待什么。我们也希望通过串口最小化流量，以便我们得到更及时的响应。

就像我们的串行读取例子一样，我们需要一个变量，用于输入数据，而我们需要点亮的 LED 指针的变量：

```
1 char val; // Data received from the serial port
2 int ledPin = 13; // Set the pin to digital I/O 13
3 boolean ledState = LOW; //to toggle our LED
```

因为我们试图提高效率，所以我们要改变我们的代码，所以我们只监听 1，每次我们收到‘1’，我们会打开或关闭 LED。为此，我们为 LED 的 HIGH 或 LOW 状态添加了一个布尔（true 或 false）变量。这意味着我们不必不断地从处理中发送 1 或 0，这样会释放我们的串行端口。

我们的 `setup()` 方法看起来大致相同，并增加了一个 `ContactContact()` 函数，我们将在后面加入。现在，只需要输入它。

```
1 void setup()
2 {
3   pinMode(ledPin, OUTPUT); // Set pin as OUTPUT
4   //initialize serial communications at a 9600 baud rate
5   Serial.begin(9600);
6   establishContact();  // send a byte to establish contact until receiver
        responds
7 }
```

在我们的 loop 函数中，我们刚刚组合、并缩小了我们前面两个程序中的代码。最重要的是，我们已经改变了我们的 LED 代码，基于我们的新布尔值进行切换。‘！’号表示每次我们看到的一个，我们将布尔值设置为与之前相反（因此 LOW 变为 HIGH，反之亦然）。我们也把我们的 “Hello, world!” 放在一个 else 语句中，这样当我们没有看到 ‘1’ 进来的时候，我们只会发送它。

```
1 void loop()
2 {
3   if (Serial.available() > 0) { // If data is available to read,
4     val = Serial.read(); // read it and store it in val
5
6     if(val == '1') //if we get a 1
7     {
8        ledState = !ledState; //flip the ledState
9        digitalWrite(ledPin, ledState);
10    }
11    delay(100);
12  }
13    else {
14    Serial.println("Hello, world!"); //send back a hello world
15    delay(50);
16    }
17 }
```

现在，我们来看看我们在 setup() 方法中使用的 establishContact() 函数。该函数只发送一个字符串（与 Processing 中需要查找的字符串相同），以查看是否听到任何内容 - 表示 Processing 已准备好接收数据。就像一次又一次地说 ‘Marco’，直到你从某个地方听到一个 ‘Polo’。

```
1 void establishContact() {
2   while (Serial.available() <= 0) {
3   Serial.println("A");   // send a capital A
4   delay(300);
5   }
6 }
```

这些就是 Arduino 边的内容，现在切换到 Processing！

步骤 2：握手 - Processing

对于 Processing 方面的代码，我们必须做一些修改。我们将使用 `serialEvent()` 方法，每当我们看到串行缓冲区中的特定字符（作为分隔符）时，它将被调用。基本上它将告诉 Processing，我们完成了一个特定的 “chunk” 的数据 - 在我们的例子中，一个是 “Hello, world!”。

我们程序的开头是一样的，除了一个新的 `firstContact` 布尔变量，它将让我们知道何时连接到 Arduino。

```
1 import processing.serial.*; //import the Serial library
2  Serial myPort;  //the Serial port object
3  String val;
4 // since we're doing serial handshaking,
5 // we need to check if we've heard from the microcontroller
6 boolean firstContact = false;
```

我们的 `setup()` 函数与我们的串行写下的程序相同，除了我们新添了一行 `myPort.bufferUntil('\n')`。它将让我们将传入的数据存入缓冲区，直到我们看到我们正在寻找的特定字符。在当前情况下，它是一个回车（`\n`），因为我们从 Arduino 发送了一个 `Serial.println`。结束处的 “ln” 表示 String 以回车符终止，所以我们知道这将是我们看到的最后一个东西。

```
1 void setup() {
2   size(200, 200); //make our canvas 200 x 200 pixels big
3   //  initialize your serial port and set the baud rate to 9600
4   myPort = new Serial(this, Serial.list()[4], 9600);
5   myPort.bufferUntil('\n');
6 }
```

因为我们将不断地发送数据，我们的 `serialEvent()` 方法现在作为我们的新的 `draw()` 循环，所以我们可以把它留空：

```
1 void draw() {
2   //we can leave the draw method empty,
3   //because all our programming happens in the serialEvent (see below)
4 }
```

现在来看看更大的方法：`serialEvent()`。每次我们看到一个回车，这个方法将被调用。每次我们都要做一些事情来保持事情顺利进行：

- 读取传入的数据
- 看看它里面是否有任何东西（即它不是空的或者 'null'）
- 移除空白和其他不重要的东西
- 如果这是我们第一次监听到正确的东西，将改变我们的 firstContact 布尔值，让 Arduino 知道我们准备好了更多的数据
- 如果它不是我们的第一次运行，则打印数据到控制台，并在我们的窗口中发回任何有效的鼠标点击
- 最后，告诉 Arduino 我们准备好了更多的数据

这里有很多的步骤，但幸运的是，我们 Prcoessing 有相关的函数，使其中的大部分任务很容易实现。让我们来看看这一切如何完成：

```
1  void serialEvent( Serial myPort) {
2  //put the incoming data into a String -
3  //the '\n' is our end delimiter indicating the end of a complete packet
4  val = myPort.readStringUntil('\n');
5  //make sure our data isn't empty before continuing
6  if (val != null) {
7    //trim whitespace and formatting characters (like carriage return)
8    val = trim(val);
9    println(val);
10
11   //look for our 'A' string to start the handshake
12   //if it's there, clear the buffer, and send a request for data
13   if (firstContact == false) {
14     if (val.equals("A")) {
15       myPort.clear();
16       firstContact = true;
17       myPort.write("A");
18       println("contact");
19     }
20   }
21   else { //if we've already established contact, keep getting and parsing
         data
22     println(val);
23
24     if (mousePressed == true)
```

```
25     {                            //if we clicked in the window
26 myPort.write('1');         //send a 1      println("1");
27     }
28
29     // when you've parsed the data you have, ask for more:
30     myPort.write("A");
31     }
32   }
33 }
```

这是有很多代码需要消化，但如果你仔细阅读（特别是注释），它将开始有意义。如果您的 Arduino 代码已完成，并加载到您的主板上，请尝试运行此程序。您应该在控制台上看到 “Hello，world！”，当您在 “Processing” 窗口中单击时，应该看到 Arduino 上的 13 引脚上的 LED 指示灯亮起。成功！你现在是一个串行握手专家。

步骤 3：技巧和窍门

在进行 Arduino 和 Processing 合作开发自己的项目时，以下是可能遇到的问题，以及解决技巧：

- 确保您的波特率匹配
- 确保您正在读取 Processing 中的正确端口 - 有一个 `Serial.list()` 命令，将显示您可以连接到的所有可用端口。
- 如果使用 `serialEvent()` 方法，请确保在 `setup()` 方法中包含 `port.bufferUntil()` 函数。
- 请确保您正在缓冲的任何字符（例如’\ n’）是您实际从 Arduino 发送的字符。
- 如果要发送多个传感器值，最好计算您期望的字节数，以便您知道如何正确解析传感器数据。

相关资源

现在，您已经知道如何将数据从 Arduino 发送到 Processing 并发送回去（即使是同时），您也可以为某些严酷的项目做好准备。通过连接 Arduino 和 Processing，您可以进行实时的可视化传感器数据等操作，或者使用手指中的柔性传感器制作手套，使企鹅出现在屏幕上，或者使用来自 Processing 的命令控制台控制一大堆 LED 指示灯。

以下是一些相关的资源链接：

- Derek Runberg's Processing Curriculum
- Arduino & Processing ATLAS Curriculum
- Processing the Danger Shield
- Arduino, Processing, & MaxMSP

原文链接：https:// learn.sparkfun.com/ tutorials/ connecting-arduino-to-processing

原文链接：https://www.wandianshenme.com/play/arduino-processing-communication-part-3-shaki