从零开始学 LabVIEW

——基于 Arduino 的 LabVIEW 数据采集与仪器控制实验

19 物理学系 张翊凡

目录

从零开始学 LabVIEW

LabVIEW 是美国国家仪器公司（NATIONAL INSTRUMENTS，简称 NI）开发一种图形化编程语言和可视化的虚拟仪器开发环境，被公认为是一种标准化的数据采集和仪器控制软件。其**图形化编程方式**和丰富的工具箱让其在测量、控制、自动化等领域具有广泛的应用。相比于其他控制软件，LabVIEW 最大优点是可以让用户将时间精力高效地投入仪器控制与数据采集（高价值任务）中，而不是绞尽脑汁进行繁琐的程序设计（低价值任务）。

LabVIEW 程序中使用图标、图表及连线等创建了计算机屏幕上的“虚拟仪器”，故程序以.vi（Virtual Instrument）为后缀。虽然仪器的界面是虚拟的，但 LabVIEW 可以通过标准硬件接口连接各式各样的仪器设备，使得一个虚拟仪器可以灵活控制大量实验室设备。

课前预习：熟悉 LabVIEW 操作界面

一、LabVIEW 图形化界面

LabVIEW 图形化界面分为“**前面板**”和“**程序框图（后面板）**”。前面板是用户交互界面，类似于仪器表盘，用于显示和控制，如图 1 所示。前面板上放置的按钮、输入框、下拉框等组件统称为“**控件**”。后面板是编程界面，可使用各种**函数**、**结构**和**连线**实现运算和控制的功能，类似于仪器内部的线路连接，如图 2 所示。前面板的每个控件都对应后面板的一个函数图标用于数据传输，反之则不是。

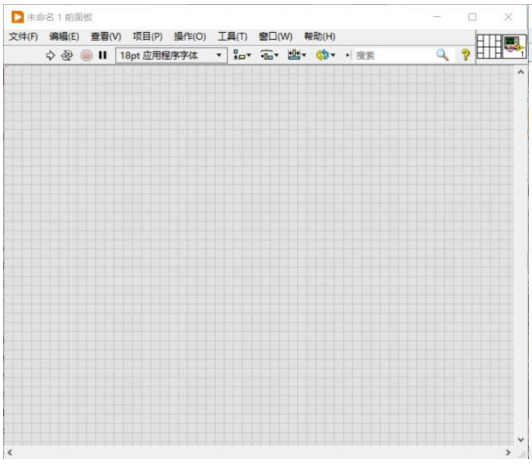


图 1 LabVIEW 前面板

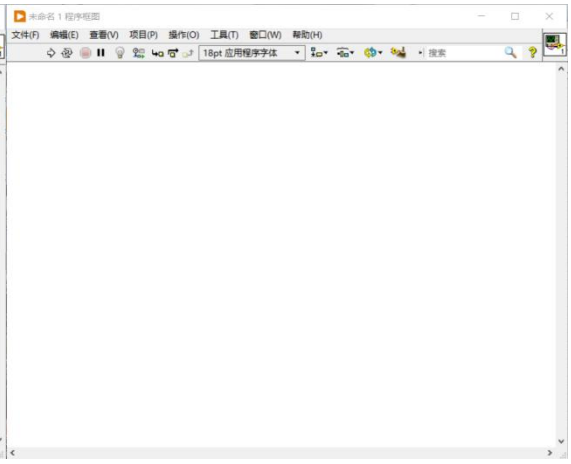


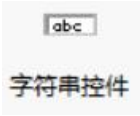
图 2 LabVIEW 程序框图

在前面板右键单击鼠标，可以放置各类控件；在后面板右键单击鼠标，可以放置各类函数、结构等。

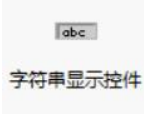
二、LabVIEW 常用控件

LabVIEW 控件是前面板上执行输入、显示、提示、装饰等功能的工具，可直接由用户（仪器使用者，而非编程者）阅读或控制。常用的控件有以下几种：

1）输入控件

控件	描述	数据类型
	用户向电脑输入一个数值，可以是整数或小数	数值（整形/浮点）
	用户点击可切换开关状态，开为 T，关为 F	布尔
	用户点下鼠标后变为 T，随后自动弹起变回 F	布尔
	功能与确定按钮相同，仅图示不同	布尔
	用户向电脑输入一个字符串	字符串

2）显示控件

控件	描述	数据类型
	显示一个数值，可以是整数或小数	数值（整形/浮点）
	显示一个布尔值，亮为 T，灭为 F	布尔
	显示一个字符串	字符串

3）数据容器

控件	描述	数据类型
	数组容器，可以放入一个输入或显示控件，变成对应的数组（可以是一维，也可以是二维） 例：数组容器中放入圆形指示灯，变成布尔数组显示控件，可以用一排指示灯显示一个布尔数组	数组（元素类型由放入的控件决定）

三、LabVIEW 常用函数

LabVIEW 函数是程序框图上执行运算、比较等功能的工具，相当于程序代码块或电子线路、运算模块等。常用的函数有以下几类：

函数类型	描述	输入类型	输出类型
	包含加、减、乘、除、求余等数值运算	数值	数值
	包含与、或、非等逻辑运算	布尔	布尔
	比较两个输入值之间的关系，如是否相等、是否 A 大于 B	数值	布尔
	包含 for 循环、while 循环、if 结构等常用函数结构，以及 LabVIEW 中十分常用的事件结构	任意	任意

四、数据传递

LabVIEW 程序框图中，要将数据在不同函数之间传输，需要进行线路连接。每个控件都对应一个函数，每个函数都有输入或输出接口。鼠标单击某一接口，再单击另一接口，即可连接一条数据线。在数据线上单击鼠标，即可创建该数据线的一个分支，连接其它接口。一条数据线只能连接一个输出接口，可以连接任意个输入接口。

图 3 展示了一个简单的 LabVIEW 程序，通过函数和数据接线实现两个数值相加，并输出结果。“数值 1”、“数值 2”为数值输入控件，“数值 3”为数值显示控件。

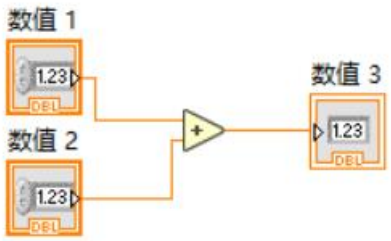


图 3 将两个数值相加并输出的 LabVIEW 程序框图

五、运行程序

顶部菜单栏左侧（前后面板均有）是程序运行控制按钮，如图 4 所示。从左至右分别为：运行（单次执行程序）、连续运行（循环执行程序直到按中止按钮为止，慎用）、中止按钮（强行停止程序，慎用）、暂停按钮、高亮执行按钮（实时展示数据流过程，调试程序用）。一般情况下，在前面板点击最左侧的运行按钮即可。

图 4 程序运行控制按钮

注意：运行程序前，请确保前面板上所有输入控件均已设定了值。若运行按钮显示断开（无法运行），说明程序存在错误，点击后可显示错误原因。

六、其它有用的事项

1）即时帮助

点击窗口右上角的问号图标（前后面板均有）后，将鼠标悬停在函数、线路等图标上，即可获得实时悬浮窗口的帮助提示。在即时帮助窗口也可以跳转到详细帮助文档。即时帮助也可通过菜单栏“帮助→显示即时帮助”打开。

2）从函数创建常量/控件

在后面板某个函数的某一个接线端右键单击，可弹出菜单，如图 5 所示。可以从该菜单直接“创建常量/输入控件/显示控件”，系统会自动创建一个适配于该函数的常量/控件，非常方便。

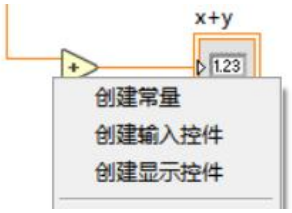


图 5 函数接线端的右键弹出菜单

3）删除断线/整理程序框图

在后面板的顶部菜单点击“编辑”，可以使用“删除断线（Ctrl+B）”和“整理程序框图”功能，为程序设计带来方便。

4）添加标签（注释）

后面板右键菜单“结构——修饰——自由标签”，可以添加标签（注释）。标签右下角有可拉伸的箭头，可以指向任意一个物体。

七、课后练习

1）用按钮控制小灯

要求：在前面板放置一个开关按钮（命名为：按钮），一个圆形指示灯（命名为：小灯）。进入连续运行模式，按钮按下时小灯熄灭，按钮弹起时小灯亮起。

提示：需要使用布尔函数“非”。

结果示例：如图 6 所示。

图 6 用按钮控制小灯

2）一元二次方程根的判别式

要求：输入 3 个数（a、b、c），对应一元二次方程 $ax^2 + bx + c = 0$ 的三个系数，单次运行后输出该一元二次方程根的判别式的值 Δ（delta）。$\Delta = b^2 - 4ac$。

提示：右键输入控件，选择“表示法”，将其更改为浮点型（一般用 DBL）。若为整型则无法输入小数。

结果示例：如图 7 所示。

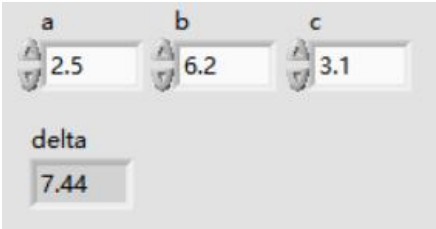


图 7 一元二次方程根的判别式

3）抽奖小程序

要求：输入一个值，代表中奖率百分比（0-100），单次运行后用字符串显示控件显示是否中奖。

提示：需要使用选择函数（在比较选板中）、0-1 随机数（在数值选板中）、某一比较函数（如小于等于）、字符串显示控件等。

结果示例：如图 8 所示。

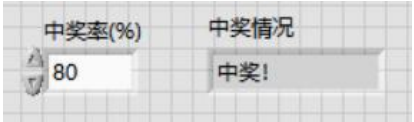


图 8 抽奖小程序

第一次课：LabVIEW 程序设计基础

一、条件结构

1）条件结构的基本知识

LabVIEW 的条件结构是根据输入“条件值”的不同，在几个平行分支中选择一个执行，相当于 C 语言中的 if...else 结构或 switch...case 结构。条件结构框图如图 9 所示。对应的 C 语言程序如图 10 所示。

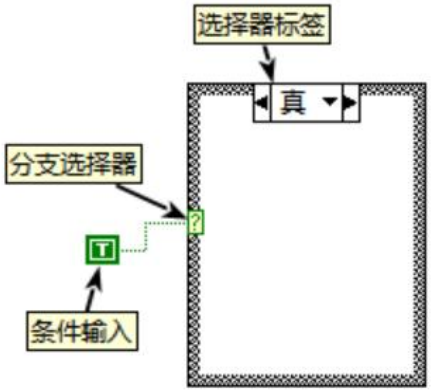


图 9 条件结构

```
1   if(条件1)
2   {
3     语句1;
4   }
5   else if(条件2)
6   {
7     语句2;
8   }
9   else
10  {
11    语句3;
12  }
```

图 10 条件结构对应的 C 语言代码

2）条件结构的分支

条件结构有一个**分支选择器**，可连接布尔值/数值/字符串等各种类型的条件输入。还有一个**选择器标签**，可以编辑在不同条件下所要执行的代码框图。整个条件结构类似于一组“平行世界”，根据条件不同，执行不同代码。

若条件输入为布尔类型，则条件结构只有“真”和“假”两个分支；若条件输入为数值或字符串类型，则条件结构可以有多个分支。可以编辑选择器标签的框，输入需要作为条件的数值。右键选择器标签，可以增加或删除分支。除非条件输入为布尔型，否则必须有一个“默认”分支，相当于 C 语言中 if...else if...else 中的最后一个 else（LabVIEW 条件结构中至少要有一个分支执行）。

图 11 展示了一个利用条件结构判断“大小月”的程序。注意 3 个框是一个条件结构的不同分支，并非 3 个条件结构。

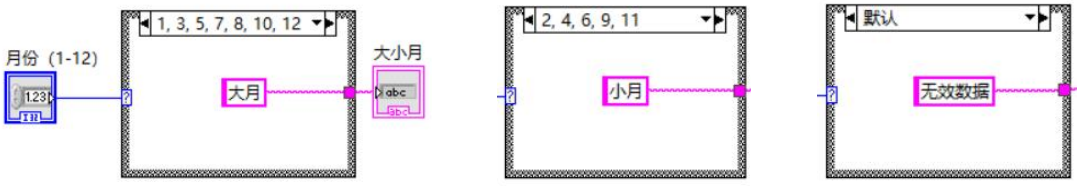


图 11 利用条件结构判断大小月（3 个分支）

二、循环结构

1）循环结构的基本知识

LabVIEW 的循环结构分为 While 循环和 For 循环两种，可以在后面板“结构”选板中找到。

While 循环：不断执行循环体（即循环结构框内的代码），直到一个 T（真）输入给右下角的**条件接线端**（红色圆点）后退出循环。循环条件判定发生在循环体执行完毕之后。右键单击条件接线端，可以切换为另一种模式（T 时继续）。While 循环至少会执行一次。普通 While 循环结构框图如图 12 所示。对应的 C 语言程序如图 13 所示。“T 时继续”模式的 While 循环结构框图如图 14 所示。对应的 C 语言程序如图 15 所示。

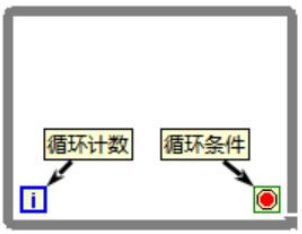


图 12 While 循环结构（T 时停止）

```c
while(true)
{
  //循环体
  if(结束条件) break;
}
```

图 13 While 循环结构对应的 C 语言代码

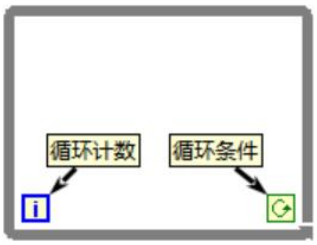


图 14 While 循环结构（T 时继续）

```c
do
{
  //循环体
} while(继续条件)
```

图 15 对应的 C 语言代码

For 循环：循环在执行指定次数后自动退出。该数值必须是一个整数，放在循环体外，连接至循环结构框左上角的“循环总数”接线端。For 循环结构框图如图 16 所示。对应的 C 语言程序如图 17 所示。

右键 For 循环的边框，也可以激活“条件接线端”，让 For 循环也具有按条件退出功能（相当于 C 语言的 break）。

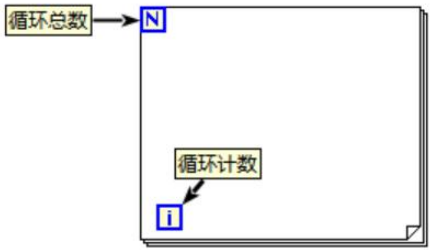


图 16 For 循环结构

```
1   for(int i = 0; i < 循环总数; i++)
2   {
3     //循环体
4   }
```

图 17 For 循环对应的 C 语言代码

两种循环的左下角都有**循环计数**，输出一个整数，表示当前循环是第几次循环（从 0 开始计数）。

图 18 展示了 While 循环和 For 循环，分别让其执行 100 次的基本代码。

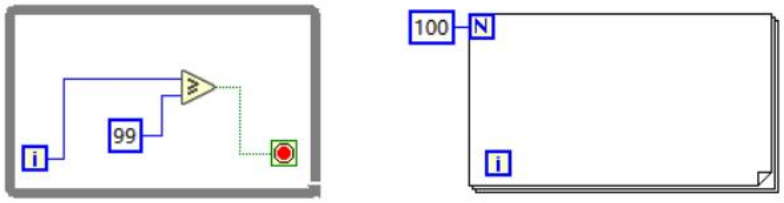


图 18 分别执行 100 次的 While 循环（左）和 For 循环（右）

2）循环隧道

循环隧道是循环内外的数据通道。当数据线连接循环内外时，会自动创建隧道。循环隧道有 2 种常用模式：**最终值**和**索引**。“最终值”是将最后一次循环时的值输出到循环外；“索引”是将每一次循环时的值放入一个数组中，在循环退出后输出一个数组。若循环隧道是输入通道，“索引”模式则是将输入的数组按顺序拆分为一个个元素。

右键单击循环隧道，可以更改隧道模式。同时隧道模式可以选择附带“**条件**”（一般配合“索引”模式使用）：只在条件接线端为 T 时，才将数据放入数组，可用于筛选需要输出的值。

几种隧道模式的图标有区别，如图 19 所示。

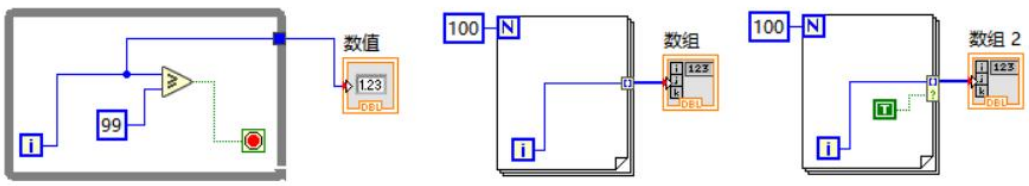


图 19 最终值模式（左）、索引模式（中）、条件索引模式（右）

3）移位寄存器

移位寄存器的作用是将本次循环时的某一变量保存下来，在下一循环时读取该变量，相当于将局部变量从循环末尾传递至下一循环初始。移位寄存器是成对出现的：右侧移位寄存器用于输入本次循环的值，也可以作为最终值模式的循环隧道；左侧移位寄存器可以读出上一次循环时存储的值，也可以从循环外接入初始值（一般情况下都需要接初始值）。图 20 展示了一对移位寄存器，用以实现从 0 开始不断累加的功能（直到 100）。

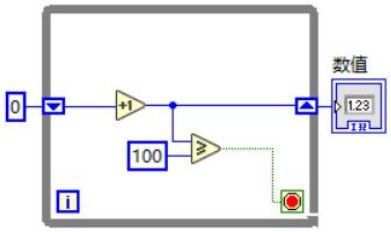


图 20 用一对移位寄存器实现累加

在循环结构的侧壁右键单击，弹出菜单中可以选择创建移位寄存器。也可以右键单击任一循环隧道，弹出菜单中可以将其更改为移位寄存器。

4）循环延时、循环计时

循环延时：后面板的“定时——等待(ms)”函数可用于让循环延时。将该函数放置在循环内任意位置，单次循环执行完毕，即会等待指定时间，再进行下一个循环。等待函数一般都是循环结构中最后一个运行的函数。“定时——等待下一个整数倍毫秒”也可以实现相似功能。

循环计时：“等待”函数可以输出毫秒计时值（计时零点是电脑开机时刻），利用移位寄存器保存上一循环的毫秒计时值，减去当前循环的毫秒计时值，即可得到一个循环所消耗的时间。

图 21 展示了让 While 循环每次等待 1s 并计时的程序框图，按下“停止”按钮可以退出循环，结束程序。

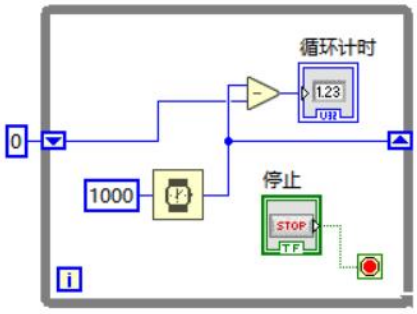


图 21 等待 1s 并计时

思考题：若在程序运行后的第 1.5 秒时按下“停止”按钮，程序会在第几秒时退出？提示：读取“停止”按钮的值发生在循环刚开始时，而等待函数是最后一个运行的。

5）循环结构练习例题

要求：输入一个正整数，对其进行分解素因数，输出一个由其所有素因数组成的数组。

思路：创建一个除数，初始值为 2。在 While 循环结构中，将被除数（输入的原数）不断与除数相除。如果能整除，商覆盖被除数；如果不能整除，被除数不变，除数加一。被除数与除数相等时结束循环，通过带条件的索引模式循环隧道输出除数数组（条件是能整除）。

答案（程序框图）如图 22 所示，运行效果（前面板）如图 23 所示。

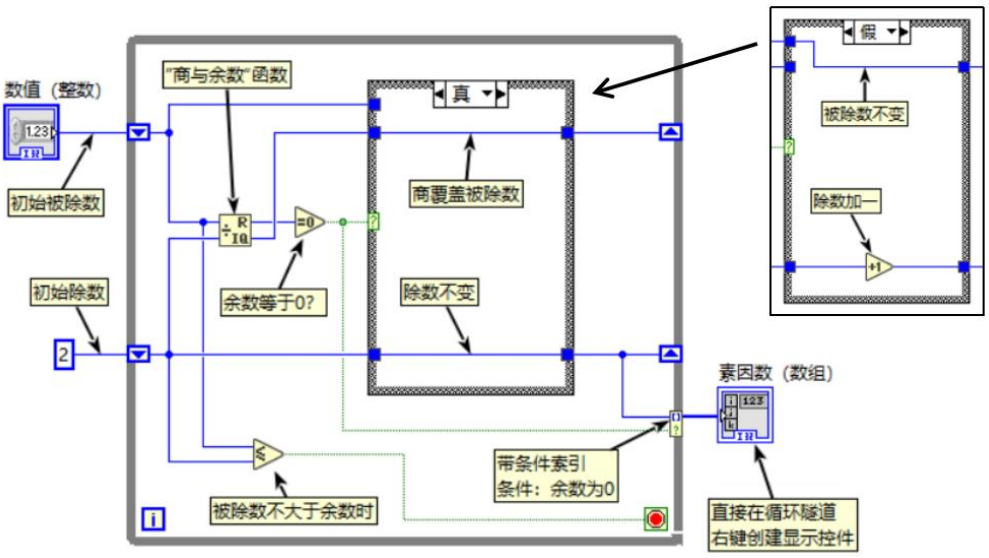


图 22 分解素因数程序框图

图 23 分解素因数程序运行效果

三、子 vi

在 LabVIEW 中，每个 vi 都可以作为子 vi 被其它 vi 程序调用。而且这种子

程序调用可以无限嵌套。被调用为子 vi 时，输入控件可以定义为输入接口，显示控件可以定义为输出接口。

1）定义子 vi

要将一个 vi 程序定义为可以被其它程序调用的子 vi，需要将其输入/显示控件定义为输入/输出接口，该操作在前面板完成。以“一元二次方程根的判别式”程序为例，可将其 3 个输入控件（数值 A/B/C）定义为 3 个输入接口；1 个显示控件（数值 delta）定义为输出接口。

在程序前面板右上角有接线面板（若没有，右键单击 vi 图标，在弹出菜单中可以调出），如图 24 红框内所示。

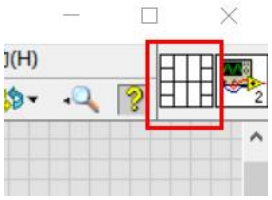


图 24 前面板接线面板

接线面板的左半部分用于定义输入接口，右半部分由于定义输出接口。右键单击接线面板，选择“模式”可以更改接口模式，可选的接口模式如图 25 所示。

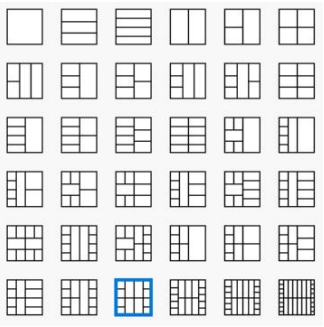


图 25 可选的接口模式

以“一元二次方程根的判别式”程序为例，选择第二行第二列的接口模式（3 输入 1 输出）。左键点击某一输入接口，再点击某一输入控件，即可将两者建立连接（输出接口同理）。已定义的接口会以数据类型对应的颜色显示。

所有输入输出接口定义完毕后，保存程序即可。

您也可以编辑程序图标，方法是右键单击右上角程序图标，选择“编辑图标”，即可进入绘图模式编辑图标。

2）调用子 vi

在其他 vi 程序的后面板中，可以调用子 vi。在后面板右键单击，选择列表

下方的“选择 VI...”，即可调用此前保存的 vi 程序，作为一个子 vi。“即时帮助”也会显示该子 vi 的相关帮助信息。如图 26 所示。

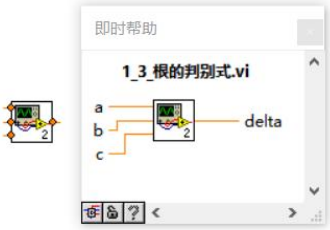


图 26 调用的子程序“根的判别式”

注意：子 vi 被调用时相当于一个函数，只有输入输出接口，不会包含任何控件。

3）子 vi 练习例题

要求：编写 LabVIEW 程序，调用“一元二次方程根的判别式”为子 vi，计算一元二次方程的两个解。若无实数解，会自动输出 NaN（Not-a-Number，非法数字）。

提示：可以在子程序的输入接口直接右键创建输入控件。

答案：如图 27 所示。

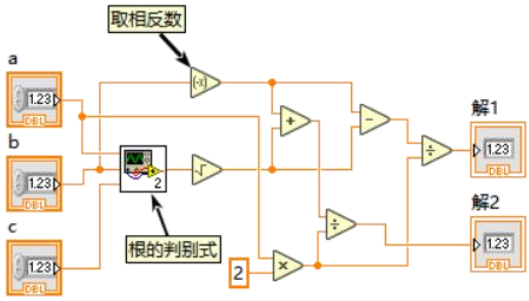


图 27 一元二次方程求解

四、事件结构

事件结构是 LabVIEW 特有的结构，可用于在各种事件发生时（如前面板的某个按钮被按下）执行结构内的程序，如图 28 所示。

图 28 LabVIEW 事件结构

本讲义只介绍事件结构最常用的功能：处理按钮事件。

1）事件结构的运行原理

事件结构可以指定多个事件（如同条件结构）。当程序执行到事件结构时，会开始等待某一事件发生，等待的过程不会影响其它并列的代码运行。当有一个指定事件发生时，事件结构就会执行该事件所对应的代码，执行完毕后退出事件结构（即一个事件结构一次只会对一个事件作出响应）。事件结构默认会有一个“超时”分支，若指定时间内没有发生任何事件，则执行超时分支并退出事件结构。“超时”分支的指定时间默认为-1，即永不超时。

事件结构有一个特殊性质：执行事件结构时（如超时发生时），会中断其它代码的执行（就算在循环过程中），先执行事件结构，执行完毕后再回到中断位置继续执行。

2）添加事件分支

事件结构可以用类似条件结构的方法添加事件分支。在前面板放置一个“布尔——确定按钮”，在后面板绘制一个事件结构（不要包住按钮）；右键单击选择器标签，选择“添加事件分支”，即可添加控件“确定按钮”的事件分支“值改变”。此时只要按钮的值改变，就会执行事件结构的该分支。

3）While 循环嵌套事件结构

由于一个事件结构只能处理一次事件，将事件结构放在 While 循环中，即可以实现让程序不断等待新事件发生。此时需要在前面板创建一个“停止按钮”，并在事件结构创建一个“停止按钮——值改变”分支，当停止按钮按下时，事件结构输出一个 T，以结束循环，否则 While 循环可能永远无法结束。

需要作为事件的按钮需要放在 While 循环内，才能保证其工作正常。

4）事件结构练习例题

控件：在前面板创建 2 个确定按钮，分别命名为“加 1”和“减 1”；创建一个“停止按钮”，命名为“停止”；创建一个数值显示控件，命名为“数值”。

要求：每次单击“加 1”按钮时，“数值”加 1；每次单击“减 1”按钮时，“数值”减 1；单击“停止”按钮时，程序停止。3 个按钮都不能有任何连线。

提示：1）需要使用 While 循环嵌套事件结构。2）要读取“数值”的值，可以在后面板右键该控件，在弹出菜单选择“创建——局部变量”。局部变量相当于是对该控件的一个引用。右键单击局部变量，在弹出菜单选择“转换为读取”，即可利用该局部变量读取“数值”的值。局部变量可以创建无限个，可以是读取

也可以是写入。3）事件结构的隧道可以右键选择“未连接时使用默认值”，此时若有某些分支未对隧道赋值，可直接使用默认值，程序不会报错。

答案：如图 29 所示。

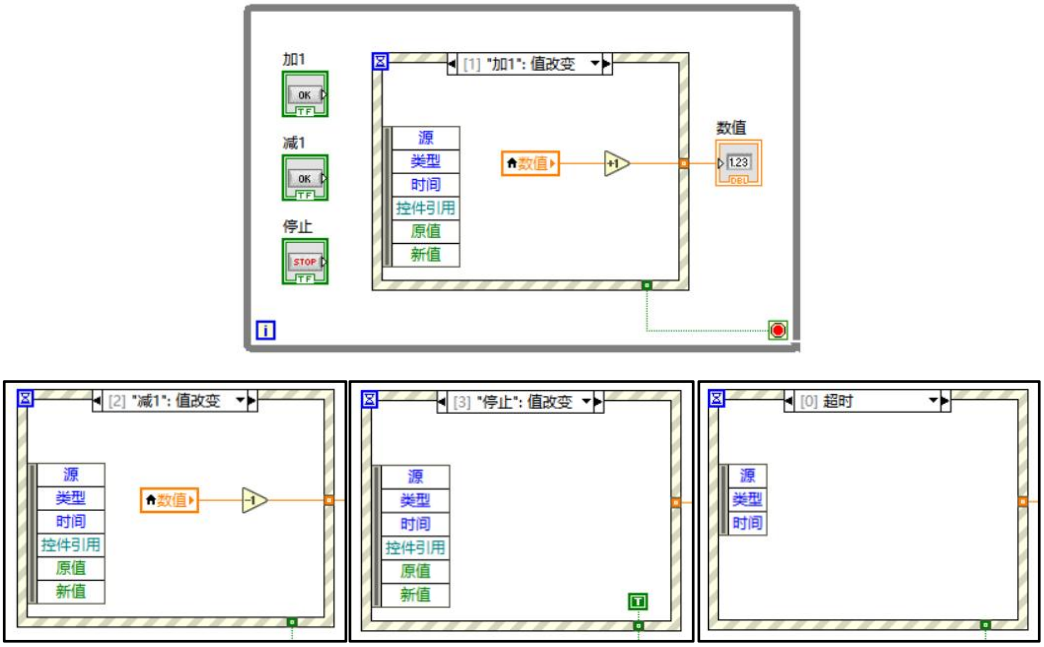


图 29 多按钮控制数值程序框图（小图为事件结构其它分支）

五、平铺式顺序结构

由于 LabVIEW 的程序并非自上而下的顺序结构，因此有时候难以控制两段程序的执行顺序。此时，平铺式顺序结构可以控制程序的执行顺序。

在后面板右键，可以创建平铺式顺序结构。右键单击平铺式顺序结构，可以选择“在后面添加帧”。此时，平铺式顺序结构的各帧之间将按照顺序执行。

六、二维图表

LabVIEW 提供了多种可以将数据以二维图表显示的控件，其中常用的如下表所示：

控件	描述	示例
波形图表	输入一个会改变的数值，生成该数值的变化历史曲线，每个循环刷新一次	温度变化曲线
波形图	输入一个一维数组，将其描绘成曲线	$y=f(x)$图像

XY图	输入捆绑的两个一维数组（1 个 X 数组，1 个 Y 数组），绘制 XY 图	椭圆、散点图
强度图	输入一个二维数组，内容是某点对应的强度，以数组索引为 XY 坐标绘制强度图	大气压分布图

本讲义将以示例介绍最常用的波形图表、波形图、XY 图的使用方法。

1）波形图表示例

要求：绘制一个随机数随时间变化的曲线，每 100ms 刷新一次。

提示：“波形图表”可以直接输入数值，需要使用 While 循环。

答案：如图 30 所示。

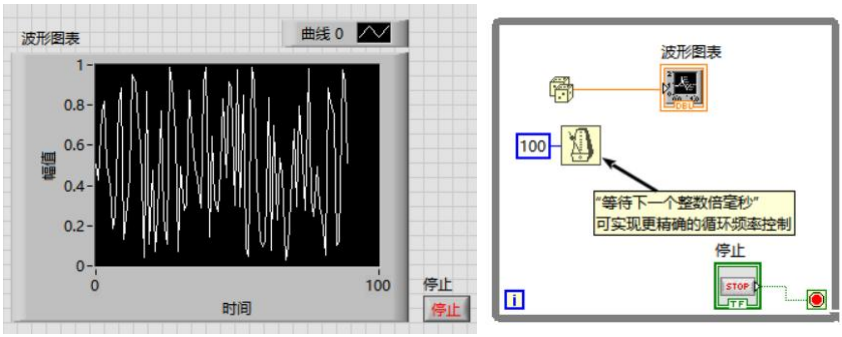


图 30 波形图表显示随机数变化

2）波形图示例

要求：绘制 $y=\sin(x)$ 的图像，x 取值从 0 到 2pi，分 100 个点。

提示：“波形图”需要输入一个数组，需要使用 For 循环。

答案：如图 31 所示。

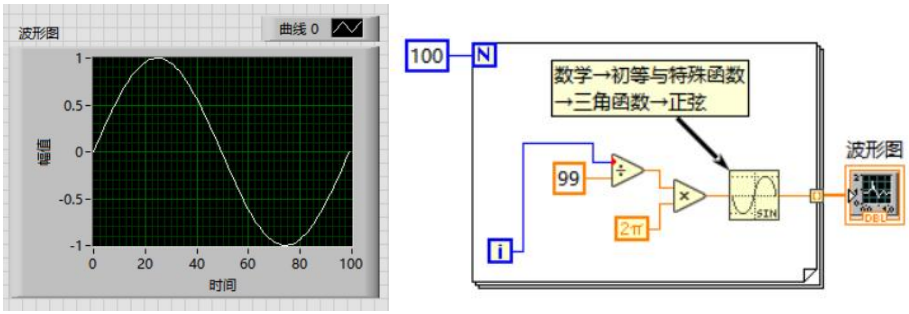


图 31 波形图显示正弦函数

3）XY 图示例

要求：绘制椭圆 $x=\sin(t)$, $y=2\cos(t)$ 的图像

提示：1）需要将两个一维数组捆绑成“簇”，使用“簇、类与变体——捆绑”函数；2）XY 图的坐标轴名称等可以右键单击 XY 图，在“属性”中修改。

答案：如图 32 所示。

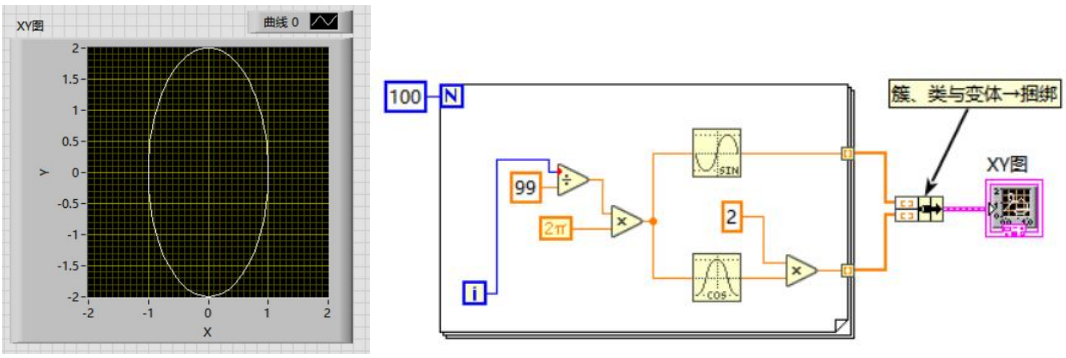


图 32 XY 图绘制椭圆

六、课后练习

1）素数统计

要求：将“分解素因数”打包为子程序，调用该子程序，统计整数 A 到 B 之间（包括两者）的所有素数。A 和 B 由用户输入，输出 1 个包含所求素数的数组，以及 1 个表示素数数量的数值。需要考虑输入零或负数的情况。

2）蒙特卡洛法计算圆周率

要求：在 $-1 < x < 1$，$-1 < y < 1$ 的范围内生成 N 个随机点，N 由用户指定（1000-100000）。用 XY 图绘制这些随机点的分布，并通过统计随机点落在 $x^2 + y^2 < 1$ 范围内的概率来估算圆周率。

第二课 LabVIEW 与 Arduino 的连接

一、Arduino 串口通信函数

这一部分将在 Arduino IDE 中讲述 Arduino 串口通信相关的内容。

1）串口初始化

在 Arduino 中，串口初始化函数为 Serial.begin()，其参数为波特率（Baud rate），即通信速率，一般为 9600。该函数一般在 void setup 中使用，如图 33 所示。若要追求更高的通讯速率，波特率可以设置为 115200。

```
void setup() {
  Serial.begin(9600);
}
```

图 33 Arduino 串口初始化函数

2）串口发送

让 Arduino 向串口发送指令的函数为 Serial.print()，参数为一个字符串。也可以使用 Serial.println()，该函数会在发送完字符串后添加一个回车符和一个换行符。如图 34 所示。

```
void setup() {
  Serial.begin(9600);
  Serial.println("Hello");
}
```

图 34 串口初始化及发送字符串

3）串口接收

Arduino 接收串口数据比发送要复杂一些，一般会通过定义函数 void serialEvent()来实现。Arduino 会在每个 loop 完成之后，检测串口是否发来了数据，若有，就会执行 serialEvent()。

以下提供了 serialEvent()函数的一种常用定义方法，可以直接照搬，其原理在注释中描述，如图 35 所示。

```
void serialEvent(){ //当收到串口数据时执行
  while(Serial.available()){ //如果缓存区有数据未读
    char inChar = (char)Serial.read(); //读取一个字节
    if (inChar == '\n') { //是换行符，结束读取
      stringComplete = true; //有未处理的字符串了
      break; //退出读取循环，回到主程序
    }
    else { //不是换行符，继续读取
      inputString += inChar; //将读到的该字节添加至字符串
    }
  }
}
```

图 35 serialEvent()函数的一种常用写法

同时需要定义两个全局变量用以存放数据，如图 36 所示。

```
String inputString = ""; //存储接收到的字符串
bool stringComplete = false; //是否有接收但未处理的字符串
```

图 36 定义两个全局变量以存放数据

注意 void serialEvent()函数一般放在 void loop()之后，即程序最末尾。

图 37 展示了一段完整的 Arduino 代码，其作用是接收到电脑发送的串口数据（以换行符结束）后，将其发回给电脑（也是以换行符结束）。

```arduino
String inputString = ""; //存储接收到的字符串
bool stringComplete = false; //是否有接收但未处理的字符串

void setup() {
  Serial.begin(9600);
}

void loop() {
  if(stringComplete){ //如果有待处理的字符串
    Serial.println(inputString); //串口把数据发回
    inputString = ""; //清空已处理的字符串
    stringComplete = false; //字符串处理完毕
  }
}

void serialEvent(){ //当收到串口数据时执行
  while(Serial.available()){ //如果缓存区有数据未读
    char inChar = (char)Serial.read(); //读取一个字节
    if (inChar == '\n') { //是换行符，结束读取
      stringComplete = true; //有未处理的字符串了
      break; //退出读取循环，回到主程序
    }
    else { //不是换行符，继续读取
      inputString += inChar; //将读到的该字节添加至字符串
    }
  }
}
```

图 37 发回字符串的 Arduino 程序

二、LabVIEW 串口通信函数

LabVIEW 的串口通信在 VISA 函数包中，位置是“后面板→仪器 I/O→VISA”。若无特别说明，该章节后续所有用到的函数均在 VISA 选板中。

VISA 函数	描述
	向目标设备写入一个字符串，可以输出写入的字符串长度。
	从目标设备中读取一个字符串，需要指定字节数，输出读取到的字符串。如果读取过程中遇到结束符（默认为换行符），则立刻结束读取。
	在“仪器 I/O→串口”中。配置串口波特率并进行初始化。也可以在此指定读取时的结束符（默认为换行符）。

	在“VISA→高级 VISA”中。断开与指定设备的通讯。在关闭程序之前，这个函数很重要，否则下一次建立连接将失败。
	在“仪器 I/O→串口”中。获取目标串口缓冲区中待读取的字节数。可用于判断是否有数据未读取。
	清空目标设备的串口缓冲区。以免缓冲区有过多数据未读，导致读到的是历史数据。

使用 LabVIEW 与 Arduino 建立串口通信有常用的程序框图模板，如图 38 所示。该模板非常实用，可以应用于几乎所有后续内容中。

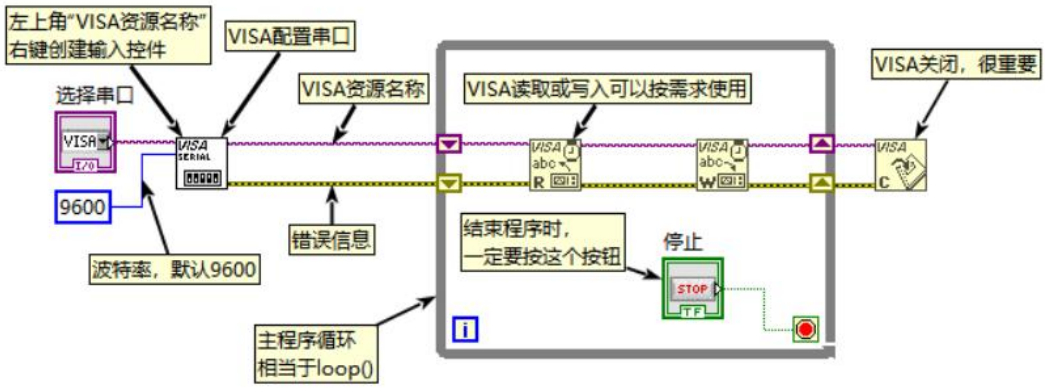


图 38 LabVIEW 串口通信程序模板

在使用该模板编写 LabVIEW 程序时，注意正常结束程序时，必须通过“**停止按钮**”来结束主循环，使得“VISA 关闭”函数能够正常运行。若使用 LabVIEW 界面自带的“中止执行”按钮，则会导致串口不能正常关闭，影响下一次串口连接。除非迫不得已（程序有错误导致主循环无法正常结束），否则请不要直接使用“中止执行”按钮。若发生了以上情况，可以创建一个仅有“VISA 关闭”函数和“选择串口”控件的程序，选择目标端口并运行一次，以将串口正常关闭。

三、LabVIEW 与 Arduino 的通信实验

图 39 展示了一个利用上述模板，与 Arduino 之间建立简单通讯的程序前面板，其程序框图如图 40 所示。其功能是：发送一个以换行符结尾的字符串给 Arduino，再从 Arduino 读取字符串直到出现换行符。该程序可配合 Arduino 回发字符串程序使用，以确认双方均工作正常。主循环前通过一个顺序结构让电脑等待 2s 时间，确保 Arduino 初始化完成，否则后续通信可能出错。

图 39 简单的串口通信实验

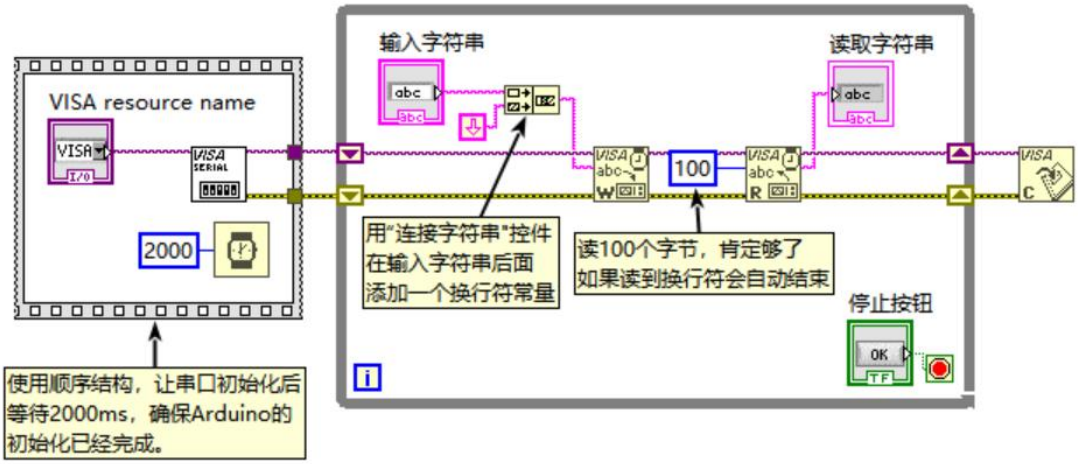


图 40 简单串口通信程序框图

值得一提的是，一般情况下“VISA 资源名称”和“错误信息”两个数据是在所有 VISA 函数之间串联的。由于两者会在 While 循环中反复调用，因此最好在进出循环时都使用移位寄存器（若使用隧道也能运行，但可能无法传递错误信息）。若您需要监测错误信息，可以在其连线上右键创建一个显示控件。

在运行与串口通信相关的程序之前，都需要在前面板**选择串口**（VISA 资源名称）。

在连接 Arduino 并运行程序之后，Arduino 需要约 2 秒时间初始化。此段时间内不要按任何按键，否则可能出错。

四、LabVIEW 对 Arduino 的控制和数据采集

以下将通过一个例子讲解 LabVIEW 如何对 Arduino 进行控制和数据采集。

控件：一个开关按钮，命名为“按钮”；一个数值显示控件，命名为“计数”。

要求：按下按钮时，Arduino 板载小灯（位于 D13）亮起；按钮弹起，小灯熄灭。同时 Arduino 统计按钮被操作了几次，将数字发回电脑，由“计数”控件

显示。注：禁止用 LabVIEW 直接计数。程序效果如图 42 所示。

图 42 小灯控制并计数程序效果

提示：1）需要使用事件结构；2）需要编写 Arduino 程序；3）Arduino 需要判断输入的字符串是否与预期相同，可以使用语句“inputString.equals()”，该函数返回一个布尔值。

答案：LabVIEW 程序（主要部分）如图 43 所示，事件结构其它分支略。Arduino 程序（serialEvent 略）如图 44 所示。

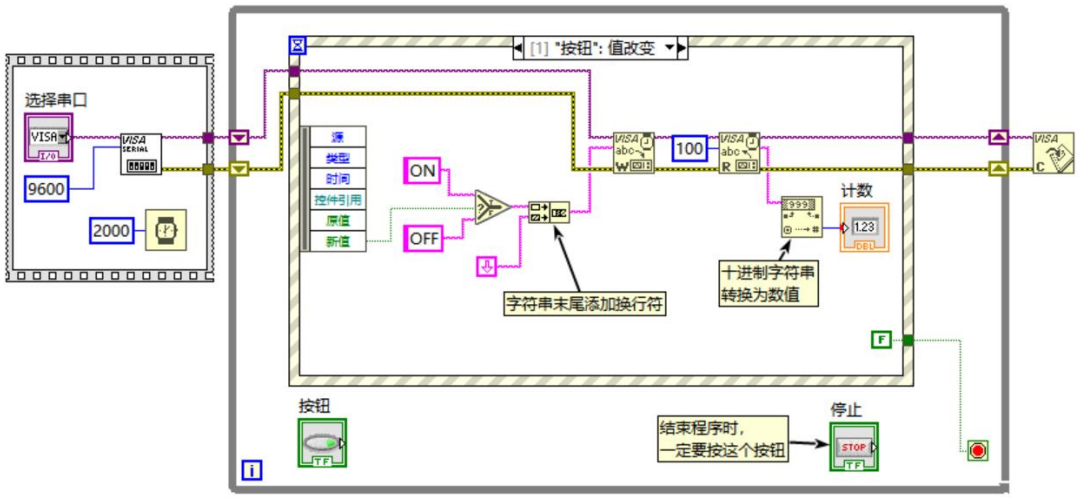


图 43 LabVIEW 控制 Arduino 小灯并计数（LabVIEW 部分）

```arduino
String inputString = ""; //存储接收到的字符串
bool stringComplete = false; //是否有接收但未处理的字符串
int count = 0; //计数菌
void setup() {
  Serial.begin(9600);
  pinMode(13,OUTPUT);
}
void loop() {
  if(stringComplete){ //如果有待处理的字符串
    if(inputString.equals("ON")){ //换行符没录入字符串，否则要equals("ON\n")
      digitalWrite(13,HIGH);
      count++;
      Serial.println(count);
    }
    else if(inputString.equals("OFF")){
      digitalWrite(13,LOW);
      count++;
      Serial.println(count);
    }
    inputString = ""; //清空已处理的字符串
    stringComplete = false; //字符串处理完毕
  }
}
```

图 44 LabVIEW 控制 Arduino 小灯并计数（Arduino 部分）

六、课后练习

1）模拟噪声采集

要求：Arduino 每 100ms 读取一次 A0 模拟引脚的电压值。若 A0 引脚未连接，所测得的电压为噪声信号。将测得的值发送到 LabVIEW，并以波形图表显示该噪声值的变化。

2）定时控制小灯

要求：LabVIEW 放置一个按钮，一个数值输入控件。点击按钮后让 Arduino 板载小灯亮起，一段时间后熄灭，亮起的时间为用户输入的数值（单位 ms）。若小灯未灭时再次点击按钮，则重新开始计时。

第三课 搭建信号测量与仪器控制系统

一、读取简单传感器（热敏电阻）

以下将通过一个例子讲解 LabVIEW 读取连接至 Arduino 的传感器。

控件：一个数值显示控件，命名为“温度/℃”。

要求：运行程序后，LabVIEW 以每秒 2 次的刷新频率显示热敏电阻测得的温度。Arduino 只允许发送 analogRead 读到的原始值（0-1023），所有计算在 LabVIEW 程序完成。运行效果如图 45 所示。

图 45 读取热敏电阻程序效果

提示：通过热敏电阻阻值计算温度的公式：$T = 1/(\ln(R/R_0)/B + 1/T_0)$，其中 T 为当前温度（单位 K），$T_0 = 298.15$K（25℃），R 为当前阻值（单位 kΩ），$R_0 = 10$kΩ，$B = 3950$。

答案：LabVIEW 程序（主要部分）如图 46 所示。Arduino 程序如图 47 所示。

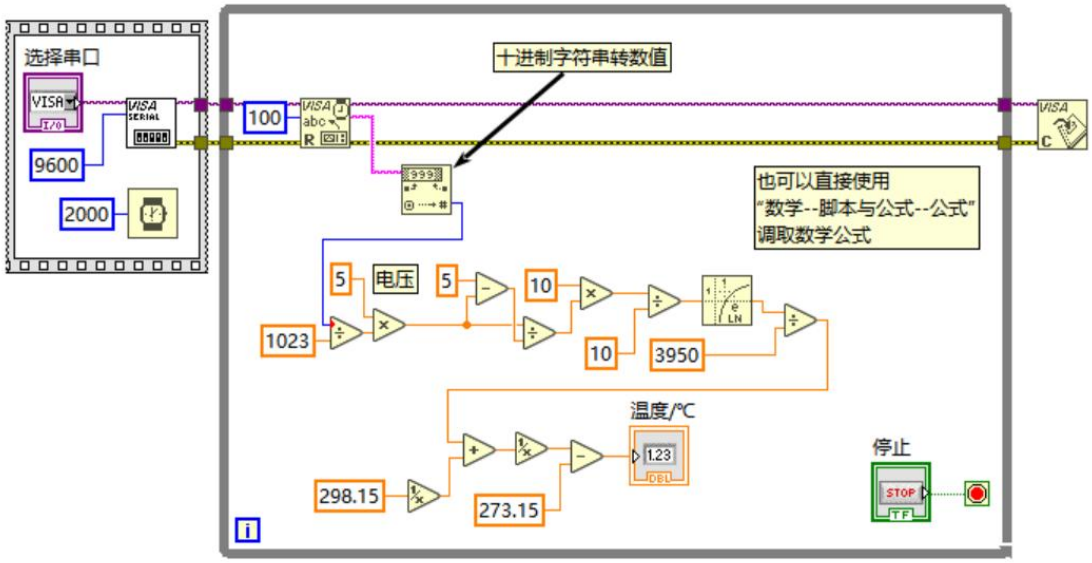


图 46 读取热敏电阻（LabVIEW 部分）

```arduino
void setup() {
  Serial.begin(9600);
}

void loop() {
  int v = analogRead(A0);
  Serial.println(v);
  delay(500); //刷新频率约2Hz
}
```

图 47 读取热敏电阻（Arduino 部分）

二、读取复杂协议传感器（BMP180）

有一些传感器并非以数字量或模拟量形式传递数据，而是借助通信协议传递更复杂的数据。如 BMP180 气压传感器，通过 I2C 协议可以传输当前气压、温度、海拔高度这 3 个物理量。

在 Arduino 中，使用 BMP180 传感器有对应的库（Adafruit_BMP085.h），使用起来非常简便。将传感器的 SDA 连接 A4，SCL 连接 A5，VCC 和 GND 连接电源。向 Arduino 写入如图 48 所示程序，即可读取以上 3 个物理量。

```arduino
//SDA连接A4
//SCL连接A5
#include <Adafruit_BMP085.h>
Adafruit_BMP085 bmp;

void setup() {
  bmp.begin(); //初始化BMP180或BMP085
}

void loop() {
  float T = bmp.readTemperature(); //读取温度
  float P = bmp.readPressure(); //读取气压
  float H = bmp.readAltitude(); //读取海拔高度
  delay(1000); //每秒刷新一次
}
```

图 48 读取 BMP180 传感器的简单程序（仅读取无输出）

要使用 LabVIEW 读取其数据，仅需将得到的数据通过串口以一定规则发送给电脑即可。这里为了区别 3 个物理量，在发送其数据之前发送一个关键字（T/P/H）来表示之后的数据是哪个物理量。

完整的一帧数据如下表所示：

‘T’	温度数值	‘P’	气压数值	‘H’	海拔数值	换行符\n

发送数据的 Arduino 代码如图 49 所示。

```
Serial.print("T"); //标识符“T”
Serial.print(T,1); //温度取一位小数
Serial.print("P"); //标识符“P”
Serial.print(P,0); //气压取整数
Serial.print("H"); //标识符“H”
Serial.print(H,1); //海拔取一位小数
Serial.println(""); //单独发送一个换行符
```

图 49 Arduino 发送 BMP180 数据的代码

请你编写 LabVIEW 程序以实现读取这些数据，以数值显示控件的方式显示在前面板上，如图 50 所示（由于传感器尚未严格定标，部分数据不准确，仅作效果展示）。

图 50 读取 BMP180 气压传感器

LabVIEW 程序如图 51 所示。

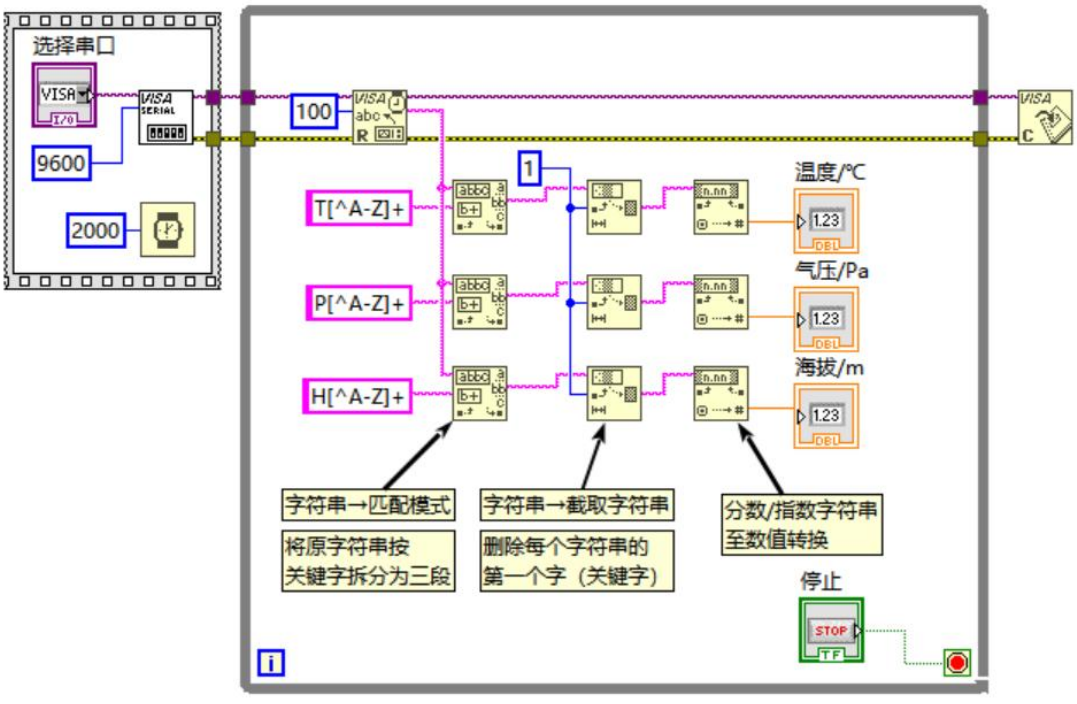


图 51 读取 BMP180 的程序框图

三、仪器设备控制（步进电机）

步进电机是一种能够以固定“步进角度”一步一步旋转的电机，因此可以很精确地控制其转动的角度，常用于运动控制。在步进电机控制器的作用下，可以实现“每个脉冲信号转动一步”。

下表是一类常用的步进电机控制器的接线方法：

控制器接口	接线描述	控制器接口	接线描述
EN+	使能线正极，接高电平则关断输出电流，可以不接。	V+	步进电机电源正极，一般接12V。
EN-	使能线负极，接 GND 或不接。	V-	步进电机电源负极。
DIR+	方向线正极，接高电平/低电平对应步进电机不同旋转方向。	A+	步进电机 A 相正极线。
DIR-	方向线负极，接 GND。	A-	步进电机 A 相负极线。
PUL+	脉冲信号线正极，每一次上升沿（低电平转换为高电平）时步进电机旋转一步。	B-	步进电机 B 相负极线。
PUL-	脉冲信号线负极，接 GND。	B+	步进电机 B 相正极线。

现在请你将 Arduino 的 D5 引脚连接 PUL+，D7 引脚连接 DIR+，并按照说明连接其它应当连接的导线。编写 Arduino 和 LabVIEW 程序，实现以下功能：

用户在前面板指定脉冲数（1-9999）、脉冲频率（1-1000Hz）和旋转方向。点击“确定”按钮后，系统控制步进电机按照要求旋转。

提示：需要使用函数 delayMicroseconds，以微秒为单位进行延时，但输入的数字不能超过 16383，否则会不准。若延时超过这个数值，只能使用 delay。

答案示例：LabVIEW 程序框图如图 52 所示，Arduino 程序（serialEvent 略）如图 53、图 54 所示。

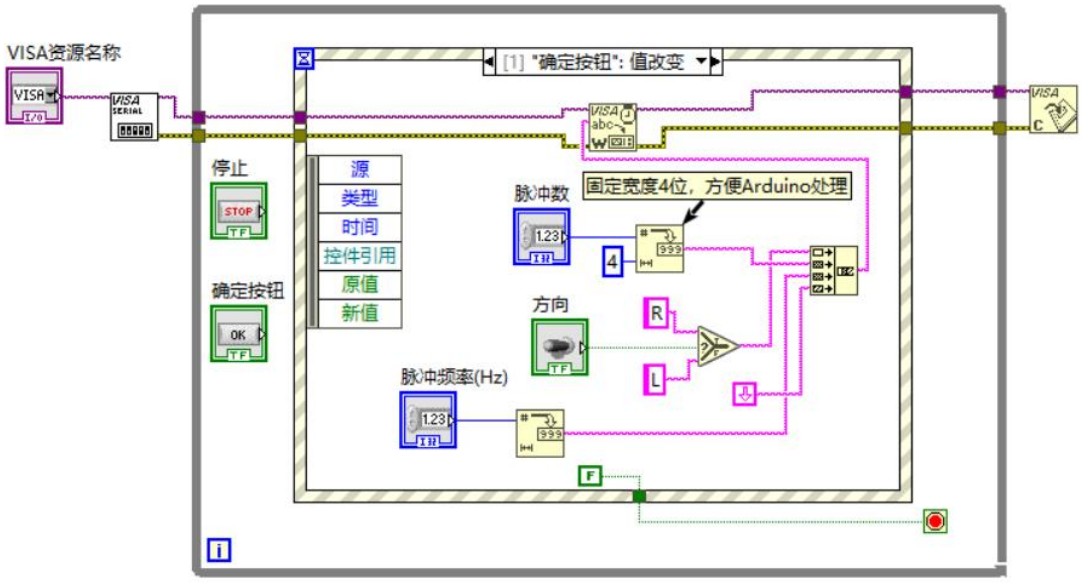


图 52 步进电机控制的 LabVIEW 程序（事件结构其它分支略）

```
#define dirpin 7 //方向控制接口
#define pulpin 5 //脉冲输出接口

String inputString = "";
bool stringComplete = false;
bool dir; //旋转方向(T/F)
int num; //脉冲数
int spd; //脉冲频率（单位Hz）

void rotate(bool dir,int num){
  //根据dir的值（T/F）确定方向，通过dirpin(D7)控制
  if(dir) digitalWrite(dirpin,HIGH);
  else digitalWrite(dirpin,LOW);
  //以指定速度和指定数量创建脉冲序列
  if(spd <= 30){ //用delayMicroseconds会超过16383
    int delaytime = 500/spd; //延时毫秒数
    for(int i=0;i<num;i++){
      digitalWrite(pulpin,HIGH);
      delay(delaytime);
      digitalWrite(pulpin,LOW);
      delay(delaytime);
    }
    //一次写HIGH一次写LOW以创建一个脉冲
  } else { //可以用delayMicroseconds
    int delaytime = 500.0/(float)spd*1000; //延时微秒数
    for(int i=0;i<num;i++){
      digitalWrite(pulpin,HIGH);
      delayMicroseconds(delaytime);
      digitalWrite(pulpin,LOW);
      delayMicroseconds(delaytime);
    }
  }
}
```

图 53 步进电机控制的 Arduino 程序（第 1 部分）

```
34
35  void setup() {
36    // put your setup code here, to run once:
37    Serial.begin(9600);
38    pinMode(dirpin,OUTPUT);
39    pinMode(pulpin,OUTPUT);
40    digitalWrite(dirpin,LOW);
41    digitalWrite(pulpin,LOW);
42  }
43
44  void loop() {
45    // put your main code here, to run repeatedly:
46    if(stringComplete){
47      if(inputString.startsWith("L")) dir=false; //判断旋转方向
48      else if(inputString.startsWith("R")) dir=true;
49      inputString = inputString.substring(1); //删去字符串首字母
50      String numstr = inputString.substring(0,4); //字符串前4位，即脉冲数
51      num = numstr.toInt(); //脉冲数
52      inputString = inputString.substring(4); //删去前4个字母
53      spd = inputString.toInt(); //脉冲速度
54      rotate(dir,num); //执行旋转
55      inputString = "";
56      stringComplete = false;
57    }
58  }
```

图 54 步进电机控制的 Arduino 程序（第 2 部分）

四、课后练习暨综合课题项目

自主选题，完成一个基于 LabVIEW 和 Arduino 的数据采集或仪器控制综合项目，所花费的工作量应在 3-5 小时左右。同时需要形成一个基于该项目的纸质实验报告/口头报告，作为本实验评分的重要依据。

实验报告/口头报告应当包含以下内容：

1）项目简介：该项目使用了哪些传感器/仪器设备？实现了哪些功能？

2）Arduino 程序原理：程序结构和功能？输入输出如何配合硬件电路？如何实现通讯？

3）LabVIEW 程序原理：程序结构和功能？如何实现通讯？如何实现数据运算和处理？如何在前面板展示数据？

4）创新点：该项目在何处体现了 LabVIEW 配合 Arduino 实现数据采集或仪器控制的优势？

5）收获、反思与展望：开展项目获得的收获？项目中遇到的问题？日后项目可能会有何种改进？

可供参考的项目如下：1）基于 HX711 模块的测力传感器数据采集；2）带调速和测速功能的风扇转速闭环控制；3）带热敏电阻的加热棒温度 PID 控制；

4）基于 ADS1115（4 通道 16 位 ADC）的简易多通道示波器；5）其它任何感兴趣的项目。