第 5 章

myRIO 系统级通信技术应用

主要内容

- UART 通信基本原理，myRIO 中 UART 通信端口配置情况；
- myRIO 工具包中 UART 通信 ExpressVI、底层 VI 的配置和使用方法；
- UART 通信技术相关工程项目开发实战；
- WiFi 通信技术基本原理，myRIO 中 WiFi 通信功能配置方法，通信程序设计相关 VI；
- 无线局域网下基于 WiFi 连接的 TCP、UDP 通信技术相关工程项目开发实战；
- 物联网环境下基于 WiFi 连接的 TCP、UDP、MQTT、HTTP 等协议相关工程项目开发实战。

5.1 UART 通信技术及应用

myRIO 在提供丰富 I/O 端口的同时，还提供了 2 组 UART 通信端口，这使得 myRIO 具备较为强大的系统级通信能力（当然也可作为器件级通信端口使用，用以实现 myRIO 和电子功能模块或器件之间的通信），可以快速开发 myRIO 与其他应用系统/功能模块之间的通信，极大地丰富和扩展了 myRIO 应用功能。

本节在简要介绍 UART 通信技术概念的基础上，给出 myRIO 中 UART 通信端口配置情况，实现 UART 通信的 ExpressVI 及其调用方法、UART 通信相关的若干底层 VI 及其应用的一般流程，并结合自发自收 UART 通信、淘晶驰触摸屏控制、蓝牙无线串口屏 3 个实例介绍 UART 通信程序设计的基本方法。

5.1.1 UART 通信技术概述

UART 意为通用异步收发传输器（Universal Asynchronous Receiver/Transmitter）。作为异步串口通信协议的一种，其基本工作原理是按位传输字节或字符数据。每字节数据的传输由空闲位、起始位、数据位、校验位、停止位 5 部分组成。UART 通信时，必须保证通信双方具有一致的通信参数设置（数据位、校验位、停止位、波特率等参数）。绝大多

数电子设备通信参数为“9600,N,8,1”,其中 9600 指的是通信波特率为 9600,即每秒传输 9600bit; N 表示无校验; 8 表示数据位由 8bit 组成; 1 表示停止位为 1bit。

UART 通信时,通信线路连接仅需 3 根线——UART-TX(发送数据线)、UART-RX(接收数据线)、GND(地线)。通信线路采取交叉连接方式,即本机 UART-TX 连接对方设备 UART-RX; 本机 UART-RX 连接对方设备 UART-TX; 通信双方共地连接,确保 TX、RX 信号具有统一的参考点。

myRIO 提供了 2 个 UART 通信端口,一个是 A 口的 Pin10(RX)和 Pin14(TX),另一个是 B 口的 Pin10(RX)和 Pin14(TX),myRIO 中 UART 通信端口分布如图 5-1 所示。

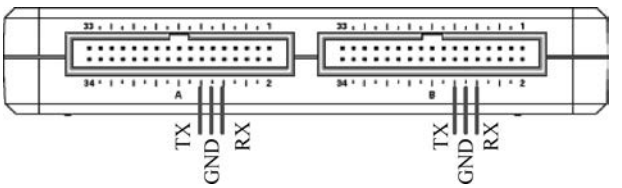

图 5-1 myRIO 中 UART 通信端口分布

5.1.2 主要函数节点

myRIO 中 UART 通信程序实现有两种方法,第一种方法是使用 myRIO 工具包中的 UART 通信 ExpressVI,如图 5-2 所示。

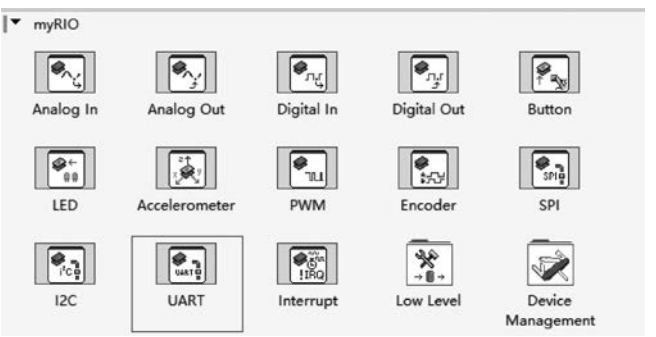

图 5-2 myRIO 中的 UART 通信 ExpressVI

将该节点拖曳至程序框图,即可弹出 UART 通信 ExpressVI 配置窗口,如图 5-3 所示。

第二种方法是使用 myRIO 工具包中提供的底层函数节点(函数→myRIO→Low Level→UART)实现 UART 通信,底层函数子选板中的 UART 通信相关 VI 如图 5-4 所示。

基于底层函数进行 UART 通信时,其数据发送基本流程为“VISA 配置串口→VISA 写入/VISA 读取→VISA 关闭”。如果需要连续读写 UART 通信端口,则将节点 VISA 写入/VISA 读取置于 While 循环结构中即可。

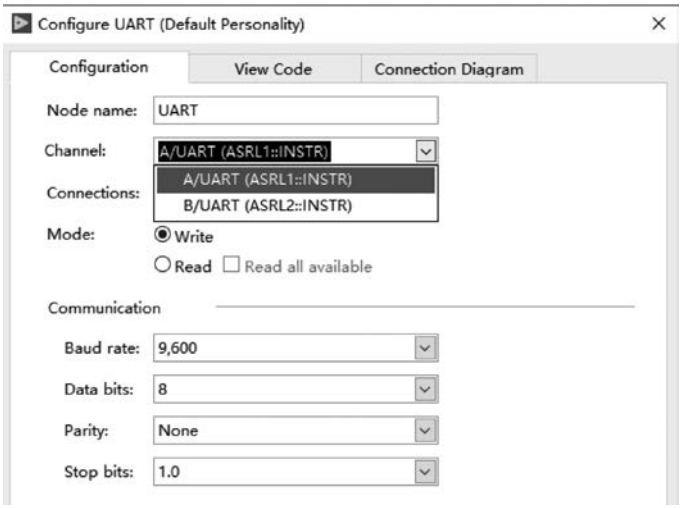

图 5-3　UART 通信 ExpressVI 配置窗口

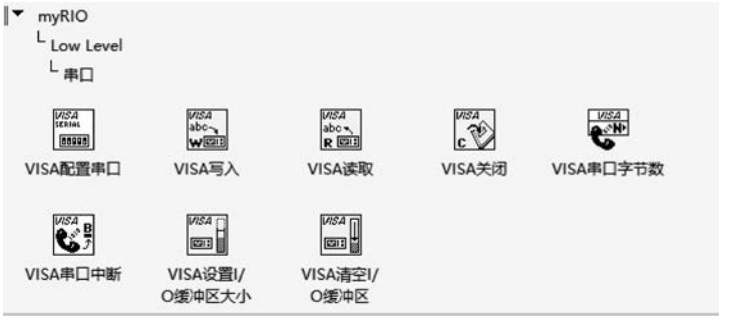

图 5-4　底层函数子选板中的 UART 通信相关 VI

5.1.3　UART 通信技术应用实例

借助 myRIO 中提供的 UART 通信端口，既可以实现 myRIO 与其他 UART 设备之间的点对点串行通信，又可以外接 UART 接口的 TTL-485 模块、蓝牙模块、GPRS 模块、NB-IoT 模块、ZIGBEE 模块等进一步扩展 myRIO 开发平台的通信能力，将 myRIO 作为更大规模技术系统的组成部分。以下将通过自发自收、淘晶驰串口屏、蓝牙设备 3 个案例展示 UART 通信程序编写方法。

1. myRIO 自发自收 UART 通信程序

1）设计目标

本案例通过短接 UART 接口 TX(发送)引脚和 RX(接收)引脚的方式，构建自发自收通信连接，并编写程序实现 UART 通信端口的数据发送和接收功能。自发自收是一种简单、有效的通信功能测试方法，通常用于通信功能的早期开发。

2）硬件连线

本案例选择 myRIO 开发平台 A 口提供的 UART 接口，Pin10 为 RX，Pin14 为 TX，使用双母头杜邦线连接 RX 引脚和 TX 引脚。自发自收 UART 通信硬件连线如图 5-5 所示。

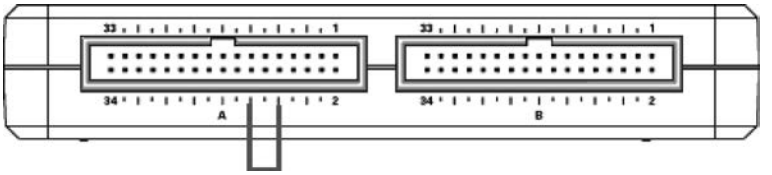

图 5-5　自发自收 UART 通信硬件连线

3）设计思路

利用 myRIO 新建项目自动生成的程序模板，保留其三帧程序基本结构。第一帧为初始化帧，主要用于有关数据、控件的初始赋值；第二帧为主程序帧，为 While 循环结构下的应用程序核心业务功能开发，即采集数据，并将采集数据通过 A 口 UART 通信端口进行发送，然后通过同一 UART 通信端口接收数据；第三帧完成程序结束运行后的设备重置功能。

4）程序实现

程序实现可分解为“采集数据、封装发送信息、串口发送、串口接收、定时控制”等步骤。

（1）采集数据。采取 0～100 随机数产生的方式模拟数据采集，调用函数节点“转换为无符号单字节整型”（函数→编程→数值→转换→转换为无符号单字节整型）将采集的模拟数据转换为单字节整数，调用函数节点“数值至十进制字符串转换”（函数→编程→字符串→数值/字符串转换→数值至十进制字符串转换）将采集数据转换为数值字符串。

（2）封装发送信息。调用函数节点“获取日期/时间字符串”（函数→编程→定时→获取日期/时间字符串）、“连接字符串”（函数→编程→字符串→连接字符串），将时间参数和采集数据封装为制表位间隔的字符串，以备 ExpressVI UART 发送。

（3）串口发送。调用 ExpressVI UART（函数→myRIO→UART），将其配置为发送模式，实现上一步封装字符串的发送功能。双击节点，进入配置窗口，设置节点名称为 UART-S，将其设置为 A 口提供的通信端口，设置该节点工作模式为 Write（发送）模式，通信参数采用默认值，最终完成的 UART 通信发送模式配置如图 5-6 所示。

（4）串口接收。再次调用 ExpressVI UART（函数→myRIO→UART），将其配置为接收模式，实现 UART 通信端口的数据接收功能。双击节点，进入配置窗口，设置节点名称 UART-R，将其设置为 A 口提供的通信端口，设置该节点工作模式为 Read（接收）模式，并勾选 Read all available，实现每次读取串口缓冲区全部数据的功能。通信参数采用默认值，最终完成的 UART 通信接收模式配置如图 5-7 所示。

（5）定时控制。为了便于调试和观测程序执行结果，While 循环结构中调用函数节点“等待”（函数→编程→定时→等待），设置等待时长为 1000ms，实现每秒采集数据一次、发生一次、接收一次的功能；同时添加移位寄存器，实现历次接收数据的暂存，并将其与当前接收数据、回车换行符连接进行显示。

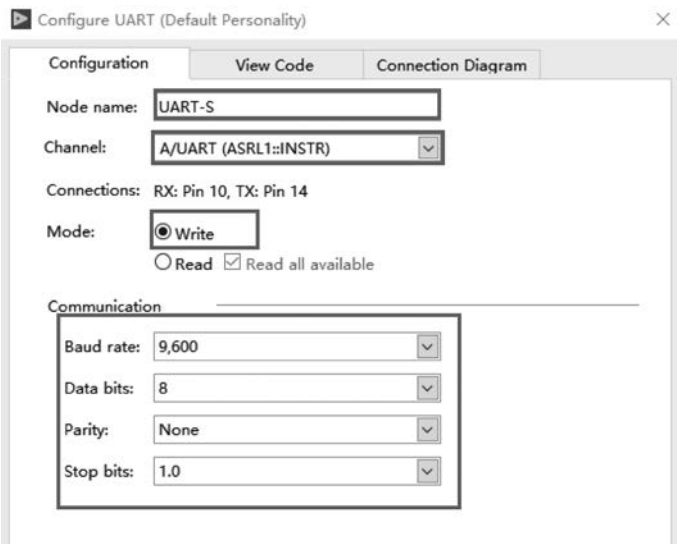

图 5-6　UART 通信发送模式配置

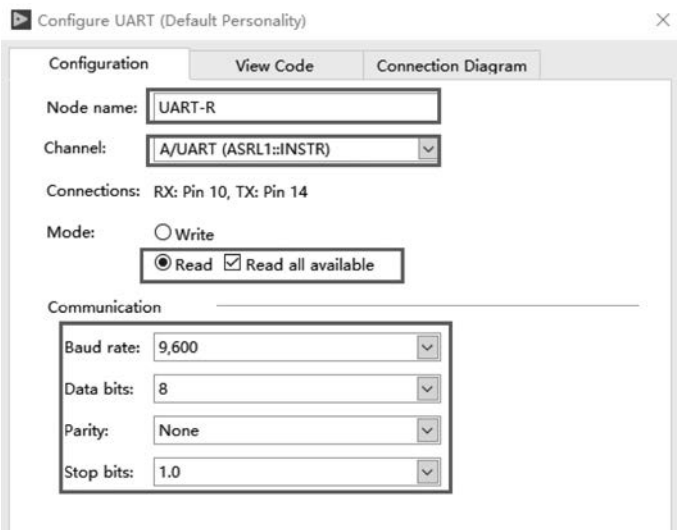

图 5-7　UART 通信接收模式配置

第三帧为后处理帧，调用函数节点 Reset myRIO(函数→myRIO→Device management→Reset)完成 myRIO 开发平台的复位工作，调用函数节点“清除错误”消除程序运行中产生的错误。

完整的 UART 自发自收通信程序实现如图 5-8 所示。

运行程序，UART 自发自收程序执行结果如图 5-9 所示。

值得注意的是，myRIO 并无板载电源维持时钟芯片运行，这会导致当程序部署于 myRIO 实时系统运行时，节点“获取日期/时间字符串”输出的并非当前实际时间。

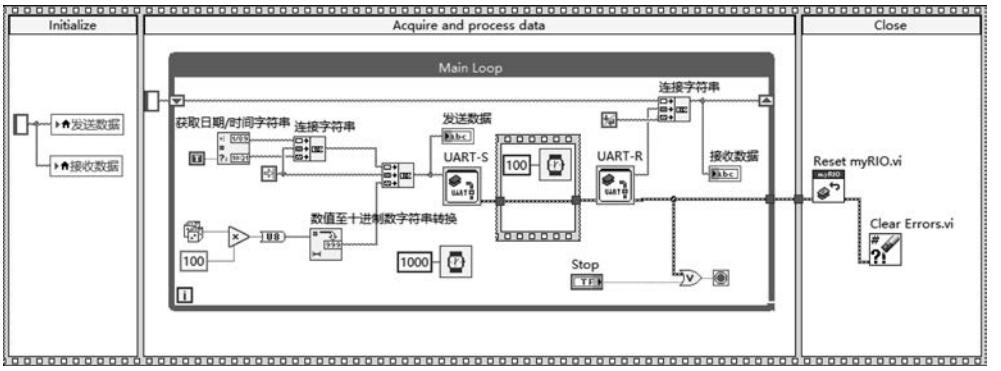

图 5-8 UART 自发自收通信程序实现

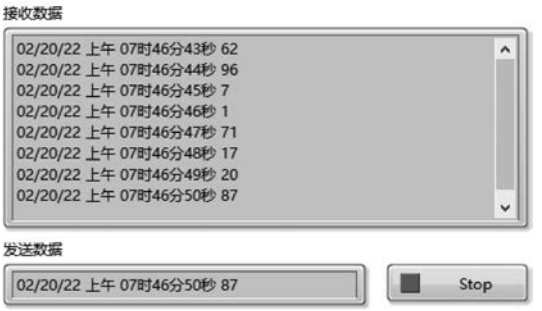

图 5-9 UART 自发自收程序执行结果

需要说明的是，使用 ExpressVI 实现 UART 通信程序设计固然有简单方便的一面，但是该节点由于集成度过高，每次发送或者接收均需要从配置端口到发送接收再到销毁引用完整经历一遍，程序执行效率比较低，因此高效率串行通信程序设计一般采用底层函数编写。

基于底层函数节点实现 UART 接口通信的方法与 Windows 操作系统下串行通信程序设计完全相同。为了提高通信效率，本案例中使用了多线程的方式进行程序编写。其中发送线程实现每秒采集一次数据、获取系统当前时间，将采集数据和当前时间封装为制表位间隔的字符串，通过 A 口 UART 通信端口发出。接收线程中检测当前打开的 UART 通信端口接收缓冲区字节数，如果接收到数据，则读取全部缓冲区中的数据。

本案例中所谓的 UART 通信底层函数节点实际上与 Windows 操作系统下 VI 开发中 VISA 节点完全一致，唯一的区别在于 Windows 操作系统下打开串口的名称一般为“COM＊”，而 myRIO 中 UART 通信端口名称为“ASRL1：INSTR”。

当以多线程方式实现 UART 通信数据发送和数据接收功能时，需要注意多线程的结束方式。本案例中，发送线程的结束依赖于按钮控件“停止”的操作及“VISA 写入”过程中的错误信息进行判断。

当发送线程结束运行后，接收线程中调用属性节点“Serial Settings→End Mode for

Reads”，判断当前 UART 接口读写模式状态，并根据该状态取值判断结束接收循环或者停止接收循环。

多线程方式实现 UART 自发自收通信的完整程序框图如图 5-10 所示。

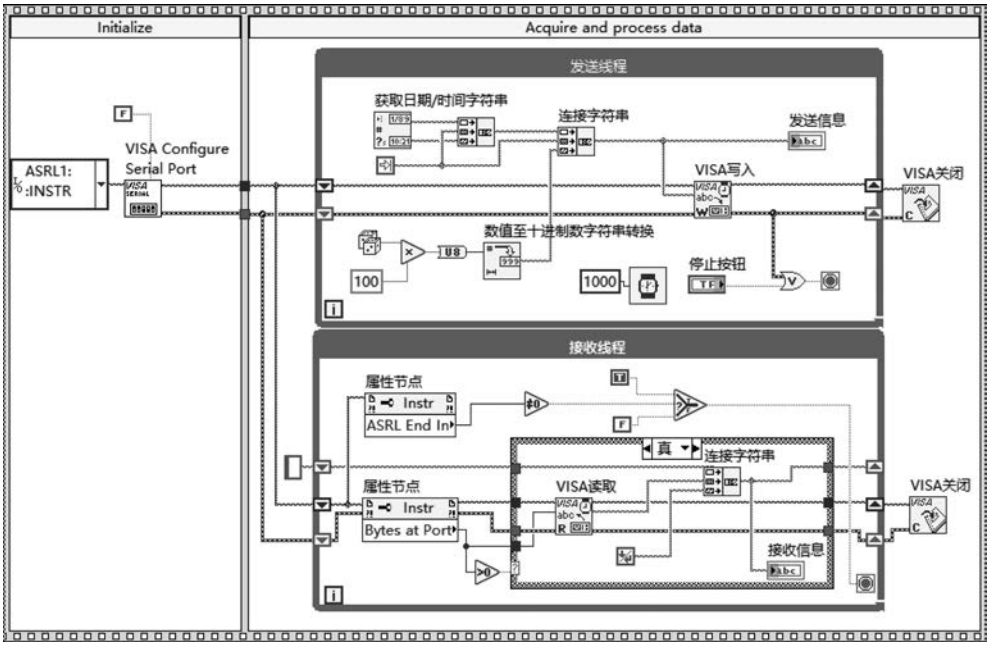

图 5-10　多线程方式实现 UART 自发自收通信的完整程序框图

2. myRIO 连接淘晶驰串口屏


微课视频

作为一种强大的嵌入式系统开发平台，myRIO 提供了无与伦比的嵌入式系统快速开发技术支持。但是美中不足的是，myRIO 标准配置中并未提供显示功能支持。这使得基于 myRIO 开发产生的有关数据在程序独立部署运行时无法直接观测。

嵌入式系统比较常见的显示装置包括数码管(7 段数码管或者级联数码管)、液晶显示屏(1602 显示屏)。这些显示装置要么占用 I/O 引脚数量较多，要么可显示的字符数极其有限，实际应用中存在一定的局限性。

近年来，异军突起的串口屏为开发 myRIO 显示功能提供了一种新的解决方案——串口屏借助强大的内置系统对液晶显示模组进行了封装，并通过 UART 通信端口实现与其他设备之间的交互功能。典型产品为深圳淘晶驰电子有限公司出品的“串口人机交互显示模组”。该产品已经广泛应用于新能源汽车充电桩、智能仓储、仪器仪表、智能家居、工业自动化设备、手持设备、医疗设备、安防设备等不同产品中。

淘晶驰串口屏的显示内容包括文本、数字、图片、动画、视频等，而且还提供按钮、曲线图表等常用人机界面显示对象。更为可贵的是，串口屏一般都提供了触摸屏配置，使其不仅仅可以作为嵌入式系统的显示系统，还可以作为输入装置向嵌入式系统传送信息，可以替代常规的键盘电路。

本案例借助淘晶驰串口屏进一步扩展 myRIO 的应用开发功能。淘晶驰串口屏作为显示装置显示 myRIO 采集的数据，同时串口屏作为用户输入装置，当用户单击串口屏显示界面中的按钮控件时，串口屏向 myRIO 发送相应的控制指令，打开或者关闭 myRIO 板载 LED 灯。

myRIO 中开发基于淘晶驰串口屏的显示系统，首先需要进行串口屏应用程序设计。打开淘晶驰官网，下载安装串口屏集成开发环境 USART HMI。运行 USART HMI，串口屏集成开发环境操作界面如图 5-11 所示。

图 5-11　串口屏集成开发环境操作界面

新建 HMI 工程项目，在弹出的对话框中选择当前使用的设备型号，如图 5-12 所示。

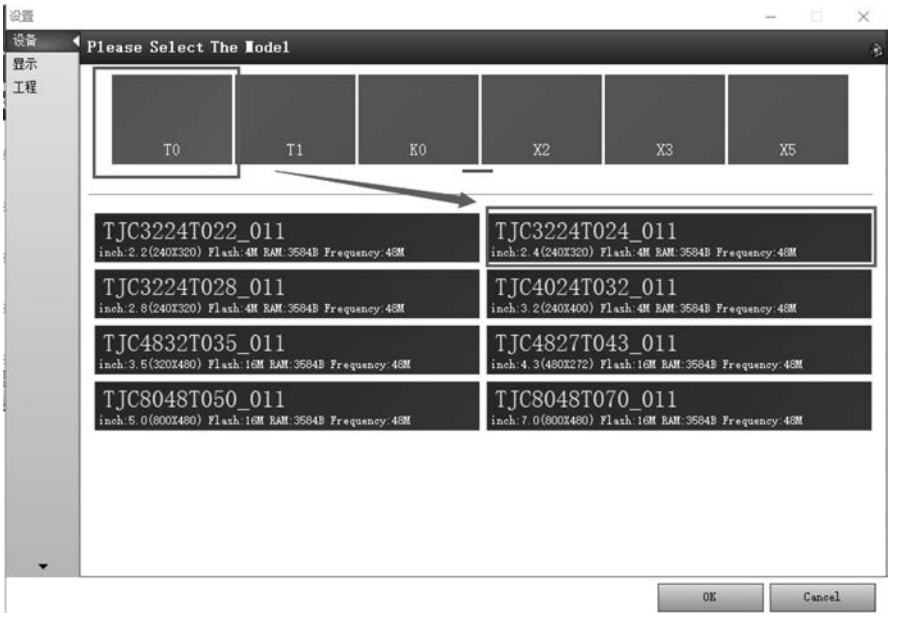

图 5-12　选择当前使用的设备型号

单击 OK 按钮，在弹出的对话框中设置显示方向及字符编码方式（一般显示内容包含中文信息时，多选择 GB2312 编码），如图 5-13 所示。

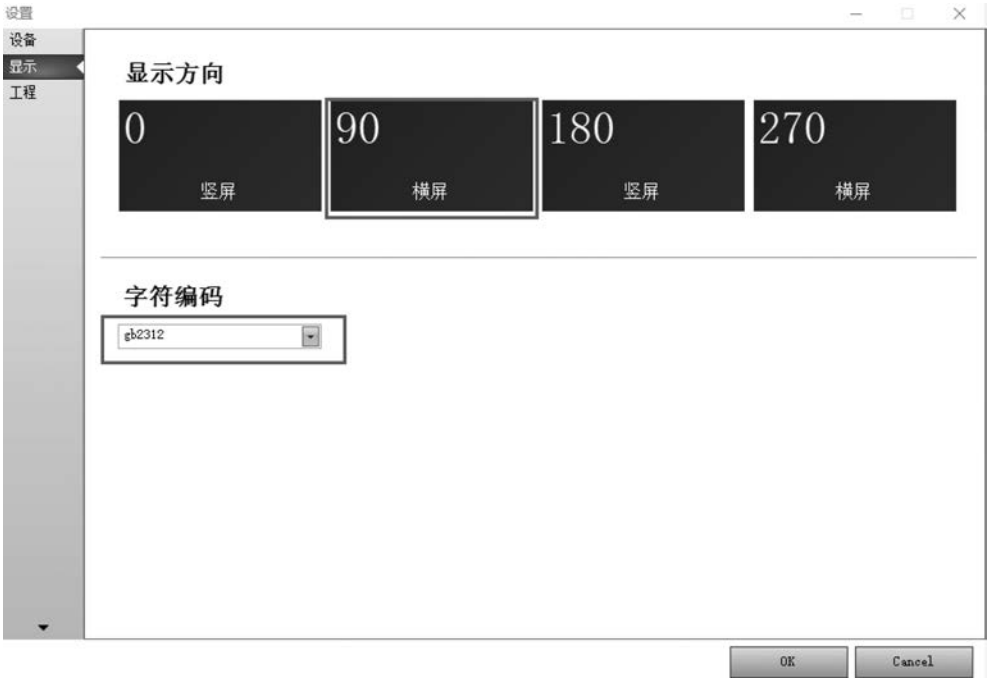

图 5-13　设置显示方向及字符编码方式

单击 OK 按钮，进入如图 5-14 所示的串口屏新建项目操作界面。

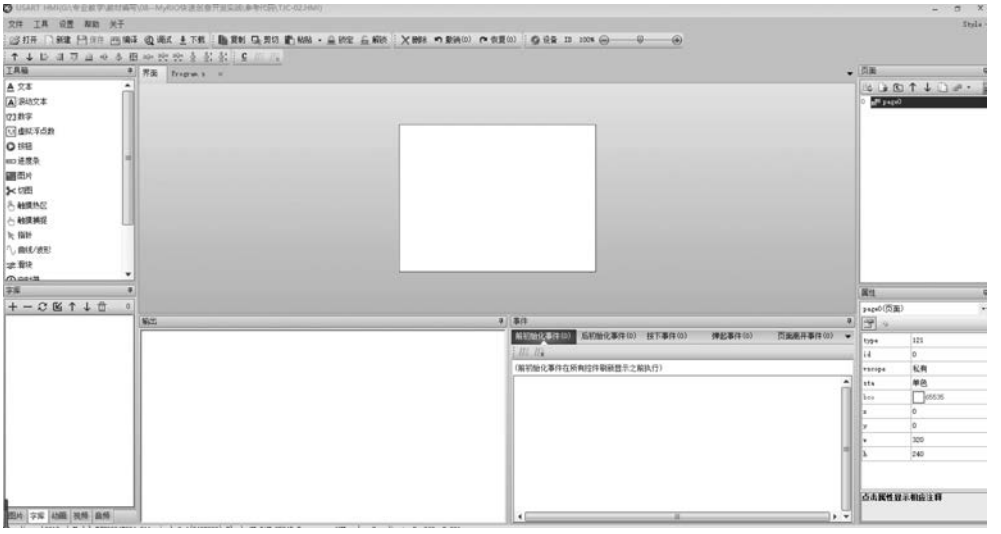

图 5-14　串口屏新建项目操作界面

控件工具箱中选择“文本”“双态按钮”(按下、弹起两种状态)两类控件,拖曳至空白窗口,设置控件属性,完成如图 5-15 所示的串口屏显示控制界面。

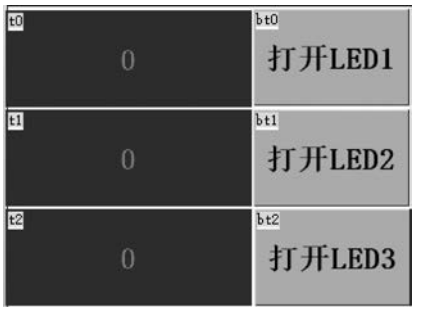

图 5-15 串口屏显示控制界面

文本控件 t0、t1、t2 用来显示来自 myRIO 的采集数据,双态按钮 bt0、bt1、bt2 被单击时,串口屏根据按钮状态值,向 myRIO 发送对应的控制指令,实现 myRIO 板载 LED 的亮灭控制。

需要注意的是,串口屏显示控制界面中无论是按钮还是文本控件,都需要装载已经生成的字库,才能实现有关文本的正常显示。

串口屏作为显示装置时,由 myRIO 发送控制指令,对于有关控件的属性数据进行设定,实现信息显示功能。控制指令由“指令内容”+“结束符”两部分组成。“指令内容”为 ASCII 格式的显示信息设定,通过设置文本控件 txt 属性的方式设定其显示内容。比如 t0 的显示信息设定指令如下:

```
t0.txt = " **** "// *** 为显示内容,本条指令设定文本控件 t0 的显示内容
```

“结束符”为 HEX 格式指令 FF FF FF,表示一条指令的结束。

myRIO 依次完成两部分指令的发送,即可完成文本控件 t0 的显示内容的设定和刷新。

串口屏作为输入装置时,是指对于串口屏显示界面中有关控件的操作事件进行处理。本案例中选择显示界面中双态按钮控件 bt0,在其事件处理代码编写区域键入如下代码:

```
if(bt0.val == 1)//
{
  bt0.txt = "打开 LED1"
  printh FF 01 01 EE//FF 命令头,01 表示 LED1,01 表示打开,EE 命令尾
}else//
{
  bt0.txt = "关闭 LED1"
  printh FF 01 00 EE//FF 命令头,01 表示 LED1,00 表示关闭,EE 命令尾
}
```

项目下载至串口屏后,首次单击串口屏显示界面中双态按钮 bt0,按钮显示文本设定为“打开 LED1”,串口屏发出 16 进制指令 FF 01 01 EE。

双态按钮 bt1、bt2 对应的事件处理代码与 bt0 控件相同,区别仅在于按钮操作过程中按钮显示文本及指令中表示按钮编号的字节内容。其中 bt1 按钮按下事件处理代码如图 5-16 所示。

完成串口屏界面设计和有关控件事件处理代码设计后,单击串口屏集成开发环境 USART HMI 工具栏“调试”按钮,进入如图 5-17 所示的串口屏项目调试界面。

在调试界面的指令输入区键入指令:t0.txt="21",单击“执行所有代码”按钮,文本控件 t0 显示内容被设置为 121。单击串口屏显示界面中双态按钮 bt0,其显示内容由“打开 LED1”改变为“关闭 LED1”,模拟器返回数据区域显示内容为 FF 01 01 EE。这一结果与预

图 5-16　bt1 按钮按下事件处理代码

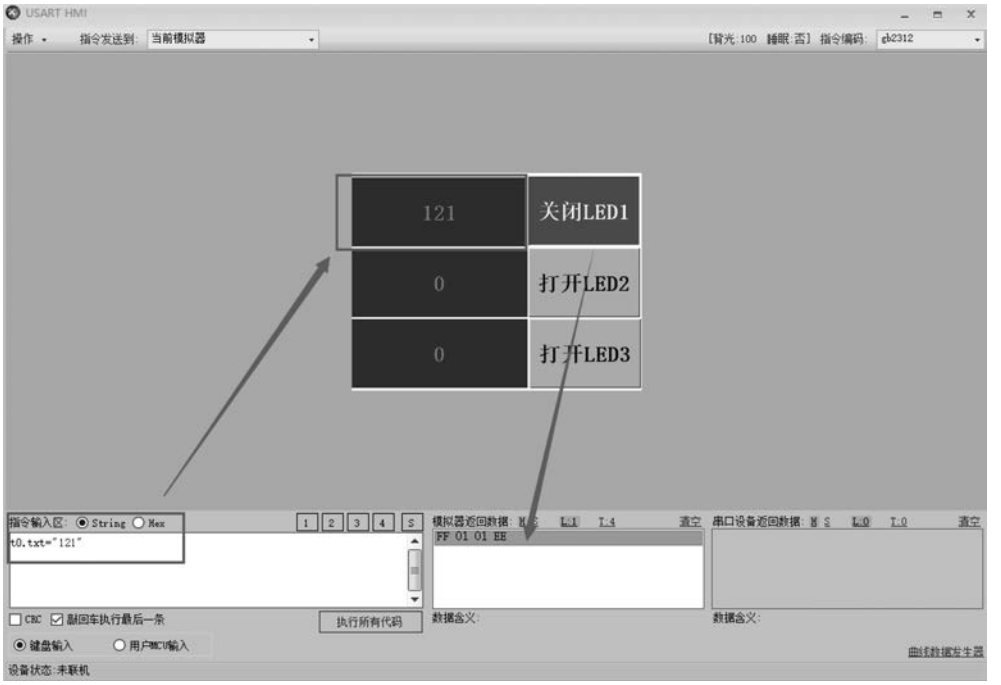

图 5-17　串口屏项目调试界面

期设计目标完全一致，则可以单击工具栏“下载”按钮将测试正确的项目文件下载至串口屏。

完成上述准备工作之后，即可开启 myRIO 连接淘晶驰串口屏的应用程序设计。

1）设计目标

作为验证 myRIO 与串口屏相关功能的应用例程，本案例中 myRIO 每 200ms 采集一次板载三轴加速度传感器数据，将采集数据发送至串口屏中文本控件 t0、t1、t2 显示；同时 myRIO 接收来自串口屏的指令，分析指令内容，控制板载 LED 灯的亮灭。

2）硬件连线

本案例选择淘晶驰 TJC3224T024_011 串口屏，选择 myRIO 开发平台 A 口提供的 UART 接口，myRIO 与串口屏连接引脚对应关系如表 5-1 所示。

表 5-1 myRIO 与串口屏连接引脚对应关系

myRIO(A 口 UART)	串口屏 UART 接口
A 口 Pin1(+5V)	+5V
A 口 Pin10(UART. RX)	TX
A 口 Pin14(UART. TX)	RX
A 口 Pin12(DGND)	GND

3) 设计思路

利用 myRIO 新建项目的程序模板,在 While 循环结构内,实时检测板载三轴加速度计采集结果,将其格式化为控制淘晶驰串口屏的显示指令,通过 UART 端口发送至淘晶驰串口屏;同时实时检测 UART 端口缓冲区数据,当接收到来自淘晶驰串口屏发送的数据时,读取接收数据,并对接收到的数据进行解析,利用解析结果控制 myRIO 板载 LED 显示状态。

程序设计中将生成显示指令封装为子 VI,将解析接收数据控制 LED 封装为子 VI,可简化主程序框图,并增强 LabVIEW 程序的可读性。

4) 程序实现

程序实现可分解为程序总体结构设计、打开 myRIO 串口、板载加速度计数据采集、淘晶驰显示指令生成、发送淘晶驰串口屏显示指令、淘晶驰串口屏返回数据接收、解析淘晶驰返回数据控制板载 LED、myRIO 重置等步骤。

(1) 程序总体结构设计。利用 myRIO 项目模板自动生成的三帧程序结构。第一帧中配置并打开串口;第二帧中采集三轴加速度计数据,生成发送至淘晶驰串口屏的指令,同时接收淘晶驰串口屏返回数据,根据数据解析结果控制 myRIO 板载 LED 显示;第三帧中进行设备重置操作。

(2) 打开 myRIO 串口。程序框图初始化帧中,调用函数节点“VISA 配置串口”(函数→myRIO→Low Level→UART),设置输入参数“VISA 资源名称”为 myRIO 开发平台 A 口对应的 UART 端口。

(3) 板载加速度计数据采集。保持新建的 myRIO 项目自动生成的三轴加速度数据采集程序不变,作为后续发送至淘晶驰串口屏的数据来源。

(4) 淘晶驰显示指令生成。板载加速度计采集的 3 个数据作为淘晶驰串口屏的显示内容,需要将其封装为淘晶驰显示指令。为了增强程序可读性,该功能进行模块化设计,将其设计为子 VI。子 VI 有关设计信息如下。

输入参数 1:浮点类型数据 d0。

输入参数 2:浮点类型数据 d1。

输入参数 3:浮点类型数据 d2。

输出参数:字符串(通过 UART 端口发送的数据)。

子 VI 文件名称:CreateSendData. vi。

按照如下步骤完成子 VI 设计。

子 VI 调用函数节点“格式化字符串”(函数→编程→字符串→格式化字符串)、节点“字符串至字节数组转换”(函数→编程→字符串→路径/数组/字符串转换→字符串至字节数组转换)及节点“数组插入”(函数→编程→数组→数组插入),将传入的浮点数转换为串口屏刷新文本控件显示内容的指令格式。

调用函数节点“数组插入”(函数→编程→数组→数组插入)将采集数据(子 VI 输入参数,数值类型)对应的串口屏更新文本控件显示内容指令进行合并。

调用函数节点“字节数组至字符串转换”(函数→编程→字符串→路径/数组/字符串转换→字节数组至字符串转换),将合并指令转换为串口可发送的字符串类型数据(子 VI 输出参数)。

对应的 myRIO 发送串口屏指令生成子 VI 程序实现如图 5-18 所示。

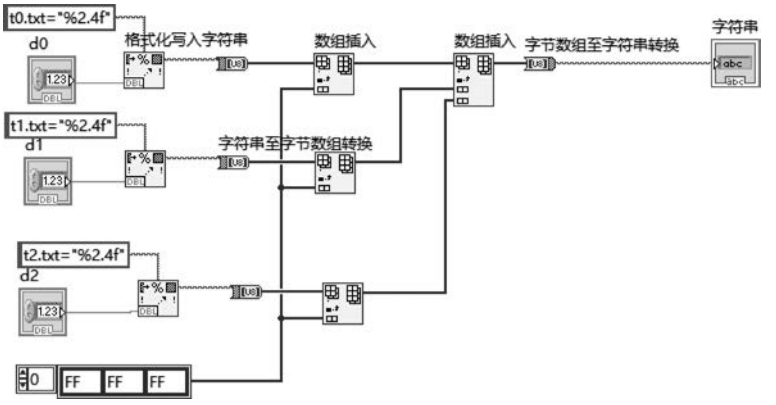

图 5-18　myRIO 发送串口屏指令生成子 VI 程序实现

(5)发送淘晶驰串口屏显示指令。调用函数节点“VISA 写入”(函数→myRIO→Low Level→UART→VISA 写入),实现 myRIO 采集数据的连续发送功能,其中发送内容由 Step4 中自定义 VI 生成。

(6) 淘晶驰串口屏返回数据接收。调用属性节点“VISA 串口字节数”(函数→myRIO→Low Level→UART→VISA 串口字节数),实现 myRIO 串口缓冲区读取字节数的实时监测。当串口接收到串口屏返回的数据时,调用函数节点“VISA 读取”(函数→myRIO→Low Level→UART→VISA 读取)接收全部缓冲区数据。由于串口屏实际返回数据为字节数组类型,而 LabVIEW 串行通信接收的数据全部为字符串类型,调用函数节点“字符串至字节数组转换”(函数→编程→字符串→路径/数组/字符串转换),将串口接收的字符串类型数据转换为字节数组。

(7) 解析淘晶驰返回数据控制板载 LED。将(6)中接收字节数组内容控制板载 LED 亮灭功能封装为子 VI,子 VI 有关设计信息如下。

输入参数:字节数组。

子 VI 文件名称:ReceivDisp. vi。

按照如下步骤完成子 VI 设计。

调用函数节点“索引数组”(函数→编程→数组→索引数组),设定 2 个索引值分别为 1 和 2,提取字节数组中第 2 个和第 3 个数据元素。

添加条件结构,其分支选择器连接字节数组中第 2 个数据元素,用以判断驱动哪个 LED。

条件结构内,添加选择结构,判断字节数组中第 3 个数据元素是否等于 0,并根据判断结果驱动回应的 LED。

当字节数组第 2 个数据元素值为 1 时,对应的 myRIO 接收指令解析程序实现如图 5-19 所示。

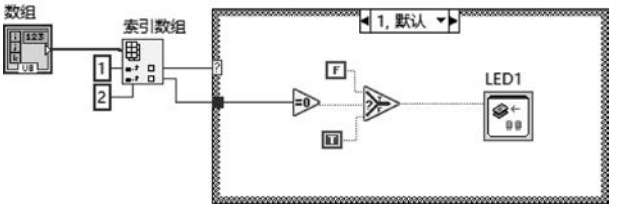

图 5-19 myRIO 接收指令解析程序实现

当串口接收字节数组的第 2 字节取值为 2、3 时,可参照修改程序自行实现。

调用 ExpressVI LED 实现基于接收指令解析结果控制板载 LED 的功能。双击 ExpressVI LED 节点图标,配置当前程序控制对象为 LED0,如图 5-20 所示。

图 5-20 配置当前程序控制对象为 LED0

(8) myRIO 设备重置。在程序第三帧中调用函数节点 Reset myRIO(函数→myRIO→Device management→Reset)完成 myRIO 开发平台的复位工作。

完整的 myRIO 与串口屏交互程序如图 5-21 所示。

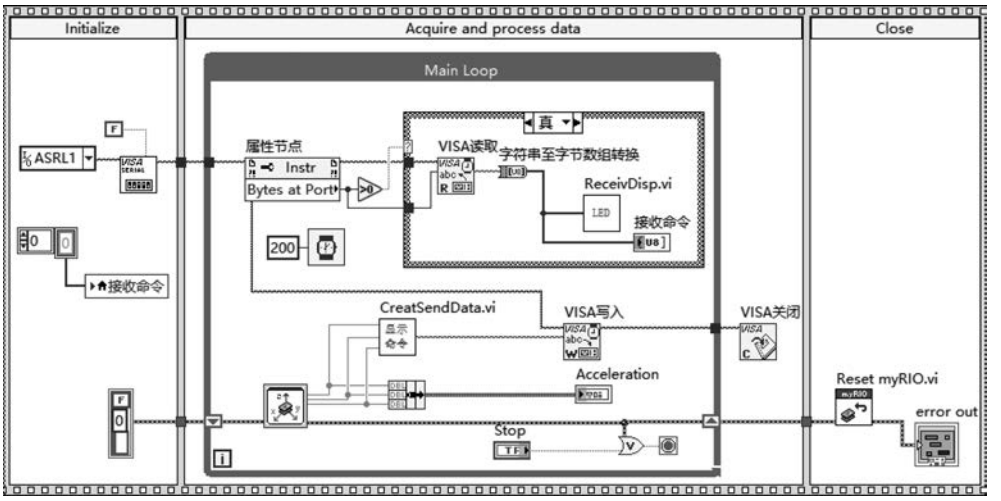

图 5-21　myRIO 与串口屏交互程序

运行程序，myRIO 与串口屏交互程序执行结果如图 5-22 所示。

图 5-22　myRIO 与串口屏交互程序执行结果

由图 5-22 可见，板载三轴加速度传感器测量数据得以实时显示，而且用户单击淘晶驰串口屏中双态按钮，可以控制对应的 myRIO 板载 LED 亮灭。实践结果证明了串口屏既可以作为输出显示装置，亦可作为数据输入装置，极大地方便了 myRIO 有关功能的扩展。

3. myRIO 与蓝牙设备之间的通信


微课视频

如前所述，串口屏可以极其简单方便地实现 myRIO 的显示功能，但是总有一些应用场景下串口屏和 myRIO 之间无法直接连线，导致 myRIO 相关应用系统依旧存在显示功能需求无法解决的问题。此时可以借助具有透传功能的 UART 接口无线通信电路模块，将直接连线的串口屏转换为无线串口屏，进一步扩展 myRIO 的显示功能。

具有透传功能的 UART 接口无线通信电路模块可选对象比较多，比如蓝牙、WiFi、ZIGBEE、LORA 等。这里选择蓝牙模块实现 myRIO 和串口屏之间的无线连接，从而实现 myRIO 蓝牙无线串口屏。这一显示功能拓展的 myRIO 实现分为三个阶段——蓝牙模块配对、串口屏开发、myRIO 程序设计。

首先进行蓝牙模块的配对。使用一对 HC05 蓝牙模块，在 AT 指令模式下设置其中一个为主模式，另一个为从模式（HC05 为主从一体，出厂默认为从模式）。重新上电后，HC05 自动配对、自动连接。

蓝牙模块 HC05 配置为主模式工作状态常用的 AT 指令如表 5-2 所示。

表 5-2 HC05 配置为主模式工作状态常用的 AT 指令

步骤	发送命令	返回结果	结果含义
1	AT+\r\n	OK	设备正常,可以进行下一步操作
2	AT+ORGL\r\n	OK	设备已经恢复出厂设置
3	AT+VERSION?\r\n	VERSION:3.0-20170601	返回设备版本信息
4	AT+PSWD?\r\n	+PIN:"1234" OK	返回蓝牙模块默认连接密码
5	AT+PSWD="1010"\r\n	OK	设置新连接密码成功,可再次调用 AT+PSWD?\r\n 指令查看连接密码
6	AT+ROLE=1\r\n	OK	设置该模块为主模式工作(0 为从模式)
7	AT+ROLE?\r\n	+ROLE:1 OK	查询模块工作模式,1 表示为主模式
8	AT+UART?\r\n	+UART:9600,0,0 OK	查询模块波特率,当前设置为 9600
9	AT+CMODE=0\r\n	OK	设置模块连接模式为固定地址连接模式
10	AT+BIND=98d3,33,813aca\r\n	OK	设置绑定地址(该地址需要第二块蓝牙模块通过 AT 指令查询获得,查询指令 AT+ADDR?\r\n),第一个逗号前地址不足 4 位时,需在前补 0 凑足 4 位
11	AT+BIND?	+BIND:18,E4,400006 OK	查询当前绑定的蓝牙地址

串口调试助手中完成上述操作之后,相应的蓝牙模块被设置成为主模式,并且绑定地址为 98d3,33,813aca 的蓝牙模块,通信速率为 9600b/s。

进一步地,蓝牙模块配置为从模式工作状态常用的 AT 指令如表 5-3 所示。

表 5-3 HC05 配置为从模式工作状态常用的 AT 指令

步骤	发送命令	返回结果	结果含义
1	AT+\r\n	OK	设备正常,可以进行下一步操作
2	AT+ORGL\r\n	OK	设备已经恢复出厂设置
3	AT+VERSION?\r\n	VERSION:3.0-20170601	返回设备版本信息
4	AT+PSWD?\r\n	+PIN:"1234" OK	返回蓝牙模块默认连接密码
5	AT+PSWD="1010"\r\n	OK	设置新连接密码成功,可再次调用 AT+PSWD?\r\n 指令查看连接密码,主模式和从模式的密码需要保持一致
6	AT+ROLE=0\r\n	OK	设置该模块为从模式工作(0 为从模式)
7	AT+ROLE=?\r\n	+ROLE:0 OK	查询模块工作模式,0 表示为从模式
8	AT+UART=9600,0,0\r\n	OK	设置模块波特率,当前设置为 9600
9	AT+ADDR?\r\n	+ADDR:=98d3,33,813ac OK	查询模块物理地址(该地址主模块绑定时需要)

重新上电后进入常规工作模式，等待 1～2s，指示灯两闪一停，表示自动完成配对。两个蓝牙模块一旦上电后自动配对工作完成，就可以完全替代原来有线连接的串行端口，变身为无线串口，使用极为方便。

然后进行串口屏开发。打开串口集成开发环境 USART HMI，创建蓝牙串口屏项目显示界面如图 5-23 所示。

显示界面由“曲线/波形”控件及 3 个“虚拟浮点数”控件构成。“曲线/波形”控件用于显示 myRIO 板载三轴加速度传感器测量数值的趋势曲线，3 个“虚拟浮点数”控件分别显示三轴加速度传感器每个轴向测量数值。

在屏幕前初始化事件处理中，清空控件“曲线/波形”显示内容，并对 3 个“虚拟浮点数”控件进行初始化操作，对应的串口屏前初始化事件相关代码如图 5-24 所示。

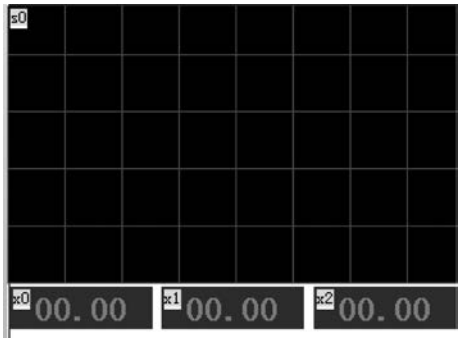

图 5-23　蓝牙串口屏项目显示界面

图 5-24　串口屏前初始化事件相关代码

由于控件“曲线/波形”需要显示 3 路数据波形，因此设置其属性 ch 值为 3，表示 3 个波形显示的数据通道。其他属性亦可根据需要进行修改，如 3 条曲线的颜色属性：pco0、pco1、pco2 的设置，控件“曲线/波形”主要属性配置结果如图 5-25 所示。

参考淘晶驰公司提供的 USART HMI 集成开发环境中各类控件使用说明，控件“曲线/波形”用于在串口屏上显示波形或者曲线。该控件涉及指令 add、cle、addt。cle 指令用于清除曲线控件中的数据。串口屏执行指令“cle 1,0 ”，可清除 ID 为 1 的曲线控件的 0 通道数据，执行指令“cle 1,255”，可清除 ID 为 1 的曲线控件的所有通道数据(参数 255 意为全部通道)。

add 指令用于向曲线控件逐点添加显示数据，串口屏执行指令“add 1,0,30”，实现 ID 为 1 的曲线控件通道 0 添加新的显示数据 30。该指令第一个参数 1 表示曲线控件的 ID，如有多个曲线控件同屏显示，则该参数区别不同的曲线控件(每个 page 页面最多支持 4 个曲线控件)。第 2 个参数 0 表示当前曲线控件的数据通道 0(每个曲线控件最多支持 4 个通道，可以连续发送数据，控件会自动平推显示数据)，第 3 个参数 30 表示曲线控件中需要新显示的数据是 30。需要说明的是，控件“曲线/波形”接收 0～255 的单字节整数，实际采集的数据一般需要进行数据类型转换和区间映射才能正确显示。

控件“虚拟浮点数”用于在串口屏上显示浮点数(只是显示效果是浮点数，本质是整数)。该控件浮点数显示参数设置如图 5-26 所示。

图 5-25 控件“曲线/波形”主要属性配置结果

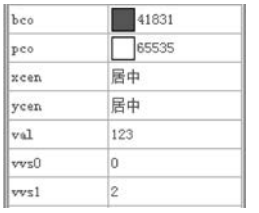

图 5-26 浮点数显示参数设置

其中属性 val 指定控件显示的数值，vvs1 指定显示数据的小数点位数。本案例中 val 值设置为 123，因为 vvs1 属性(小数位数)设置为 2，所以控件实际显示的数据为 1.23。

这类控件显示的数据需要刷新时，需要程序对于实际的浮点数进行格式化操作，将其转换为控件指定小数位数的浮点数，然后放大 100 倍取整，作为控件 val 取值。例如，程序中拟显示的浮点数取值为 1.2312，则首先将其格式化为小数点后取 2 位，得到浮点数 1.23，然后放大 100 倍取整，得到整数 123。此时程序向串口屏发送指令 x0.val=123，设置控件的 val 属性值为 123。串口屏接收该数据后，按照控件设置的 2 位小数位数规则，设置屏幕最终显示数据为 1.23。

完成上述准备工作后，即可开展基于蓝牙串口屏的 myRIO 应用程序设计相关工作。

1) 设计目标

根据上述淘晶驰串口屏使用方法，编写 myRIO 应用程序采集板载三轴加速度传感器数值并显示，将采集的数据封装为刷新串口屏的显示指令，通过 UART 通信端口发出，用以驱动串口屏分别以趋势曲线、实时数值 2 种方式分别显示 3 个轴向的加速度测量值。由于 UART 端口连接已经配对连接的蓝牙模块，因此显示指令可借助蓝牙模块无线传输至串口屏，从而实现 myRIO 采集数据的蓝牙无线串口屏显示功能。

2) 硬件连线

完成配对连接的蓝牙模块使用方法与计算机自身配置的串口使用方法完全一致，只要会编写串行通信程序，就能借助蓝牙模块实现数据的短距离无线传输。分别将 2 个 HC05 蓝牙模块与 myRIO 和淘晶驰串口屏连接，对应的蓝牙无线串口屏硬件连接如图 5-27 所示。

myRIO 与蓝牙模块连接引脚对应关系如表 5-4 所示。

表 5-4 myRIO 与蓝牙模块连接引脚对应关系

myRIO(A 口 UART)	蓝牙模块 HC05
A 口 Pin1(+5V)	VCC
A 口 Pin10(UART.RX)	TXD
A 口 Pin14(UART.TX)	RXD
A 口 Pin12(DGND)	GND

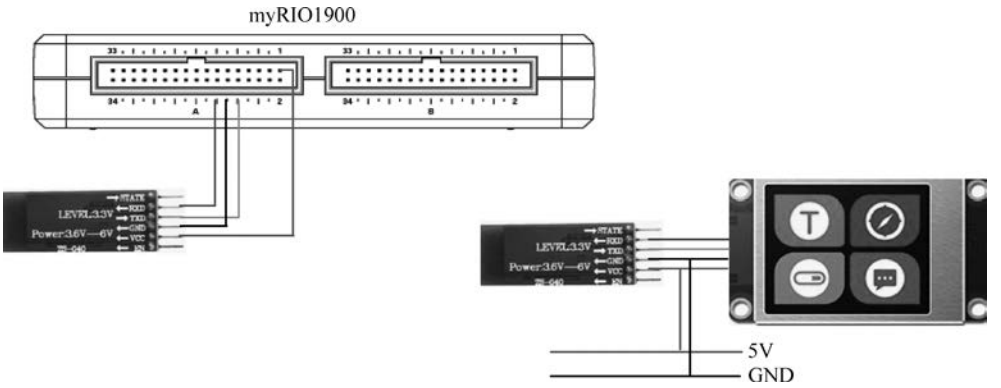

图 5-27　蓝牙无线串口屏硬件连接

串口屏与蓝牙模块连接引脚对应关系如表 5-5 所示。

表 5-5　串口屏与蓝牙模块连接引脚对应关系

蓝牙模块 HC05	串口屏 UART 接口
VCC	VCC
TXD	RXD
RXD	TXD
GND	GND

注：串口屏、蓝牙模块 HC05 均需要独立供电，表 5-4 中 VCC、GND 引脚的连接关系，表示 2 个引脚并联至对应的电源线。

3）设计思路

利用 myRIO 新建项目中自动生成的程序模板，保留其三帧程序基本结构。第一帧为初始化帧，打开并配置连接蓝牙模块的 UART 通信端口；第二帧为主程序帧，在该帧 While 循环结构内，实时检测板载三轴加速度计数据，并将采集的数据进行区间线性映射，转换为淘晶驰串口屏趋势曲线显示数据，并进一步将映射后的数据封装为驱动淘晶驰串口屏对应的显示指令，通过 UART 通信端口发送，进而实现基于蓝牙通信技术无线串口屏显示 myRIO 采集数据功能；第三帧为程序后处理帧，实现 myRIO 程序运行结束后的设备重置工作。

4）程序实现

程序实现可分解为初始化处理、主程序结构设计、串口屏显示指令生成与封装、串口屏显示指令发送、myRIO 设备重置等步骤。

（1）初始化处理。初始化帧中，调用函数节点“VISA 配置串口”（函数→myRIO→Low Level→UART→VISA 配置串口），设置其参数“VISA 资源名称”为 A 口对应的 UART 通信端口，设置其参数“启用终止符”为逻辑假（默认为逻辑真），其他通信参数均可采用默认值。

（2）主程序结构设计。主程序帧中，While 循环结构中实时采集板载三轴加速度计数

据，将采集的数据分别封装为用于淘晶驰串口屏曲线显示指令、数值显示指令，并将两条指令合并为一条发送至淘晶驰串口屏的命令，调用函数节点“VISA 写入”（函数→myRIO→Low Level→UART→VISA 写入），完成串口数据发送。

（3）串口屏显示指令生成与封装。串口屏显示指令生成与封装分为“实时数据显示指令生成与封装”和“趋势曲线显示指令生成与封装”两部分完成。

串口屏显示指令生成与封装子 VI 设计信息如下。

输入参数 1：浮点类型数据 d1。

输入参数 2：浮点类型数据 d2。

输入参数 3：浮点类型数据 d3。

输出参数：字符串（通过 UART 端口发送的数据）。

子 VI 文件名称：CreatData. vi。

按照如下步骤完成子 VI 设计。

“实时数据显示指令生成与封装”设计时，首先将 3 个采集数据乘 100，再调用函数节点“格式化字符串”（函数→编程→字符串→格式化字符串），将其转换为刷新 3 个“虚拟浮点数”控件的指令，调用函数节点“字符串至字节数组转换”（函数→编程→字符串→路径\数组\字符串转换→字符串至字节数组转换），将 ASCII 字符串形式的“虚拟浮点数”控件刷新指令转换为字节数组，并调用函数节点“数组插入”（函数→编程→数组→数组插入），在指令字节数组后插入 3 字节 FF FF FF，形成一条完整显示指令。进一步调用函数节点“字节数组至字符串转换”（函数→编程→字符串→路径\数组\字符串转换→字节数组至字符串转换），将完整的指令转换为串行通信可发送的字符串数据类型，从而完成一个“虚拟浮点数”控件数据刷新显示指令的生成。

类似方法完成剩余 2 个“虚拟浮点数”控件数据刷新指令的生成，然后调用函数节点“连接字符串”（函数→编程→字符串→连接字符串），将 3 条“虚拟浮点数”控件的数据刷新指令进行组合。

“趋势曲线指令生成与封装”设计时，由于“曲线/波形”控件只能接收 0～255 的单字节整数，而采集的加速度数值最小为 0，最大一般不超过 5g，直接显示采集数据，必然导致数据波形位于控件底部，不宜观测。因此采取区间映射的方式，将实际采集的 0～5 数据映射至 20～200，而且大于 200 的数据钳位至 200。可按照如下公式进行采集数据与显示数据之间的映射。

$$y=\frac{200-20}{5-0}x+20=36x+20$$

调用函数节点“格式化字符串”（函数→编程→字符串→格式化字符串），将 3 个采集的数据分别格式化为“曲线/波形”控件 3 个数据通道刷新显示的指令，再将 ASCII 指令转换为字节数组，最后添加后缀 FF FF FF，然后转换为串口发送需要的字符串类型。

将上述两部分指令再次调用函数节点“连接字符串”，形成完整的显示驱动指令，对应的串口屏显示驱动子 VI 程序如图 5-28 所示。

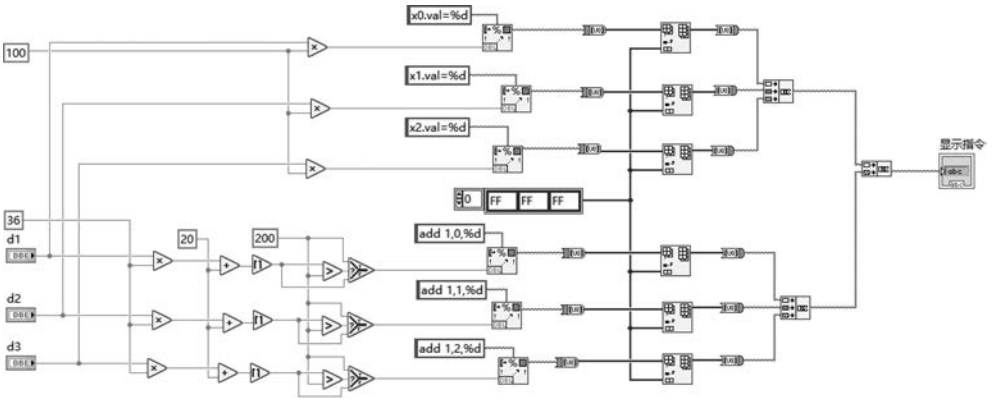

图 5-28　串口屏显示驱动子 VI 程序

(4)串口屏显示指令发送。调用函数节点“VISA 写入”(函数→myRIO→Low Level→UART→VISA 写入),发送封装好的串口屏显示驱动指令,实现 myRIO 对于串口屏显示内容的控制。

主程序帧中,While 循环结束后调用函数节点“VISA 关闭”(函数→myRIO→Low Level→UART→VISA 关闭),释放通信程序所占用的资源。

(5) myRIO 设备重置。第三帧中,调用函数节点“Reset myRIO”(函数→myRIO→Device management→Reset),实现程序结束后设备的复位操作。完整的蓝牙无线串口屏显示程序如图 5-29 所示。

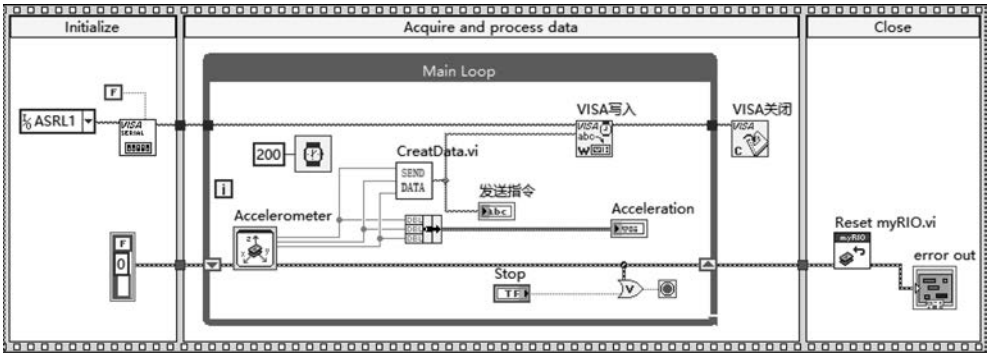

图 5-29　蓝牙无线串口屏显示程序

运行程序,蓝牙无线串口屏显示程序界面如图 5-30 所示。

由图 5-30 可见 myRIO 发出的串口屏控制指令(指令后缀 FF FF FF 不可见)。指令中有关数据完全符合放大取整及区间映射预期。进一步地,串口屏中可观测到虚拟浮点数控件实际显示的数值与 myRIO 测量结果完全一致,“曲线/波形”控件显示波形也处于 20～200 的数值区间(实际上只具备了波形趋势观测的意义),串口屏显示效果如图 5-31 所示。

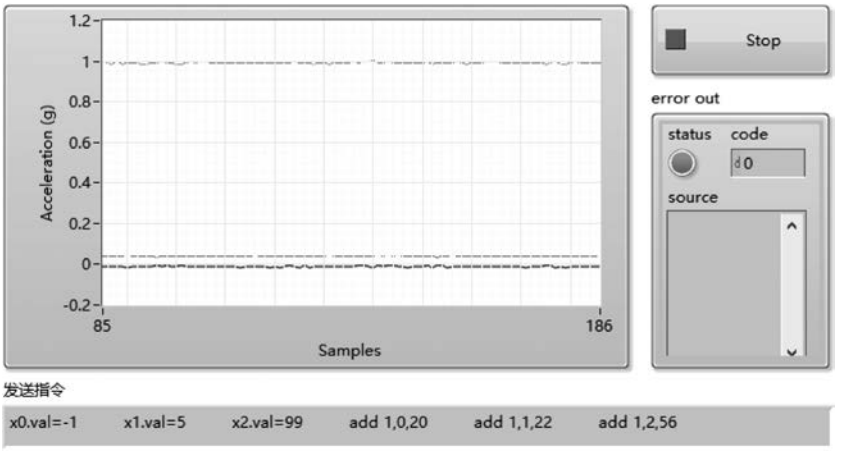

图 5-30　蓝牙无线串口屏显示程序界面

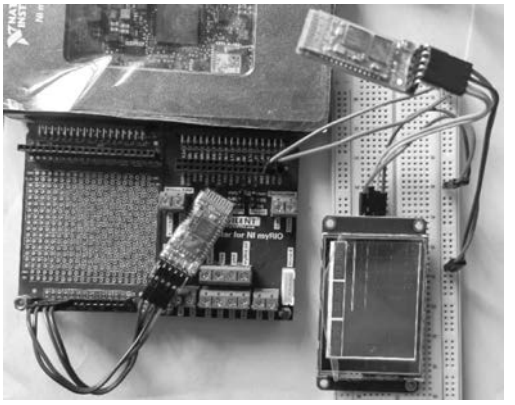

图 5-31　串口屏显示效果

5.2　WiFi 通信技术及应用

myRIO 内置的 WiFi 模块使其具备了强大的系统级通信能力，借助内置的 WiFi 模块，myRIO 可以轻而易举地创建无线局域通信网络或者接入无线因特网，进而实现 myRIO 与其他应用系统之间无线通信，是扩展 myRIO 应用系统技术规模的重要途径，尤其对于物联网相关的应用开发具有重要的意义。本节主要介绍 WiFi 通信技术基本概念，myRIO 中基于 WiFi 通信的 TCP、UDP、HTTP 通信程序设计相关 VI，并针对 myRIO 开发时 WiFi 通信技术的应用场景，将 WiFi 通信技术应用分为无线局域网、物联网 2 类应用模式，并通过无线局域网应用场景下 TCP 客户端应用、TCP 服务器应用、UDP 广播通信等实例及物联网应用场景下基于 TCP、UDP、HTTP、MQTT 协议的应用实例介绍基于 WiFi 连接的通信程序设计的基本方法。

5.2.1　WiFi 通信技术概述

WiFi(Wireless Fidelity)技术是一种典型的短距离无线连网的技术，最初主要用于替代传统网线，快速部署无线局域网。常见的连网方式是通过一个无线路由器构建热点，热点发射信号覆盖的有效范围都可以采用 WiFi 的连接方式接入热点，形成无线局域网，进行相关设备之间的数据通信。

WiFi 技术遵循 IEEE 所制定的 802.11x 系列标准，主要由 802.11a、802.11b、和 802.11g 3 个标准组成。WiFi 通信技术采用 2.4GHz 频段，链路层采用以太网协议为核心，以实现百米范围内多设备之间的信息传输。WiFi 通信技术传输速度较高，达到 11Mbps，有效传输距离也满足绝大部分场景下的无线通信需求，因而受到厂商和用户的青睐，占据着无线传输的主流地位。WiFi 无线网络的基本组成为 AP 热点＋无线网卡，也可配合现有的有线架构分享网络资源实现更大范围的数据通信。

myRIO 中搭载了 WiFi 模块，可以灵活设置为“接入无线网络”“创建无线网络”两种方式中的任何一种。

“接入无线网络”方式多用于实现 myRIO 终端接入无线互联网或无线局域网的功能。“创建无线网络”方式多用于实现 myRIO 终端作为 WiFi 热点，与其他电子设备构建无线局域网的功能。

5.2.2　主要函数节点

一旦建立基于 WiFi 的无线通信网络，其通信程序的设计方法与常规的以太网通信并无不同，可以直接调用 LabVIEW 中提供的常用网络通信协议相关函数节点，包括 TCP、UDP、HTTP 等函数选板中提供的全部函数节点。

LabVIEW 中有关 TCP 通信程序编写的函数节点如图 5-32 所示。

图 5-32　TCP 通信相关函数节点

LabVIEW 中有关 UDP 通信程序编写的函数节点如图 5-33 所示。

LabVIEW 中有关 HTTP 通信程序编写的函数节点如图 5-34 所示。

需要指出的是，myRIO 中的 WiFi 通信技术实现的前提条件是 myRIO 中的 WiFi 模块得到正确的配置。

图 5-33　UDP 通信相关函数节点

图 5-34　HTTP 通信相关函数节点

5.2.3　基于 WiFi 的局域网通信应用实例

myRIO 实现 WiFi 环境下的局域网络通信，一般主要用于 myRIO 和其他电子系统之间的通信。使用 WiFi 通信技术，需要对 myRIO 内置的 WiFi 模块进行配置，配置方法可参见 2.4 节相关内容。myRIO 中应用 WiFi 通信技术，分为“创建无线网络”“连接无线网络”两种应用模式。

“创建无线网络”模式下，其他电子设备接入 myRIO 创建的 AP 热点，形成无线局域网，实现 myRIO 和其他电子设备之间的无线通信功能。

“连接无线网络”模式下，myRIO 与其他电子设备接入同一无线网络，形成无线局域网或者连接无线互联网，实现无线通信功能。


微课视频

1. myRIO 作为 TCP 客户端的无线局域网数据通信

本案例中将 myRIO 内置的 WiFi 模块配置为“接入无线网络”，与计算机或者手机接入同一个无线网络。无线局域网中 myRIO 作为 TCP 客户端，计算机或者手机作为 TCP 服务器。作为 TCP 客户端，myRIO 接入无线网络后分配的 IP 地址可通过 MAX 软件查看（本案例中 myRIO 开发平台分配的 IPv4 地址为 192.168.0.107）。

1）背景知识

TCP 客户端应用程序开发，其基本结构由“打开 TCP 连接”“写入 TCP 数据”和“关闭 TCP 连接”三个函数节点组成。

（1）函数节点“打开 TCP 连接”主要用于建立与服务器端的 TCP 连接，需要设置服务

器端 IP 地址及服务器监听的端口。

（2）函数节点“写入 TCP 数据”用于向服务器端发送数据。需要说明的是，LabVIEW 中，通信数据都必须转换为字符串才能发送。

（3）函数节点“关闭 TCP 连接”用于释放 TCP 通信所占用的资源。

如果客户端需要连续向服务器端发送数据，一般循环前调用函数节点“打开 TCP 连接”；循环结构中调用函数节点“写入 TCP 数据”。循环结束后调用函数节点“关闭 TCP 连接”。这一程序结构设计避免了连续发送中频繁建立连接、释放资源，有助于提高程序运行效率。

2）设计目标

myRIO 与计算机接入同一 WiFi 热点，构建无线局域网。myRIO 作为客户端，计算机作为服务器。myRIO 向服务器发起连接请求，建立连接后，myRIO 采集两路数据，并将采集的两路数据转变为以“,”间隔的字符串，发送至服务器。服务器端记录 myRIO 采集的数据以备后续使用。

篇幅所限，这里使用网络调试助手软件工具模拟服务器端应用程序。打开 Windows 操作系统控制面板，选择“网络和 Internet→WLAN→硬件属性”，可以查看计算机分配获得 IPv4 地址：192.168.0.102。该地址作为 TCP 通信服务器 IP 地址。

TCP 通信首先运行服务器端应用程序。计算机端打开网络调试助手，按照图 5-35 所示方式将网络调试助手配置为服务器模式。

图 5-35　配置网络调试助手为 TCP Server 模式

3）设计思路

利用 myRIO 新建项目自动生成的程序模板，保留其三帧程序基本结构。第一帧为初始化帧，主要用于有关数据、控件的初始赋值，建立 myRIO 与服务器之间的 TCP 连接；第二帧为主程序帧，完成应用程序的核心业务功能的开发，程序基于 While 循环结构实现数据采集、TCP 通信报文封装、TCP 数据发送等功能；第三帧为后处理帧，实现 myRIO 应用程序结束运行后的设备重置功能。

4）程序实现

程序实现可分解为初始化、采集数据、发送报文封装、TCP 发送、本地显示、定时控制、设备重置等步骤。

（1）初始化。调用函数节点“打开 TCP 连接”（函数→数据通信→协议→TCP→打开 TCP 连接），建立 myRIO 和计算机之间的 TCP 连接，借助局部变量完成采集数据显示相关控件的初始化赋值。

（2）采集数据。为了聚焦 TCP 通信程序编写，这里采取 0～100 随机数产生的方式模拟数据采集过程。

（3）发送报文封装。调用函数节点“数值至十进制数字符串转换”（函数→编程→字符串→数值字符串转换→数值至十进制数字符串转换）将采集的数据转换为字符串格式，再调用函数节点“连接字符串”（函数→编程→字符串→连接字符串）构造以符号“,”为间隔的发送报文。

（4）TCP 发送。将第（3）步封装的发送报文作为函数节点“写入 TCP 数据”（函数→数据通信→协议→TCP→写入 TCP 数据）的发送内容，完成基于 TCP 的 myRIO 采集数据向服务器发送功能。

（5）本地显示。为了显示采集数据，使用“量表”控件显示 2 路采集参数，并将 2 路采集数据借助函数节点“捆绑”（函数→编程→簇、类与变体→捆绑）封装为簇类型数据，作为“波形图表”控件的显示内容，从而实现 2 路采集数据波形趋势曲线的显示。

（6）定时控制。为了便于观测，在 While 循环中添加函数节点“等待”（函数→编程→定时→等待），设置等待参数为 1000ms，实现每秒采集一次数据，并向服务器端发送一次数据帧的基本功能。

（7）设备重置。在第三帧中，调用函数节点“关闭 TCP 连接”（函数→数据通信→协议→TCP→关闭 TCP 连接），释放 TCP 通信程序所占用的资源，调用函数节点“Reset myRIO”（函数→myRIO→Device management→Reset）完成 myRIO 开发平台的复位工作。

完整的 TCP 客户端数据采集与发送程序实现如图 5-36 所示。

运行程序，TCP 客户端程序执行结果如图 5-37 所示。

此时网络调试助手作为 TCP 服务器接收到的数据如图 5-38 所示。


微课视频

2. myRIO 作为 TCP 服务器的无线局域网数据通信

myRIO 作为 TCP 服务器时，其所在的无线局域网既可以是 myRIO 创建的无线网络，也可以是 myRIO 选择接入的已有无线局域网。

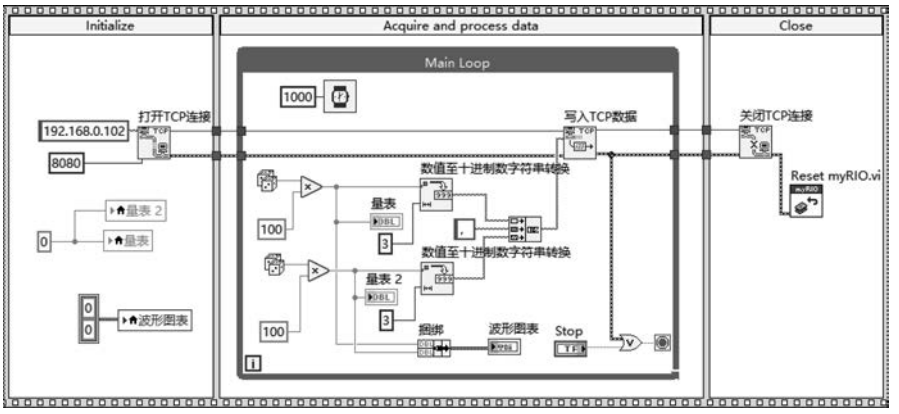

图 5-36　TCP 客户端数据采集与发送程序实现

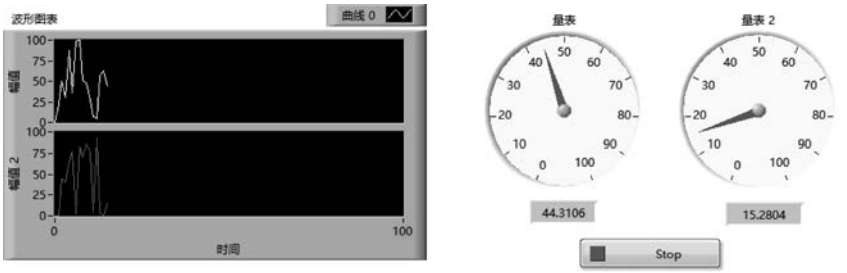

图 5-37　TCP 客户端程序执行结果

图 5-38　网络调试助手作为 TCP 服务器接收的数据

篇幅所限，这里使用网络调试助手软件工具模拟客户端应用程序。打开 Windows 操作系统控制面板，选择“网络和 Internet→WLAN→硬件属性”，可以查看无线局域网中计算机分配的 IPv4 地址为 192.168.0.102。该地址作为 TCP 通信客户端 IP 地址。

1）背景知识

TCP 服务器应用程序开发，其基本结构由“TCP 侦听”“读取 TCP 数据”和“关闭 TCP 连接”三个函数节点组成。

（1）节点“TCP 侦听”用于创建侦听器，等待接受 TCP 连接，该节点的使用需要指定服务器侦听的端口。

（2）函数节点“读取 TCP 数据”用于读取客户端发送的数据。

（3）函数节点“关闭 TCP 连接”用于释放 TCP 通信所占用的资源。

如果服务器需要连续接收客户端发送的数据，一般循环前调用函数节点“TCP 侦听”，循环中调用函数节点“读取 TCP 数据”，循环结束后调用函数节点“关闭 TCP 连接”。循环结构内调用函数节点“读取 TCP 数据”实现基于 TCP 的数据连续接收功能。这一程序结构设计，避免了连续发送中频繁建立连接、释放资源，有助于提高程序运行效率。

2）设计目标

这个案例中，将 myRIO 的 WiFi 连接配置为“接入无线网络”，myRIO 作为 TCP 服务器，监听指定端口（通过 MAX 软件，可以查看 myRIO 接入无线网络后获取的 IP 地址为 192.168.0.107），等待客户端连接请求。当客户端发起连接请求时，myRIO 建立其与客户端之间的 TCP 通信连接，并接收、显示来自客户端发送的数据。

3）设计思路

利用 myRIO 新建项目自动生成的程序模板，保留其三帧程序基本结构。第一帧为初始化帧，实现有关数据、控件的初始赋值，并启动指定端口的 TCP 监听，等待客户端连接请求；第二帧为主程序帧，完成应用程序的核心业务功能的开发，当读取指定字节数或者读取到回车换行符号时，程序将读取的数据作为一帧客户端发送的数据，按照预定的帧结构进行处理和显示；第三帧为程序后处理真，实现 myRIO 程序运行结束后的设备重置工作。

4）程序实现

程序实现可分解为初始化、TCP 读取、TCP 读取数据的类型转换、提取数组中测量值、myRIO 设备重置等步骤。

（1）初始化。调用函数节点“TCP 侦听”（函数→数据通信→协议→TCP→TCP 侦听），等待其他终端的连接请求，并借助局部变量完成采集数据显示相关控件的初始化赋值。

（2）TCP 读取。假设客户端发送的数据格式为“IP 地址：端口-数据 1-数据 2”＋CRLF 的字符串，所以调用函数节点“读取 TCP 数据”（函数→数据通信→协议→TCP→读取 TCP 数据），并设置其读取模式为“CRLF”及永不超时，即当读取到回车换行字符时，结束一帧数据的读取。

（3）TCP 读取数据的类型转换。读取到的字符串调用函数节点“电子表格字符串至数组转换”（函数→编程→字符串→电子表格字符串至数组转换），设置参数“分隔符”为“-”，设

置参数“格式字符串”为“%d”，设置参数“数组类型”为一维整型数组常量，完成字符串至字节数组转换。

(4) 提取数组中测量值。调用函数节点“索引数组”(函数→编程→数组→索引数组)，读取转换结果数组中的第 2 个和第 3 个数据元素，即客户端发送的采集数据 1、采集数据 2，借助仪表控件的方式显示接收客户端采集的数据。

(5) myRIO 设备重置。在程序第三帧中调用函数节点“关闭 TCP 连接”(函数→数据通信→协议→TCP→关闭 TCP 连接)，释放 TCP 通信程序所占用的资源，调用函数节点 Reset myRIO(函数→myRIO→Device management→Reset)完成 myRIO 开发平台的复位工作。

完整的 TCP 服务器程序如图 5-39 所示。

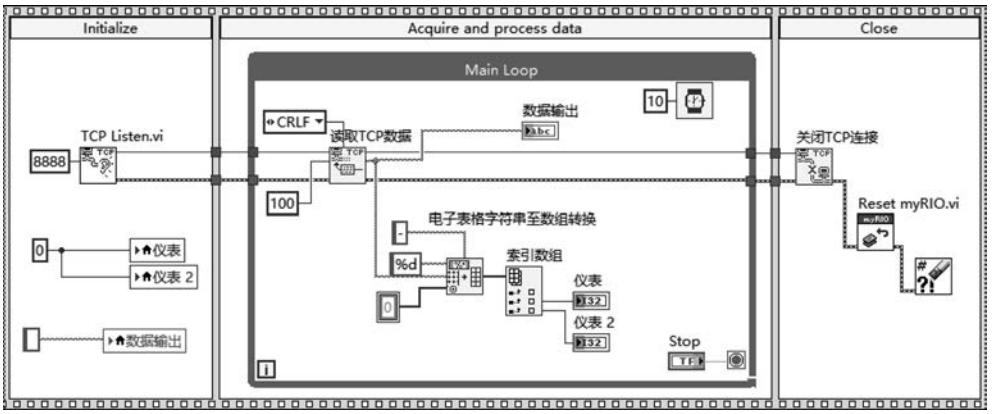

图 5-39　完整的 TCP 服务器程序

运行 myRIO 服务器端应用程序，等待客户端的连接请求。然后打开网络通信调试助手，设置“协议类型”为 TCP Client，设置远程主机地址为 myRIO 开发平台的 IP 地址及其侦听的端口号“192.168.0.107:8888”，单击“连接”按钮，建立计算机与服务器 myRIO 的 TCP 连接。

完成上述设置后，网络调试助手数据发送区输入字符串“192.168.0.102:3338-42-53”及回车换行符号，单击“发送”按钮，完成一次客户端模拟电子设备发送采集数据的操作，如图 5-40 所示。

此时，打开 myRIO 运行界面，myRIO 接收、解析并显示客户端发送数据结果如图 5-41 所示。

需要注意的是，客户端发送的数据如果忘记附加 CRLF(回车换行)，myRIO 应用程序中无法观测接收的数据——直至接收字节数达到 100 或者接收到回车换行字符。

3. myRIO 进行 UDP 广播发送的无线局域网数据通信

UDP 是局域网通信常用的协议，相比于 TCP，UDP 以其简单易用而广受欢迎。myRIO 使用 UDP 进行数据通信，同样首先必须配置其内置 WiFi 模块接入无线网络，该无


微课视频

图 5-40 网络调试助手作为 TCP 客户端发送数据

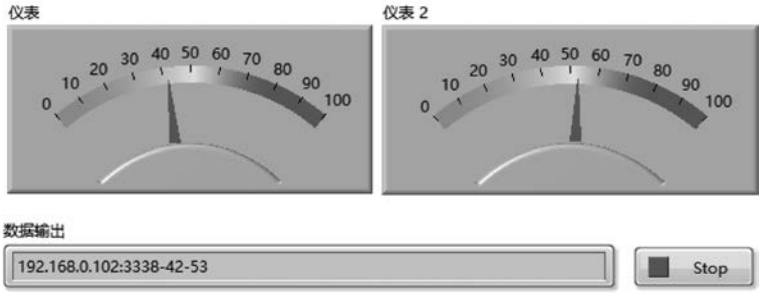

图 5-41 myRIO 接收、解析并显示客户端发送数据结果

线网络既可以是 myRIO 创建的无线网络，也可以是 myRIO 选择接入的无线局域网。

1）背景知识

UDP 通信应用程序开发，其基本结构由“打开 UDP”“写入 UDP 数据”和“关闭 UDP”三个函数节点组成。

（1）函数节点“打开 UDP”用于打开端口或服务名称的 UDP 套接字，该节点的使用需要指定本地通信端口。

（2）函数节点“写入 UDP 数据”将采集的数据写入 UDP 套接字。由于 LabVIEW 中数据通信均以字符串的形式进行，如果发送数据为非字符串格式，则需要进行转换。

(3) 函数节点“关闭 UDP”用于释放 UDP 通信所占用的资源。

如果 myRIO 需要连续向其他终端发送采集的数据,一般在循环前调用函数节点“打开 UDP”,循环结构内调用函数节点“写入 UDP 数据”实现基于 UDP 的数据连续发送功能,循环结束后调用函数节点“关闭 UDP”。这一程序结构设计,避免了连续发送中频繁建立连接、释放资源,有助于提高程序运行效率。

2) 设计目标

这个案例中,将 myRIO 内置的 WiFi 模块配置为“接入无线网络”(通过 MAX 软件可以查看接入无线网络后分配的 IPv4 地址为 192.168.0.107)。myRIO 采集数据,将采集数据以 UDP 通信报文的形式进行“广播”操作,使得同一网段内所有终端均可接收到 myRIO 采集的数据。

3) 设计思路

利用 myRIO 新建项目自动生成的程序模板,保留其三帧程序基本结构。第一帧为初始化帧,打开 UDP 通信套接字,设置 UDP 通信端口,设置 UDP 广播通信 IP 地址;第二帧为主程序帧,在该帧 While 循环结构内,以产生随机数的方式模拟实时数据采集,并将采集的数据以 UDP 数据报文的形式进行广播,同时通过实时曲线形式显示采集数据;第三帧为程序后处理帧,实现 myRIO 程序运行结束后的设备重置工作。

4)程序实现

篇幅所限,这里使用网络调试助手软件工具模拟其他 UDP 应用程序,为了检验广播机制效果,同时也使用手机端安装的网络调试助手模拟同一网段内不同 IP 地址的 UDP 应用程序。

myRIO 程序实现可分解为初始化、采集数据、定时控制、采集数据本地显示、UDP 发送数据、设备重置等步骤。

(1) 初始化。调用函数节点“打开 UDP”(函数→数据通信→协议→UDP→打开 UDP),设置本地 UDP 通信端口为 9999。由于意图实现 myRIO 采集的数据向局域网内所有终端发送,调用函数节点“字符串至 IP 地址转换”将广播地址“255.255.255.255”转换为节点“写入 UDP 数据”可用的 IP 地址,并设置指数数值常量 8080 作为节点“写入 UDP 数据”需要设置的通信端口。

(2) 采集数据。为了简化程序设计,聚焦 UDP 通信,这里采取借助 While 循环结构中产生随机数的方式模拟采集数据,并调用函数节点“数值至小数字符串转换”(函数→编程→字符串→数值字符串转换→数值至小数字符串转换),将其转换为字符串类型数据。

(3) 定时控制。为了便于观测,While 循环结构内调用函数节点“等待”(函数→编程→定时→等待),设置等待时长为 1000ms,实现每秒采集一次数据,UDP 广播一次的功能。

(4) 采集数据本地显示。采集的数据借助波形图表和数值显示控件进行显示——实际上这一步骤并不是必需的,因为 myRIO 应用程序独立部署运行过程中不可能在计算机上查看程序运行情况,初学阶段这样处理可以用来诊断应用程序的执行过程,而部署运行时,可以删除前面板相关显示功能。

(5) UDP 发送数据。调用函数节点“写入 UDP 数据”(函数→数据通信→协议→UDP→写入 UDP),将采集数据转换为字符串的结果在局域网内进行广播,完成 UDP 通信数据报文的发送。

(6) 设备重置。在程序第三帧中调用函数节点“关闭 UDP”(函数→数据通信→协议→UDP→关闭 UDP),释放 UDP 通信程序所占用的资源,调用函数节点“Reset myRIO”(函数→myRIO→Device management→Reset)完成 myRIO 开发平台的复位工作。

完整的 myRIO 数据采集及 UDP 广播程序实现如图 5-42 所示。

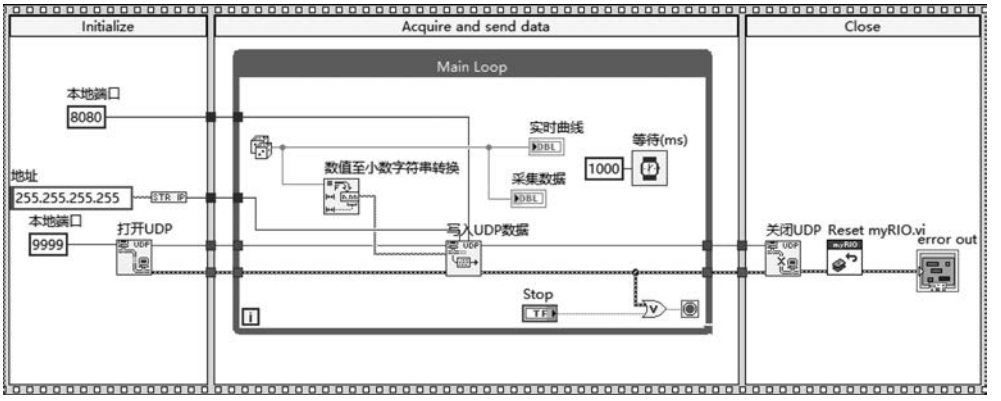

图 5-42 myRIO 数据采集及 UDP 广播程序实现

编译运行程序,myRIO 端数据采集及 UDP 广播程序执行结果如图 5-43 所示。

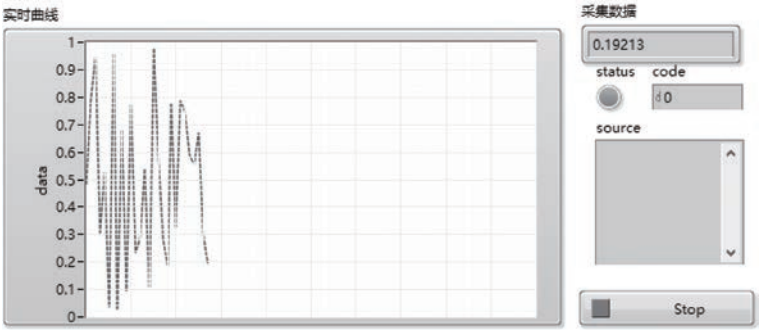

图 5-43 myRIO 端数据采集及 UDP 广播程序执行结果

打开 Windows 操作系统中网络调试助手,设置协议类型为 UDP,设置本地主机端口与 myRIO 发送端口保持一致,设置为 8080,单击“连接”按钮,启动 UDP 终端监听程序,可以观测到计算机端网络调试助手接收 myRIO 发送 UDP 数据包如图 5-44 所示。

打开手机端安装的“网络调试助手”(安卓操作系统),创建 UDP 服务端,设置端口 8080,与 myRIO 发送端口保持一致,进入 UDP 监听模式,可以观测到手机端网络调试助手接收到的 16 进制 UDP 数据包,如图 5-45 所示。

可以看到安卓系统中网络调试助手最后一次接收的 ASCII 编码格式的数据 30 2E 31

39 32 31 33 30，对应的 10 进制数据为 0.192130，这一结果与计算机端网络调试助手接收的数据、myRIO 端采集的数据完全一致，说明了 myRIO 采集的数据可以广播至局域网内全部终端。这一功能对于分布式数据采集系统的设计与开发具有十分重要的实践意义。

图 5-44　计算机端网络调试助手接收 myRIO 发送 UDP 数据包

图 5-45　手机端网络调试助手接收到的 16 进制 UDP 数据包

当然，如果希望进行点对点通信，只需要在程序设计时将上述程序框图中的广播地址255.255.255.255更改为局域网内目标终端的IP(192.168.0.*)即可。

5.2.4 基于WiFi的物联网通信应用实例

如果配置myRIO的WiFi模块为“连接无线网络”，而其连接的热点由接入互联网的路由器提供，则myRIO可以在无线局域网通信的基础上，进一步实现互联网通信。典型的应用场景就是myRIO作为智能终端采集数据，利用内置的WiFi模块接入互联网，将采集的数据上传至物联网平台，或者接收物联网平台推送的消息。这样一来，开发者可以借助物联网平台，将单机环境myRIO应用程序升级改造为“云、网、端”技术架构下的myRIO应用程序，从而使得myRIO嵌入式平台可以在各大应用系统规模下发挥更加重要的作用。

目前市面上可以供使用的物联网云平台有很多，如中移物联(OneNET)、TLink、机智云、阿里云、百度云、涂鸦云、传感云、乐为物联及巴法云等。其中，对开发者比较开放的物联网平台为TLink和机智云。

TLink物联网平台是一个免费开放的设备连接平台，主要应用于工业领域。对于初学者或者开发者来说，TLink物联网平台最大的特点是提供丰富的联网方式，支持TCP/UDP/HTTP/MQTT/ ModBus/COAP/MB TCP/NB-IOT等主流物联网通信协议，包含工业应用的几乎所有场景，具有重要的学习与实践价值。

基于物联网平台的应用系统开发，一般分为两个阶段——第一阶段为云平台物联网设备创建，第二阶段为物联网终端设备开发。

第一阶段：物联网平台云端设备创建。

注册后进入控制台。首先在控制台左侧导航栏选择“设备管理→添加设备”，打开TLink物联网平台云端设备创建页面，如图5-46所示。

图5-46 TLink物联网平台云端设备创建页面

云端设备创建页面中需要依次设置以下 6 类参数。

(1) 设备分组。对设备进行分组,分组之后在设备管理页面进行调整。

(2) 设备名称。开发者可自定义云端设备的名称。

(3) 链接协议。选择云端设备支持的通信协议。

(4) 掉线延时。此时间只与“已连接”和“未连接”状态有关系,如果在该时间范围内没有数据传到平台,那么该设备连接状态显示“未连接”。所以此时间要设置为比实际上行数据间隔要大,才不会在正常传输数据过程中出现“未连接”。最小值 60s。

(5) 传感器。添加不同类型的数值,用来显示设备的不同变量,一个传感器代表设备的一个变量,比如 PLC 的寄存器变量。

(6) 位置信息。给设备标注一个地理位置,可以通过搜索框输入地名、搜索框输入经纬度、地图上单击一个位置等任何一种方式完成设备位置标注。

完成云端设备创建,即可按照链接协议定义的云端通信数据帧格式,由本地计算机或者嵌入式终端,向云端设备发送数据或接收云端返回的信息。值得注意的是,TLink 物联网平台中云端设备通信协议支持用户自定义,图 5-47 为自定义协议示例。

图 5-47　自定义协议示例

该协议规定云端设备通信数据帧格式为字节数组类型,帧头为 FF,帧尾为 EE,每帧数据包含 2 个传感器测量数据,均由一字节表示测量值。

用户自定义协议设计方法,详情参见 TLink 物联网平台开发者中心相关文档说明。

第二阶段:物联网终端设备开发。

物联网终端设备开发可以基于单片机、嵌入式、PC 等任意一种开发平台,当然也包括本书所介绍的 myRIO 开发平台。物联网终端设备一般为数据采集终端,也可以是控制终

端，还可以是数据采集和自动控制二者兼备的复杂终端。物联网终端设备的关键在于能够基于物联网相关通信协议，将采集数据发布于物联网平台或访问物联网平台获取其他终端发布信息，并根据获取信息执行相应的动作，实现跨平台多终端协同的复杂应用。物联网云平台支持的联网协议较多，如 TCP、UDP、HTTP、MQTT 等，给了开发者极大的选择自由。本书将通过实例分别介绍基于 myRIO 的典型物联网应用程序开发。


微课视频

1. 基于 TCP 的采集数据上传物联网平台

如前所述，基于 TCP 的采集数据上传物联网云平台功能开发分为两个阶段。首先是 TLink 物联网平台云端 TCP 设备创建。

登录后进入控制台。控制台左侧导航栏选择“设备管理→添加设备”，打开 TLink 物联网平台云端设备创建页面，创建云端 TCP 设备，如图 5-48 所示。

图 5-48　创建云端 TCP 设备

TCP 设备的创建意味着各类终端可以通过 TCP 实现采集数据上报至 TLink 物联网平台（图中 TCP 设备包含 2 个传感器）的基本功能。

然后在控制台左侧导航栏选择“设备管理→设备列表”，选择创建的 TCP 设备，单击所选设备行操作链接“设备连接”，进入云端 TCP 设备的通信连接设置页面，获取设备连接参数信息，定义云端 TCP 设备用户数据协议，如图 5-49 所示。

设备连接参数包括云端 TCP 设备服务器 IP、端口号及设备序列号。TLink 物联网平台 TCP 服务器地址为 tcp.tlink.io，监听端口号为 8647。

当前协议部分可查看或者创建云端 TCP 设备对应数据协议的结构形式（具体创建方法参见 TLink 物联网平台开发者文档有关说明）。本节案例中数据协议为字节数组类型，该协议规定每个数据帧由 4 字节组成：字节 1 为数据帧为帧头，由 0xFF 组成；字节 2 表示传感器 1 测量值；字节 3 表示传感器 2 测量值；字节 4 为帧尾，由 0xEE 组成。

图 5-49　定义云端 TCP 设备用户数据协议

TLink 物联网平台规定，向云端 TCP 设备上报测量数据，首先以字符数据发送方式上传设备序列号，然后再按照预先定义的数据帧结构，以字节数组的形式上传测量数据。

完成上述准备工作之后，即可开启基于 TCP 的 myRIO 采集数据云端发布应用程序设计。

1）设计目标

该案例设计的目标就是 myRIO 作为物联网数据采集终端，按照 TLink 物联网平台中创建的云端 TCP 设备对应的数据协议，将采集的数据封装为云端 TCP 设备可用的数据帧，以 TCP 通信的方式将数据帧上传至 TLink 物联网平台，奠定多终端共享数据的基础。

2）设计思路

利用 myRIO 新建项目自动生成的程序模板，保留其三帧程序基本结构。第一帧为初始化帧，借助局部变量完成采集数据显示相关控件的初始化赋值，建立 myRIO 和 TLink 物联网平台中 TCP 设备服务器之间的连接；第二帧为主程序帧，在该帧 While 循环结构内，以产生随机数的方式模拟 2 路实时数据采集，并将采集的数据封装为上报物联网云平台的数据帧，完成数据帧的 TCP 发送；第三帧为程序后处理帧，实现 myRIO 程序运行结束后的设备重置工作。

3）程序实现

myRIO 程序实现可分解为初始化、采集数据、生成 TCP 上传数据帧、TCP 发送、采集数据本地显示、定时控制、设备重置等步骤。

（1）初始化。调用函数节点“打开 TCP 连接”（函数→数据通信→协议→TCP→打开 TCP 连接），设置服务器 IP 地址为 tcp. tlink. io，设置服务器端口为 8647，设置本地通信端口为 8888，建立 myRIO 和 TLink 物联网平台中 TCP 设备服务器之间的 TCP 连接；创建

局部变量，完成程序前面板显示控件初始化操作。

(2) 采集数据。采取产生 0～255 的随机数并调用函数节点“转换为无符号单字节整型”(函数→编程→数值→转换→转换为无符号单字节整型)，模拟整数型数据采集过程。

(3) 生成 TCP 上传数据帧。调用函数节点“创建数组”(函数→编程→数组→创建数组)构造[FF][D1][D2][EE]形式的字节数组类型数据帧，完成云端 TCP 设备数据帧结构的通信报文创建；然后调用函数节点“字节数组至字符串转换”(函数→编程→字符串→路径\数组\字符串转换→字节数组至字符串转换)，将创建的数据帧转换为字符串类型数据，作为节点“写入 TCP 数据”的发送内容。

(4) TCP 发送。特别需要指出的是，按照 TLink 物联网平台规定，向云端 TCP 设备上报数据，必须首先发送对应的 TCP 设备序列号，因此在发送数据帧之前首先调用函数节点“写入 TCP 数据”(函数→数据通信→协议→TCP→写入 TCP 数据)完成字符串类型的 TCP 设备序列号发送，延时 50ms，再开始(3)中创建的数据帧发送，实现采集数据向 TLink 物联网平台的上报功能。

(5) 采集数据本地显示。为了显示采集数据数值，设置滑动条显示 2 路采集参数，并将 2 路采集数据借助节点“捆绑”(函数→编程→簇、类与变体→捆绑)封装为簇类型数据，作为“波形图表”控件的显示内容，实现 2 路采集数据波形的实时显示。

(6) 定时控制。为了便于观测，While 循环中添加函数节点“等待”(函数→编程→定时→等待)，设置等待参数为 3000ms，实现每 3 秒采集一次数据，并向云端 TCP 设备上传一次数据的基本功能(物联网平台为了避免恶意上报导致网络阻塞影响用户使用体验，禁止高速刷新设备数据)。

(7) 设备重置。在 myRIO 程序第三帧中调用函数节点“关闭 TCP 连接”(函数→数据通信→协议→TCP→关闭 TCP 连接)，释放 TCP 通信程序所占用的资源，调用函数节点“Reset myRIO”(函数→myRIO→Device management→Reset)完成 myRIO 开发平台的复位工作。

myRIO 采集数据并上报云端 TCP 设备的完整程序如图 5-50 所示。

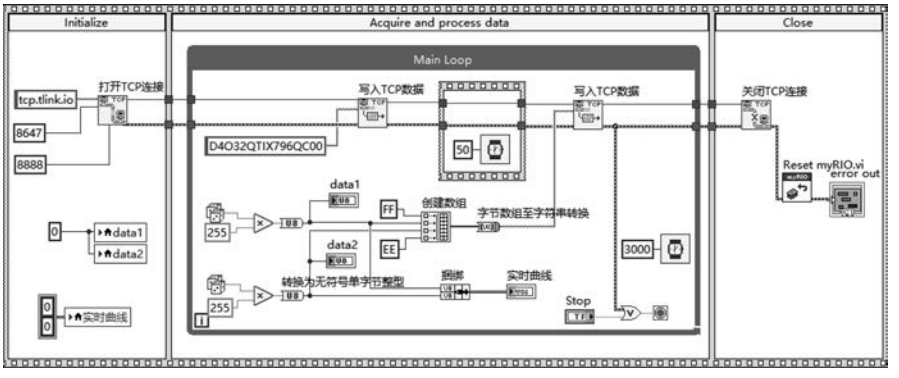

图 5-50　myRIO 采集数据并上报云端 TCP 设备的完整程序

运行程序，myRIO 端显示数据采集结果如图 5-51 所示。

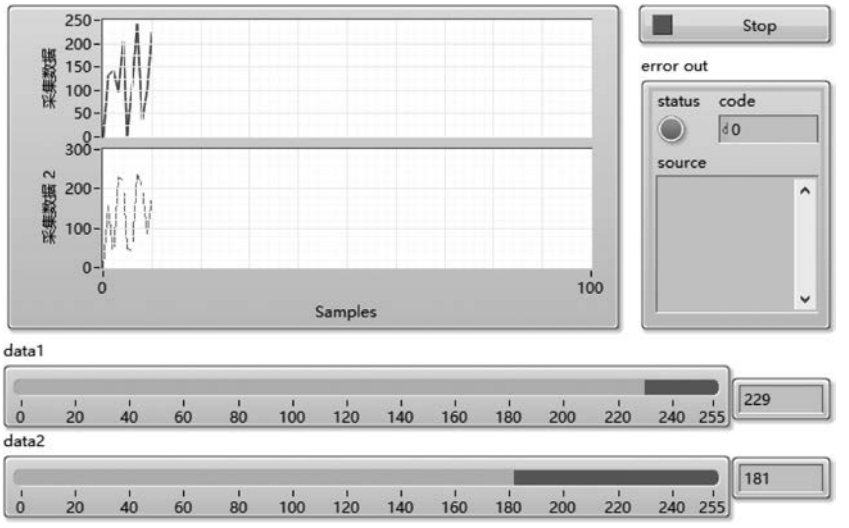

图 5-51　myRIO 端显示数据采集结果

此时登录 TLink 物联网平台，打开监控中心，可观测到对应的云端 TCP 设备名称已经由默认的灰色转变为黑色（表示基于 myRIO 的实体设备与物联网平台云端虚拟设备已经连接），对应的传感器连接状态指示显示“已连接”，云端 TCP 设备数据接收结果与 myRIO 采集数据完全一致，如图 5-52 所示。

图 5-52　云端 TCP 设备数据接收结果

2. 基于 UDP 的采集数据上传物联网平台


微课视频

如前所述，基于 UDP 的采集数据上传物联网云平台功能开发分为两个阶段。首先是 TLink 物联网平台云端 UDP 设备创建。

登录后进入控制台。控制台左侧导航栏选择“设备管理→添加设备”，打开 TLink 物联网平台云端设备创建页面，创建云端 UDP 设备，如图 5-53 所示。

UDP 设备的创建意味着各类终端可以通过 UDP 实现采集数据上报至 TLink 物联网平台（图 5-53 中 UDP 设备包含 2 个传感器）的基本功能。

图 5-53　创建云端 UDP 设备

然后在控制台左侧导航栏选择“设备管理→设备列表”，选择创建的 UDP 设备，单击所选设备行操作连接，进入云端 UDP 设备的通信连接设置页面，获取设备连接参数信息，定义云端 UDP 设备用户数据协议，如图 5-54 所示。

图 5-54　定义云端 UDP 设备用户数据协议

设备连接参数包括云端 UDP 设备服务器 IP、端口号及设备序列号。TLink 物联网平台 UDP 服务器地址为 udp.tlink.io，监听端口号为 9896。

当前协议部分可查看或者创建云端 UDP 设备对应数据协议的结构形式（具体创建方法参见 TLink 物联网平台开发者文档有关说明）。本案例中数据协议为 ASCII 编码字符串类型，每个数据帧由 5 个字符串子串组成：子串 1 为数据帧为帧头，由字符@组成；子串 2 为传感器 1 测量值转换的字符串；子串 3 为间隔符号“-”，子串 4 为传感器 2 测量值转换的

字符串；子串 5 为帧尾，由字符@组成。TLink 物联网平台规定，向云端 UDP 设备上报测量数据，如果定义数据帧结构为字符串，则将设备序列号和数据帧 2 个字符串类型数据进行连接，形成完整的云端 UDP 设备通信数据报文。

完成上述准备工作之后，即可开启基于 UDP 的 myRIO 采集数据云端发布应用程序设计。

1）设计目标

本案例设计的目标是 myRIO 作为物联网数据采集终端，按照 TLink 物联网平台中创建的云端 UDP 设备对应的数据协议，将采集的数据封装为云端 UDP 设备可用的数据帧，以 UDP 通信的方式将数据帧上传至 TLink 物联网平台，奠定多终端共享数据的基础。

2）设计思路

利用 myRIO 新建项目自动生成的程序模板，保留其三帧程序基本结构。第一帧为初始化帧，借助局部变量完成采集数据显示相关控件的初始化赋值，打开 UDP 通信套接字；第二帧为主程序帧，在该帧 While 循环结构内，按照指定的时间间隔，以产生随机数的方式模拟 2 路实时数据采集，并将采集的数据封装为上报物联网云平台的 UDP 数据报文，完成数据报文的 UDP 发送；第三帧为程序后处理帧，实现 myRIO 程序运行结束后的设备重置工作。

3）程序实现

myRIO 程序实现可分解为初始化、采集数据、生成上传数据报文、UDP 发送、采集数据本地显示、定时控制、设备重置等步骤。

（1）初始化。调用函数节点“打开 UDP”（函数→数据通信→协议→UDP→打开 UDP 连接），设置本地端口 8888，创建 myRIO 连接 TLink 物联网平台的 UDP 套接字，并借助局部变量完成采集数据显示相关控件的初始化赋值。

（2）采集数据。采取 0～100 随机数产生的方式模拟数据采集过程，调用函数节点“数值至小数字符串转换”（函数→编程→字符串→数值字符串转换→数值至小数字符串转换）将采集的数据转换为字符串类型。

（3）生成上传数据报文。调用函数节点“连接字符串”（函数→编程→字符串→连接字符串）构造“设备序列号@采集数据 1-采集数据 2@”格式的 UDP 设备数据帧结构的通信报文。

（4）UDP 发送。调用函数节点“写入 UDP 数据”（函数→数据通信→协议→UDP→写入 UDP 数据）完成数据帧发送。特别需要指出的是，TLink 中 UDP 服务器 IP 地址字符串“47.106.61.135”需要通过函数节点“字符串至 IP 地址转换”（函数→数据通信→协议→TCP→字符串至 IP 地址转换）设置节点“写入 UDP”所需的远程主机 IP 地址参数值。设置节点“写入 UDP”所需的远程主机端口号为 TLink 物联网平台 UDP 设备服务器端口 9896。

实际上，TLink 物联网平台中仅提供了其 UDP 设备服务器对应的域名“udp.tlink.io”而 UDP 通信程序编写时使用的函数节点“写入 UDP 数据”仅接受 IP 地址，并不接受域名。可在命令行窗口中键入“ping udp.tlink.io”，查看 TLink 物联网平台 UDP 服务器 IP 地址，

如图 5-55 所示。

图 5-55　命令行 ping 指令查看 UDP 远程主机 IP 地址

(5) 采集数据本地显示。为了显示采集数据数值,设置仪表控件显示 2 路采集参数,并将 2 路采集数据借助节点“捆绑”(函数→编程→簇、类与变体→捆绑)封装为簇类型数据,作为“波形图表”控件的显示内容,从而实现 2 路采集数据波形的实时显示。

(6) 定时控制。为了便于观测,While 循环中添加节点“等待”(函数→编程→定时→等待),设置等待参数为 3000ms,实现每 3 秒采集一次数据,并向云端 UDP 设备上传一次数据的基本功能(物联网平台为了避免恶意上报导致网络阻塞影响用户使用体验,禁止高速刷新设备数据)。

(7) 设备重置。在程序第三帧中调用函数节点“关闭 UDP 连接”(函数→数据通信→协议→UDP→关闭 UDP 连接),释放 UDP 通信程序所占用的资源,调用函数节点“Reset myRIO”(函数→myRIO→Device management→Reset)完成 myRIO 开发平台的复位工作。

myRIO 采集数据并上报云端 UDP 设备的完整程序如图 5-56 所示。

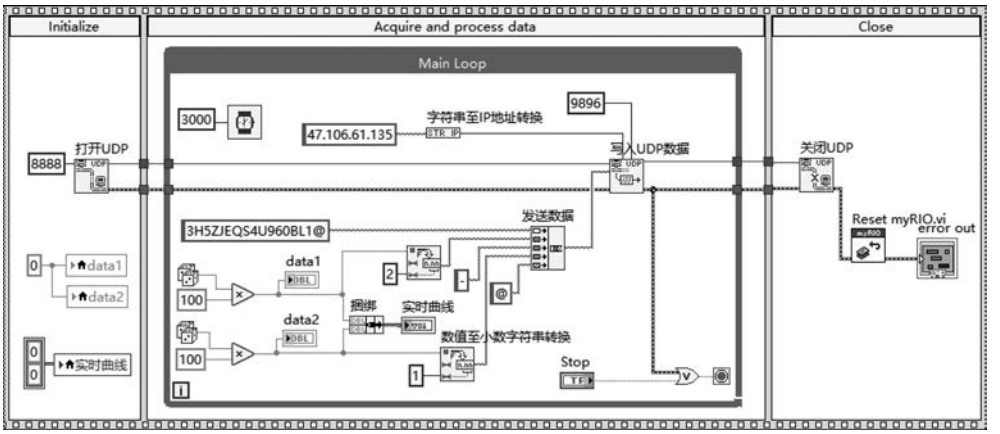

图 5-56　myRIO 采集数据并上报云端 UDP 设备的完整程序

运行程序,myRIO 端程序执行结果如图 5-57 所示。

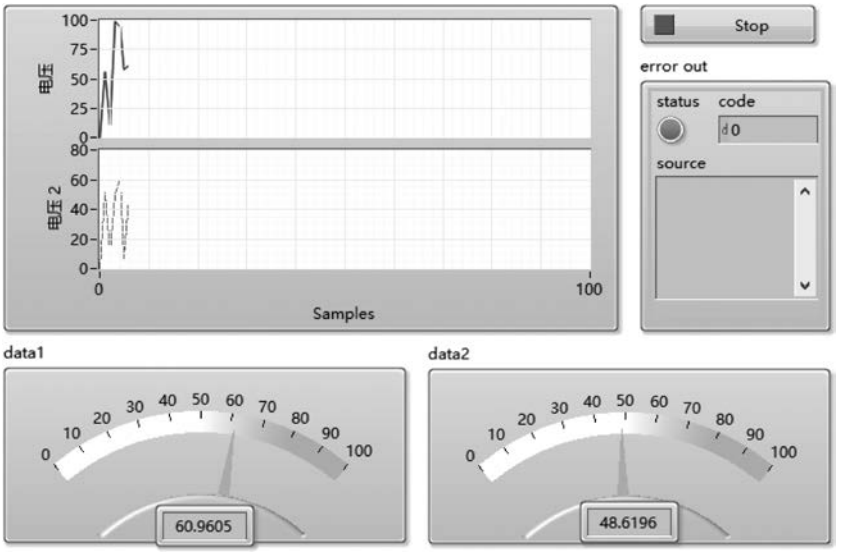

图 5-57　myRIO 端程序执行结果

此时登录 TLink 物联网平台，打开监控中心，可观测到对应的 UDP 设备名称已经由默认的灰色转变为黑色（表示基于 myRIO 的实体设备与物联网平台云端虚拟 UDP 设备已经连接），对应的传感器连接状态指示显示“已连接”，云端 UDP 设备数据接收结果与 myRIO 采集数据完全一致，如图 5-58 所示。

图 5-58　云端 UDP 设备数据接收结果

3. 基于 HTTP 的采集数据上传物联网平台


微课视频

基于 HTTP 的采集数据上传物联网平台应用程序是一个典型的 HTTP 客户端应用程序。应用程序基本结构由“打开句柄”“POST”或“GET”“关闭句柄”三个函数节点组成。如果需要特别设置 HTTP 请求头部参数，则还需要调用函数节点“添加头”。

（1）函数节点“打开句柄”用于创建一个 HTTP 请求的引用。

（2）函数节点“POST”用于向服务器端发送服务请求数据。

（3）函数节点“添加头”用于设置 HTTP 请求必需的头部参数。

（4）函数节点“关闭句柄”用于释放 HTTP 通信所占用的资源。

如果客户端需要连续向服务器端发送 HTTP 请求，一般将节点“打开句柄”置于循环结

构之外。循环前调用函数节点“打开句柄”，必要时，还会调用函数节点“添加头”；循环结束后调用函数节点“关闭句柄”；循环结构内调用函数节点“POST”实现连续发起 HTTP 服务请求。这一程序结构设计，避免了连续发送中频繁建立连接、释放资源，有助于提高程序运行效率。

如前所述，基于 HTTP 的采集数据上传物联网云平台功能开发首先需要完成物联网平台云端 HTTP 设备创建。

登录后进入控制台。控制台左侧导航栏选择“设备管理→添加设备”，打开 TLink 物联网平台云端设备创建页面，创建云端 HTTP 设备，如图 5-59 所示。

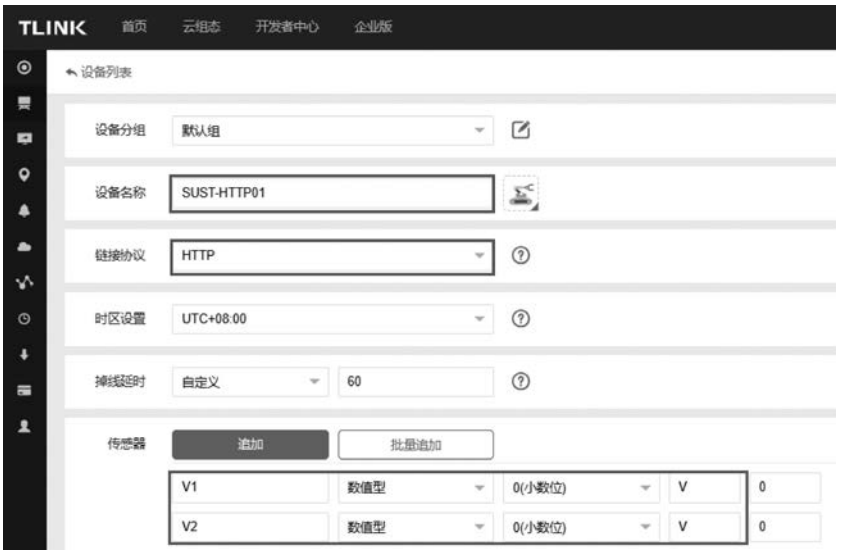

图 5-59　创建云端 HTTP 设备

HTTP 设备的创建意味着各类终端可以通过 HTTP 实现采集数据上报至 TLink 物联网平台(图 5-59 中 TCP 设备包含 2 个传感器)的基本功能。

TLink 中创建的 HTTP 设备，理所当然地使用 HTTP 进行数据发布和访问。基于 HTTP 完成传感器数据上报相关申请信息如下。

请求方式：POST。

API 地址：http://api.tlink.io/api/device/sendDataPoint。

需要设置以下 4 个头部参数。

Authorization：Bearer＋空格＋Access Token，注意 Bearer 后面的空格。

TLinkAppId：客户端 clientId。

Content-Type：application/json。

cache-control：no-cache。

HTTP 请求 Body 参数为 JSON 格式，根据采集数据常见的数值类型、定位类型、开关类型、字符串型，制定了基本规范如下所示。

```
{
  "userId":20*****080,                    //用户 ID,必选
  "addTime":"2019-10-28 12:01:00",        //上报时间,可选
  "deviceNo":"2U8******N8I9Z",            //设备序列号,必选
  "sensorDatas":[//上报的数据集合
  {//数值型,档位型
      "sensorsId":20****003,              //传感器 ID
      "value":"10.0"                      //数值型数值
  },
  {//定位型
      "sensorsId":20******6,              //传感器 ID
      "lat":39.9,                         //定位型纬度
      "lng":116.3                         //定位型经度
  },
  {//开关型
      "sensorsId":20****597,              //传感器 ID
      "switcher":"1"                      //开关型数值
  },
  {//字符串型
      "sensorsId":20******98,             //传感器 ID
      "string":"字符 ABCabc"              //字符串数据
  }
 ]
}
```

应用系统开发时,可根据实际数据采集的数据类型对上述规范进行裁剪,可以仅包含数值类型数据,亦可包含全部四种数据类型。

本案例查看 TLink 个人账号信息可得知用户 ID,定义的 HTTP 设备序列号为"A7A××××××××××",该设备包括 2 个传感器参数,其传感器 ID 分别为 200585202 和 200585203,因此可构造其 Body 参数如下所示。

```
{
  "userId":200033466,                     //必须填写
  "deviceNo": "A7A××××××××××",           //设备序列号,必选
  "sensorDatas": [                        //上报的数据集合
    {
      "sensorsId": 200585202,             //传感器 ID
      "value":"10.0"                      //数值型数值
    },
    {
      "sensorsId": 200585203,             //传感器 ID
      "value": "20.0"                     //数值型数值
    }
  ]
}
```

物联网平台中 HTTP 设备的访问、数据发布可以使用任何标准的 HTTP 客户端访问 TLink 提供的相关 API 接口。为了增强 HTTP 安全机制,TLink 物联网平台 HTTP 相关

API接口访问使用了OAuth 2.0的权限管理，即基于HTTP访问相关API，需要提供Access Token方可调用。

AccessToken是客户端访问资源服务器的令牌。拥有这个令牌代表得到用户的授权。然而，这是个临时授权，有一定有效期。Access Token限定一个较短的有效期可以降低因Access Token泄露而带来的风险。

为了安全，Oauth 2.0引入Refresh token和Client Secret两个措施，刷新Access Token时，需要验证Client Secret，Refresh Token的作用是刷新Access Token。Refresh Token的有效期非常长，会在用户授权时随Access Token一起重定向到回调url，传递给客户端。

首次获取Access Token，必须先查找TLink物联网平台个人账号信息中的Client ID、secret。TLink物联网平台注册后，用户中心可以查看到账号相关信息。

首次获取accessToken需要可以使用HTTP调试助手PostMan。打开PostMan，选择认证类型，设置为OAuth 2.0，如图5-60所示。

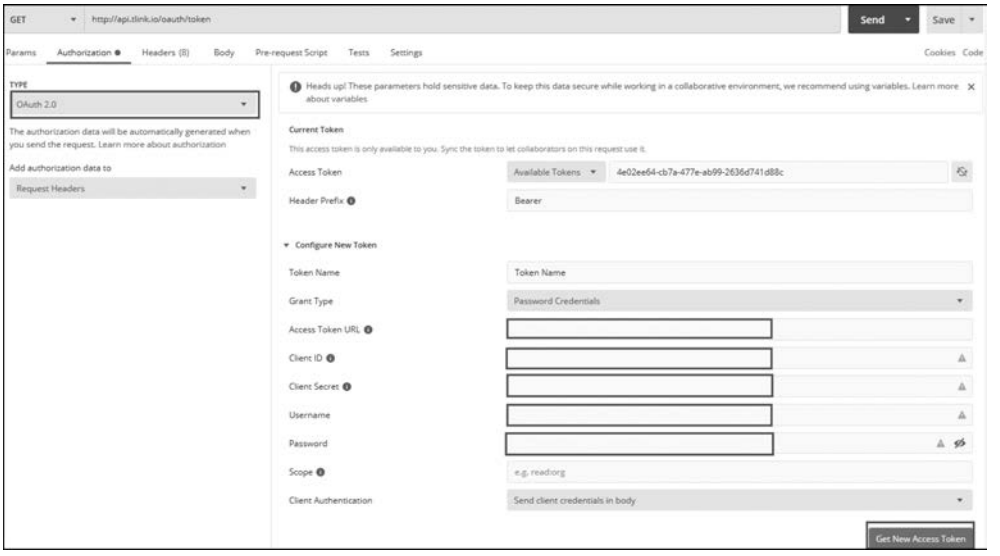

图5-60　PostMan中OAuth 2.0生成访问令牌配置窗口

单击“Get New Access Token”按钮，弹出如图5-61所示的窗口，图中Access Token就是获取的访问令牌。

由于Access Token具有一定的使用期限，因此过期后不能继续使用。幸运的是，申请Access Token时，还返回一个refresh_token(57a883b8-9417-4b68-bcbb-5d646a70d2e9)，refresh_token可以保持在很长一段时间内(60天)不改变。TLink物联网平台提供了利用refresh_token获取最新Access Token，可以通过向服务器发出HTTP POST请求的方式进行。

POST请求地址如下所示。

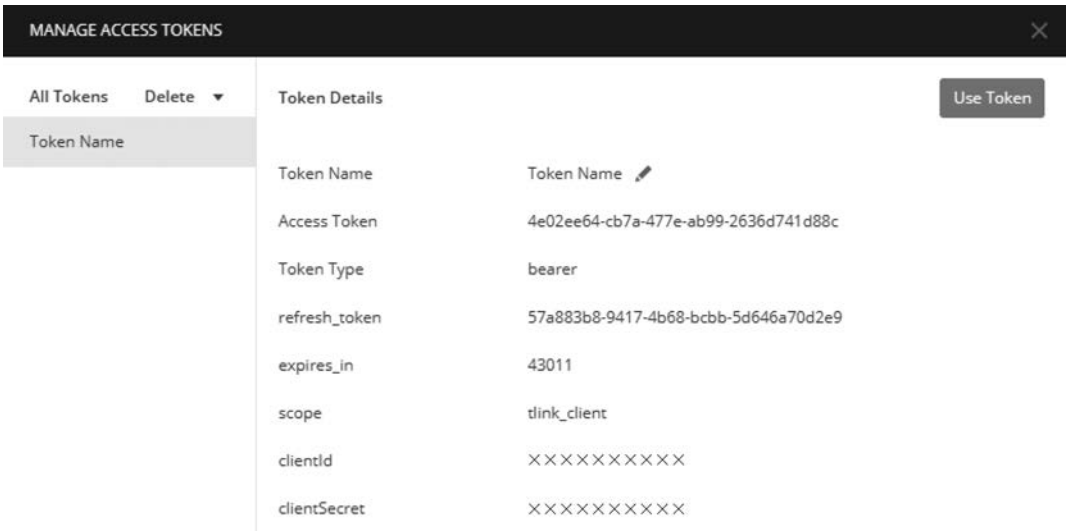

图 5-61　访问令牌创建结果

http://api.tlink.io/oauth/token? grant_type=refresh_token&refresh_token= *** &client_id= *** &client_secret= ***

将请求地址中的 *** 替换为用户账号中对应的信息，并发起一个请求，返回结果如图 5-62 所示。

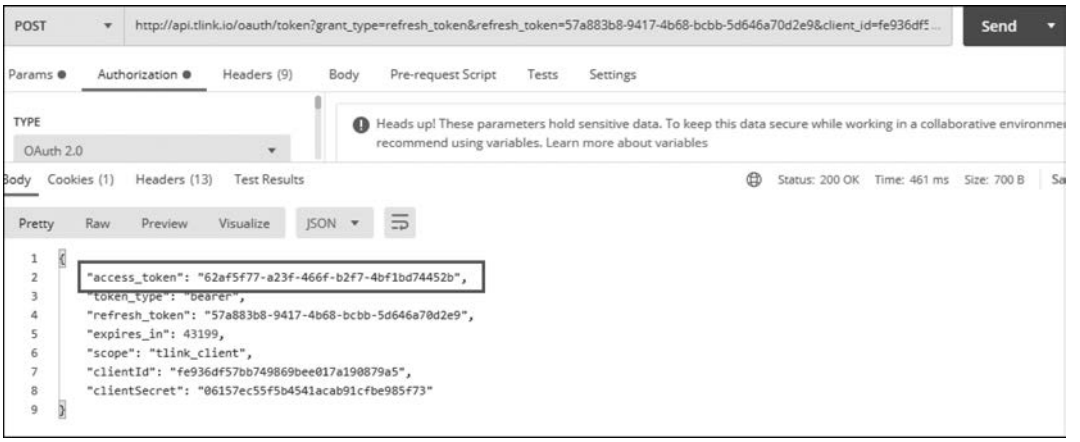

图 5-62　刷新访问令牌测试

可以获取刷新后的 access_token 为"62af5f77-a23f-466f-b2f7-4bf1bd74452b"，亦可观测到对应的 refresh_token 为"57a883b8-9417-4b68-bcbb-5d646a70d2e9"。

根据前述 TLink 中 HTTP 设备访问 API 接口相关参数说明及 HTTP 访问 API 接口的安全机制，调用 API 接口必须使用最新的 Access Token 或者使用 refresh_token 获取最新 Access Token。

完成上述准备工作之后，即可开启基于 HTTP 的 myRIO 采集数据云端发布应用程序设计。

1）设计目标

本案例设计的目标是 myRIO 作为物联网数据采集终端，按照 TLink 物联网平台中创建的云端 HTTP 设备对应的数据协议，将采集的数据封装为云端 HTTP 设备可用的数据报文，以 HTTP 通信的方式将数据帧上传至 TLink 物联网平台，奠定多终端共享数据的基础。

2）设计思路

利用 myRIO 新建项目自动生成的程序模板，保留其三帧程序基本结构。第一帧为初始化帧，完成 HTTP 客户端创建以及头部参数设置；第二帧为主程序帧，在该帧 While 循环结构内，按照指定的时间间隔，以产生随机数的方式模拟 2 路实时数据采集和数据显示，并将采集的数据封装为物联网平台云端 HTTP 设备数据报文，上报物联网云平台 HTTP 设备服务器；第三帧为程序后处理帧，实现 myRIO 程序运行结束后的设备重置工作。

3）程序实现

myRIO 程序实现可分解为初始化、采集数据、生成 HTTP 请求体、POST 请求、采集数据本地显示、定时控制、设备重置等步骤。

（1）初始化。调用函数节点“打开句柄”，建立 myRIO 和 TLink 物联网平台中 HTTP 设备服务器之间的 HTTP 连接引用，并连续三次调用函数节点“添加头”，依次设置 HTTP 请求的头部参数 Content-Type 取值为 application/json；头部参数 Authorization 取值为 Bearer AccessToken；头部参数 TLinkAppId 取值为 TLink 中用户的 Client ID。

其中 AccessToken 调用自定义函数节点 GetAccessToken. vi 获取。该函数节点的实现思路就是按照 TLink 中的 Access Token 刷新 API 使用方法，调用函数节点“POST”，设置其参数 url 取值为 API 接口地址及 refreshToken、client_id、client_secret 等参数合并结果。HTTP 请求结果调用 JSON API 工具包中提供的函数节点解析出 JSON 格式的请求结果中成员 access_token 的取值，作为最新的 Access Token 取值。自定义函数节点 GetAccessToken. vi 程序实现如图 5-63 所示。

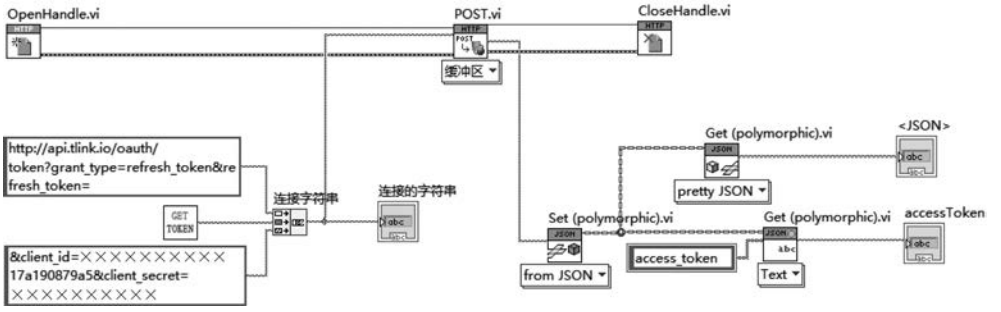

图 5-63　自定义函数节点 GetAccessToken. vi 程序实现

自定义函数节点 GetAccessToken.vi 中使用了自定义函数节点 GetRefreshToken.vi(节点图标显示文本 GET TOKEN),该函数节点的功能就是读取用户载入 myRIO 板载硬盘的文件 RefreshToken.txt,获取处于生命周期内的 refreshtoken,作为刷新 Access Token 对应的 API 调用参数。自定义函数节点 GetAccessToken.vi 程序实现如图 5-64 所示。

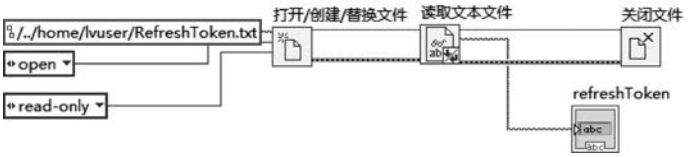

图 5-64　自定义函数节点 GetAccessToken.vi 程序实现

本案例中,myRIO 首先通过 Refresh Token 及 Client ID 等参数刷新鉴权服务器,获取最新 Access Token。为了便于 myRIO 应用程序使用处于生命周期内的 Refresh Token,将 Refresh Token 存储为 txt 格式的文本文件,以 RefreshToken.txt 为文件名,并在部署应用程序之前,装载进 myRIO 板载硬盘文件夹/home/lvuser/下。程序运行时,myRIO 首先读取指定路径下的文件/home/lvuser/RefreshToken.txt,获取当前用户装载的 Refresh Token。然后访问鉴权服务器,利用 Refresh Token 获取最新 Access Token,最后利用最新的 Access Token 参数访问 TLink 中 HTTP 设备,将采集数据上传至物联网平台。

(2) 采集数据。在第二帧主程序帧中,采取产生 0～100 的随机数及调用函数节点"转换为无符号单字节整型"(函数→编程→数值→转换→转换为无符号单字节整型)的方式模拟整数型数据采集过程。

(3) 生成 HTTP 请求体。在 HTTP 设备刷新设备数据的 API 接口中,Body 参数为 JSON 格式字符串,因此设计自定义函数节点 CreateBody.vi,将采集的 2 个整数数据封装为 API 接口中定义的 JSON 格式请求数据(为了简化程序设计,这里采取字符串连接的方式,封装 HTTP 请求 Body 参数),对应的自定义函数节点 CreateBody.vi 程序实现如图 5-65 所示。

(4) POST 请求。调用函数节点"POST"(函数→数据通信→协议→HTTP 客户端→POST),设置输入端口"缓冲区"为(3)中自定义子 VI 生成的请求数据,设置输入端口 url 为 TLink 中 HTTP 设备 API 接口地址,实现采集数据的云端发布功能。

(5) 采集数据本地显示。为了显示采集数据数值,设置滑动条显示 2 路采集参数,并将 2 路采集数据借助节点"捆绑"(函数→编程→簇、类与变体→捆绑)封装为簇类型数据,作为"波形图表"控件的显示内容,从而实现 2 路采集数据波形的实时显示。

(6) 定时控制。为了便于观测,While 循环中添加节点"等待"(函数→编程→定时→等待),设置等待参数为 3000ms,实现每 3 秒采集一次数据,并向服务器端上报一次数据的基本功能(物联网平台为了避免恶意上报导致网络阻塞影响用户使用体验,禁止高速刷新设备数据)。

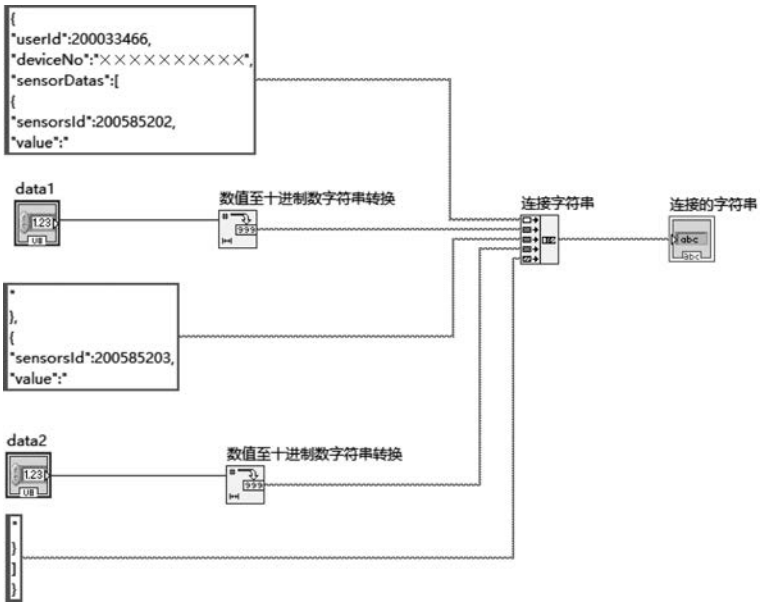

图 5-65　自定义函数节点 CreateBody. vi 程序实现

(7) 设备重置。在程序第三帧中调用函数节点“关闭句柄”(函数→数据通信→协议→HTTP 客户端→关闭句柄),释放 HTTP 通信程序所占用的资源,调用函数节点 Reset myRIO(函数→myRIO→Device management→Reset)完成 myRIO 开发平台的复位工作。

myRIO 采集数据并上报云端 HTTP 设备的完整程序实现如图 5-66 所示。

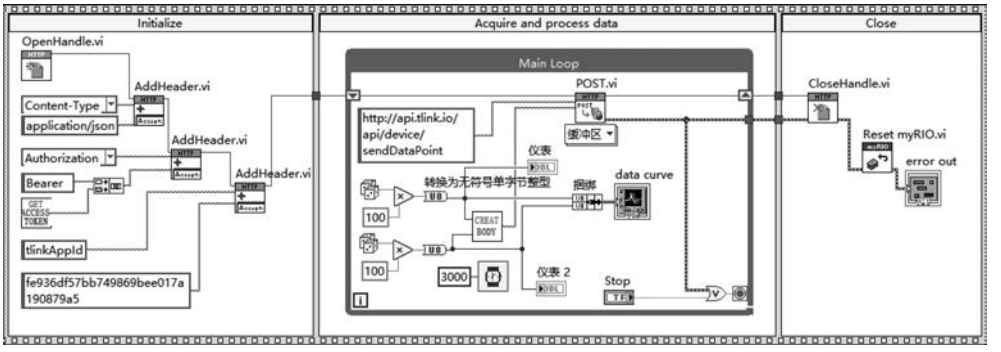

图 5-66　myRIO 采集数据并上报云端 HTTP 设备的完整程序实现

运行程序,myRIO 端基于 HTTP 的数据采集并上报云端设备程序执行结果如图 5-67 所示。

此时登录 TLink 物联网平台,打开监控中心,可观测到对应的 HTTP 设备名称已经由默认的灰色转变为黑色(表示基于 myRIO 的实体设备与物联网平台虚拟 HTTP 设备已经连接),对应的传感器连接状态指示显示“已连接”,云端 HTTP 设备数据接收结果与 myRIO 采集数据完全一致,如图 5-68 所示。

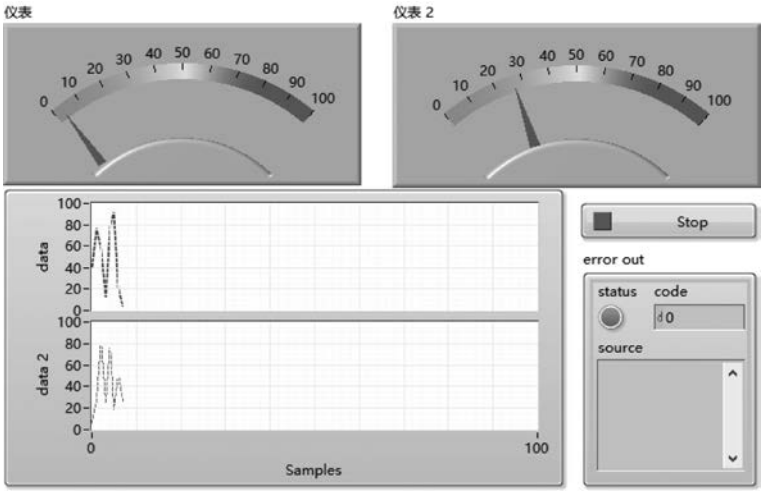

图 5-67　myRIO 端基于 HTTP 的数据采集并上报云端设备程序执行结果

图 5-68　云端 HTTP 设备数据接收结果

4. 基于 MQTT 协议的采集数据发布物联网平台

MQTT(Message Queuing Telemetry Transport,MQTT)是由 IBM 公司主导开发的一种发布/订阅范式的物联网即时通信协议。MQTT 协议基于 TCP 实现,运行在 TCP 长连接的基础上,可为网络设备提供低开销、低带宽的有序、可靠、双向连接的网络连接保障,在物联网领域具有重要的地位。

MQTT 协议的程序设计有两种实现途径。一种是借助 MQTT 工具包实现(相对简单,需要熟悉 MQTT 协议),另一种是基于 TCP 通信自行封装 MQTT 报文(还需要进一步熟悉 MQTT 协议的 14 种数据帧格式)实现 MQTT 协议通信。本案例使用 NI 提供的 MQTT 工具包实现。

MQTT 工具包提供的函数节点如图 5-69 所示。

物联网终端向物联网平台发布其采集数据的功能实现主要使用以下 5 类函数节点。

(1) api_mqttInit.vi,用于 MQTT 通信初始化设置,设置 MQTT 通信服务器 IP、端口并设置 MQTT 协议通信需要的用户名、密码等参数。

(2) api_mqttConnect.vi,用于连接 MQTT 服务器。

(3) api_mqttPublish.vi,用于向 MQTT 服务器发布数据。

图 5-69 MQTT 工具包提供的函数节点

(4) api_mqttDisconnect. vi,用于断开 MQTT 服务器连接。

(5) api_mqttFree. vi,用于释放 MQTT 通信占有的资源。

基于 MQTT 协议的采集数据上传物联网平台功能开发,首先需要完成物联网平台云端 MQTT 设备创建。

登录后进入控制台。控制台左侧导航栏选择“设备管理→添加设备”,打开 TLink 物联网平台云端设备创建页面,完成云端 MQTT 设备的创建。然后在控制台左侧导航栏选择“设备管理→设备列表”,选择创建的 MQTT 设备,单击所选设备行操作链接“设备连接”,可观测到云端 MQTT 设备通信连接相关信息如图 5-70 所示。

图 5-70 云端 MQTT 设备通信连接相关信息

MQTT 设备的创建意味着各类终端可以通过 MQTT 协议实现采集数据上报至 TLink 物联网平台的基本功能。物联网终端采集的数据上报云端 MQTT 设备,需要使用 MQTT 服务器 IP 地址(mq. tlink. io)、监听端口(1883)、MQTT 设备序列号 3 个通信连接参数。

单击物联网平台 MQTT 设备管理页面“生成示例”按钮,即可弹出窗口显示物联网终

端上报物联网平台云端 MQTT 设备的数据报文组成结构样例，如图 5-71 所示。

图 5-71　上报物联网平台云端 MQTT 设备的数据报文组成结构样例

从图 5-71 可见，物联网终端向物联网平台 MQTT 设备发布的消息为 JSON 格式字符串，具体内容与生成协议时是否勾选“ID”“读写标识”“上传时间”等选项有关。

完成上述准备工作，即可开始基于 MQTT 的 myRIO 采集数据云端发布应用程序设计。

1）设计目标

本案例设计的目标是 myRIO 作为物联网数据采集终端，遵循 TLink 物联网平台中创建的云端 MQTT 设备上传数据报文结构，将采集的数据封装为云端 MQTT 设备可处理的消息，实现 MQTT 协议的消息发布功能，奠定多终端共享数据的基础。

2）设计思路

利用 myRIO 新建项目自动生成的程序模板，保留其三帧程序基本结构。第一帧为初始化帧，创建 MQTT 连接的引用，设置 MQTT 连接参数；第二帧为主程序帧，在该帧 While 循环结构内，按照指定的时间间隔，以产生随机数的方式模拟 2 路实时数据采集和数据显示，并将采集的数据封装为 JSON 格式的物联网平台云端 MQTT 设备可处理的数据报文，完成 MQTT 协议消息发布；第三帧为程序后处理帧，主程序结束后断开 MQTT 连接，并实现 myRIO 程序运行结束后的设备重置工作。

3）程序实现

myRIO 程序实现可分解为初始化、采集数据、生成发布消息、完成消息发布、本地消息显示、定时控制、设备重置等步骤。

（1）初始化。调用函数节点 api_mqttInit.vi 创建 MQTT 连接的引用；调用函数节点 api_mqttConnect.vi 建立 myRIO 和 TLink 物联网平台中 MQTT 设备服务器之间的连接（IP 地址：mq.tlink.io；端口：1883）。

(2) 采集数据。采取 0～200 的随机数产生的方式模拟浮点型 2 个传感器数据采集过程。

(3) 生成发布消息。调用函数节点“格式化写入字符串”(函数→编程→字符串→格式化写入字符串),将采集的 2 个数据分别转换为带 1 位小数位的浮点数字符串、整数字符串;调用函数节点“连接字符串”(函数→编程→字符串→连接字符串),将采集的数据封装为 JSON 格式 MQTT 设备发布消息。

特别需要指出的是,按照 TLink 物联网平台规定,向 MQTT 设备发布消息,需要指定消息对应的 Topic,TLink 物联网平台中 MQTT 设备的 Topic 实际上就是该设备的序列号。如果发布消息仅指定 MQTT 设备的某一传感器值,则 Topic 为“设备序列号/传感器 ID”格式的字符串。

(4) 完成消息发布。调用函数节点 api_mqttPublish.vi,设置该节点输入参数 Topic 为 TLink 中创建的 MQTT 设备序列号;设置该节点输入参数 value 为上一步封装的 JSON 格式 MQTT 设备数据链接协议对应的字符串,实现采集数据的发布功能;该节点输出参数 success 连接布尔类型圆形指示灯、连接 myRIO 函数选板中提供的 Express VI LED(函数→myRIO→LED)(仅勾选 LED3),实现消息发布成功后,程序界面显示提醒及 myRIO 板载第 4 颗 LED 发光提醒功能。

(5) 本地消息显示。为了显示采集数据数值,设置滑动条显示 2 路采集参数,并将 2 路采集数据借助节点“捆绑”(函数→编程→簇、类与变体→捆绑)封装为簇类型数据,作为“波形图表”控件的显示内容,从而实现 2 路采集数据波形的实时显示。

(6) 定时控制。为了便于观测,向 While 循环中添加节点“等待”(函数→编程→定时→等待),设置等待参数为 2000ms,实现每 2 秒采集一次数据,并向云端 MQTT 设备发布一次消息的基本功能。

(7) 设备重置。程序第三帧中调用函数节点 api_mqttDisconnect.vi 断开 MQTT 连接,调用函数节点 api_mqttFree.vi 释放占用资源,调用函数节点 Reset myRIO(函数→myRIO→Device management→Reset)完成 myRIO 开发平台的复位工作。

myRIO 采集数据并向云端 MQTT 设备发布消息的完整程序实现如图 5-72 所示。

运行程序,myRIO 端数据采集及 MQTT 消息发布结果如图 5-73 所示。

此时登录 TLink 物联网平台,打开监控中心,可观测到对应的 MQTT 设备名称已经由默认的灰色转变为黑色(表示 myRIO 实体设备与物联网平台云端虚拟 MQTT 设备已经连接),对应的传感器连接状态指示显示“已连接”,云端 MQTT 设备数据接收结果与 myRIO 采集数据完全一致,如图 5-74 所示。

5. 基于 MQTT 协议的物联网平台信息订阅


微课视频

MQTT 协议中消息订阅指的是客户端关注了 MQTT 服务器某一主题,一旦该主题有新消息发布,则订阅了该主题的客户端会接收到其订阅主题最新发布的消息。本案例使用 NI 提供的 MQTT 工具包进行 MQTT 服务器连接、消息订阅等功能的程序设计。工具包中用于消息订阅相关的函数节点如下所示。

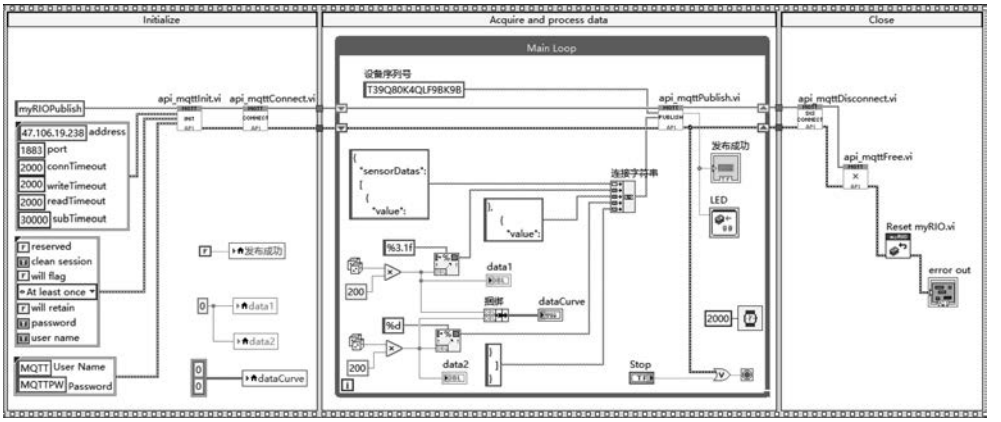

图 5-72　myRIO 采集数据并向云端 MQTT 设备发布消息

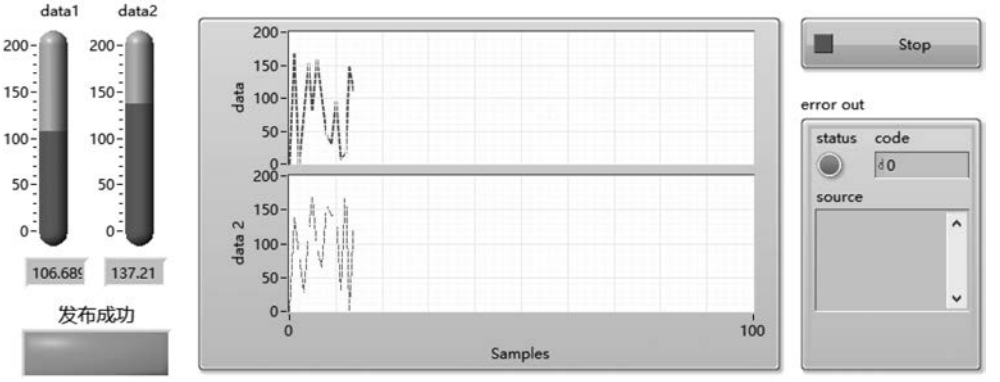

图 5-73　myRIO 端数据采集以及 MQTT 消息发布结果

图 5-74　云端 MQTT 设备数据接收结果

(1) api_mqttInit.vi,用于 MQTT 通信初始化设置,设置 MQTT 通信服务器 IP、端口并设置 MQTT 协议通信需要的用户名、密码等参数。

(2) api_mqttConnect.vi,用于连接 MQTT 服务器。

(3) api_mqttSubscribe.vi,用于向 MQTT 服务器订阅数据。

(4) api_mqttDisconnect.vi,用于断开 MQTT 服务器连接。

(5) api_mqttFree.vi,用于释放 MQTT 通信占用的资源。

另外,订阅主题消息还需要使用 2 个功能节点。

(1)“注册事件”,用于创建 MQTT 通信时的动态事件处理。

(2)“取消注册事件”,用于结束程序运行时销毁动态事件所占用资源。

基于 MQTT 协议的采集数据上传物联网云平台功能开发,首先需要完成物联网平台云端 MQTT 设备创建。具体方法与上一案例完全相同,读者可以根据项目需要模仿上一案例完成 TLink 物联网平台云端 MQTT 设备创建,这里不再赘述。

完成上述准备工作,即可开始基于 MQTT 的 myRIOmyRIO 物联网平台消息订阅应用程序设计。

1) 设计目标

本案例设计的目标是 myRIO 作为物联网控制终端,连接 TLink 物联网平台 MQTT 服务器,并订阅感兴趣的主题,实时获取该主题最新发布的消息,提取消息内容,当消息中传感器测量值大于设定的阈值时,myRIO 控制板载 LED 显示报警,实现根据订阅消息实施自动控制的功能模拟。

2) 设计思路

利用 myRIO 新建项目自动生成的程序模板,保留其三帧程序基本结构。第一帧为初始化帧,创建 MQTT 连接的引用,设置 MQTT 连接参数,完成订阅 MQTT 消息的动态事件注册;第二帧为主程序帧,在该帧 While 循环结构内,通过事件结构处理 MQTT 消息对应的动态事件,提取订阅 MQTT 消息中 Topic 和 value,其中 value 的取值为 JSON 格式的 MQTT 协议消息体,myRIO 解析并获取消息中需要的数据,并根据需要对提取的数据进行处理,根据处理结果控制板载 LED 亮灭;第三帧为程序后处理帧,主程序结束后断开 MQTT 连接,并实现 myRIO 程序运行结束后的设备重置工作。

3) 程序实现

myRIO 程序实现可分解为初始化、订阅消息获取与应用、订阅消息解析子 VI 设计、释放 MQT 资源与设备重置等步骤。

(1) 初始化。初始化部分由 2 个顺序帧组成,第一帧中对有关数据、控件的初始赋值,并且恢复板载 LED 显示状态,对应的程序实现如图 5-75 所示。

第二帧中调用函数节点 api_mqttInit.vi、节点“注册事件”、节点 api_mqttConnect.vi 及节点 api_mqttSubscribe.vi,完成 TLink 物联网平台中 MQTT 服务器的连接、消息订阅、订阅消息处理的动态事件注册(IP 地址:mq.tlink.io;端口:1883),如图 5-76 所示。

特别需要指出的是,按照 TLink 物联网平台规定,订阅 MQTT 设备消息,需要指定消

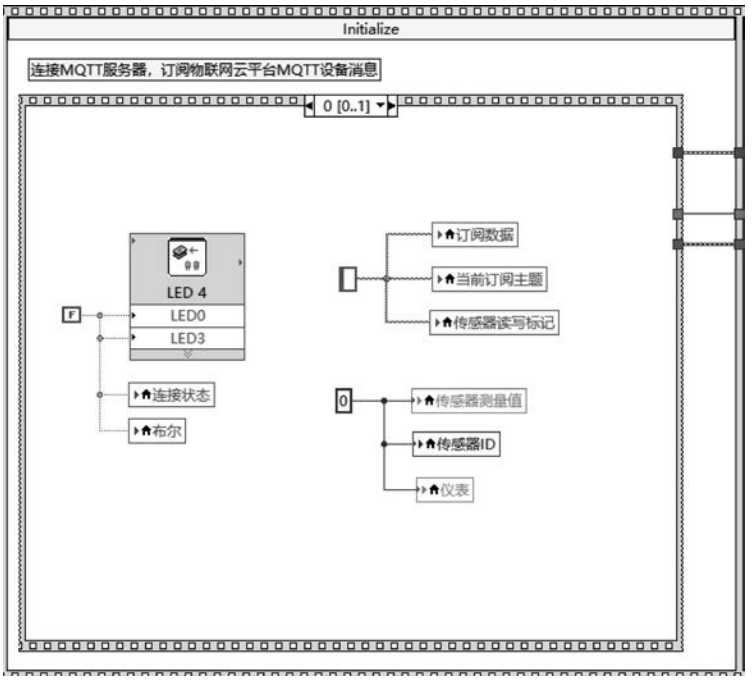

图 5-75　MQTT 订阅程序初始化部分第一帧程序框图

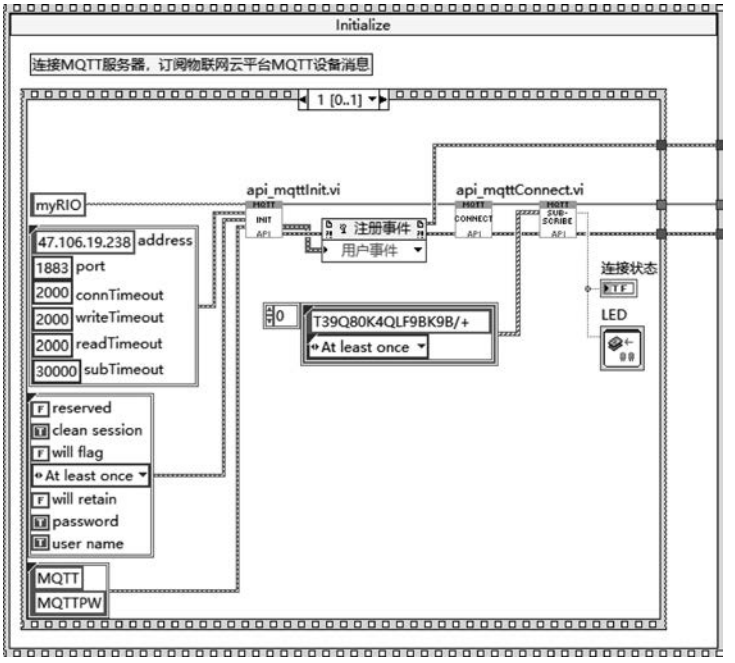

图 5-76　MQTT 订阅程序初始化部分第二帧程序框图

息对应的 Topic——TLink 中 MQTT 设备的 Topic 实际上就是该设备对应的序列号。Topic 为“设备序列号/传感器 ID”格式的字符串。本案例中订阅消息时设置订阅主题为“××××××××××/+”,“+”为通配符,表示序列号为“××××××××××”的 MQTT 设备所有传感器均处于客户端的订阅、监听状态。

(2) 订阅消息获取与应用。MQTT 协议订阅消息的获取,第二帧为主程序帧,为 While 循环结构下的应用程序核心业务功能开发,需要监听 MQTT 用户事件,获取用户事件中 Topic、value 等参数值。

MQTT 协议中接收到订阅消息需要通过主程序帧中 While 循环结构内的事件结构对于初始化帧中注册的多态事件监听实现。

右击 while 循环结构中的事件结构,选择“显示动态事件接线端”,实现事件结构中动态事件接线端的添加。动态事件接线端输入端口连接节点“注册事件”,输出端口连接节点“事件注册引用句柄”,此时右击事件结构“选择器标签”选择“添加事件分支”,在弹出对话框中选择“多态→MQTT < event >用户事件”→“确定”,完成订阅消息接收事件处理子框图的创建。

事件结构内动态事件处理程序子框图左侧的事件数据节点中可见 value、topic 等参数。如图 5-77 所示。

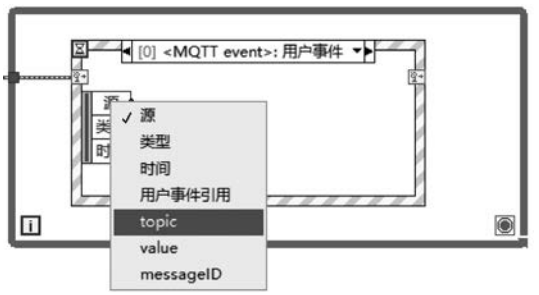

图 5-77 动态事件处理程序子框图左侧的事件数据节点

对于事件数据节点中提供的 topic、value 进行解析处理,可获得订阅消息的主题名称、消息体。对于消息体进行解析,获取消息体中包含的 2 个传感器最新测量值。如果传感器测量值大于指定阈值,则控制布尔圆形指示灯显示报警,并且控制板载第 4 颗 LED(LED3)显示报警,对应的 MQTT 订阅消息处理与显示相关程序实现如图 5-78 所示。

(3) 订阅消息解析子 VI 设计。步骤(2)中事件数据节点提供的参数“value”取值为 JSON 格式的 MQTT 服务器下行消息,因此设计子 VI 对 JSON 格式消息进行解析,获取消息体中包含的传感器 ID 、传感器读写标记、传感器测量值等信息,并根据给定的阈值输出越限报警信息,完整的订阅信息解析子 VI 程序实现如图 5-79 所示。

(4) 释放 MQT 资源与设备重置。在程序第三帧中调用函数节点 api_mqttDisconnect. vi 断开 MQTT 连接,调用函数节点“取消事件注册”停止动态事件的监听和处理,调用函数节点 api_ mqttFree. vi 释放占用资源,调用函数节点 Reset myRIO(函数→myRIO→Device

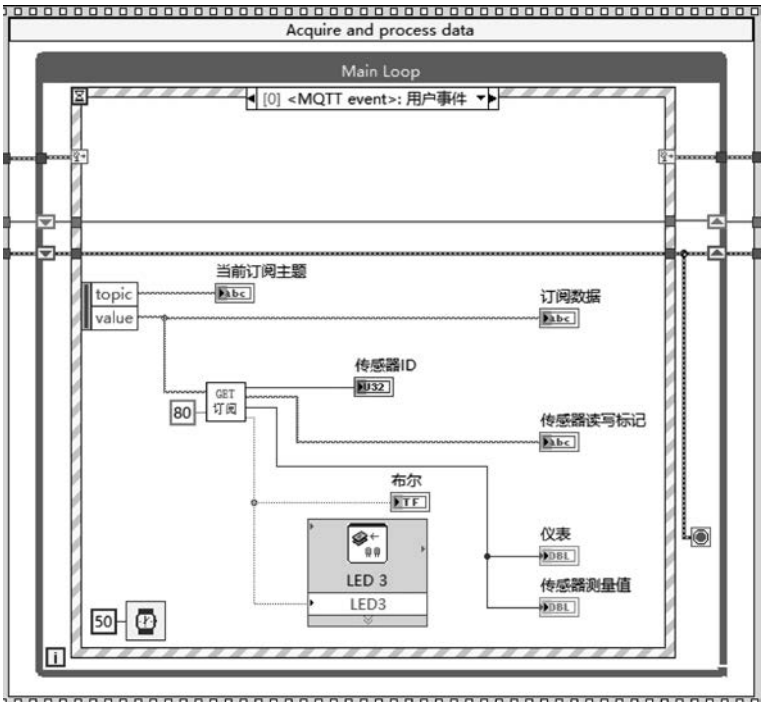

图 5-78　MQTT 订阅消息处理与显示相关程序实现

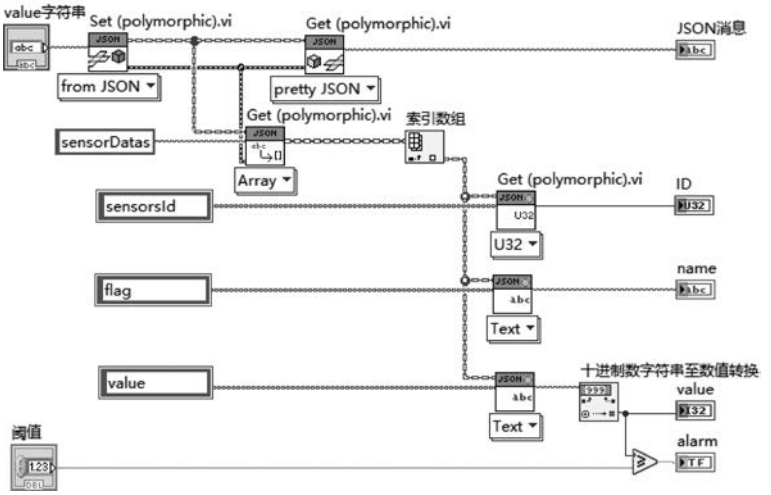

图 5-79　订阅信息解析子 VI 程序实现

management→Reset)完成 myRIO 开发平台的复位工作。对应的释放 MQT 资源与设备重置程序实现如图 5-80 所示。

myRIO 订阅云端 MQTT 设备消息的完整程序实现如图 5-81 所示。

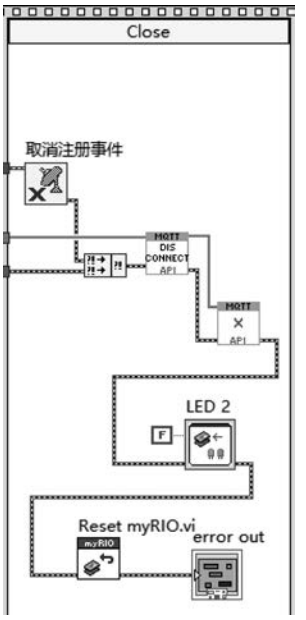

图 5-80　释放 MQT 资源与设备重置程序实现

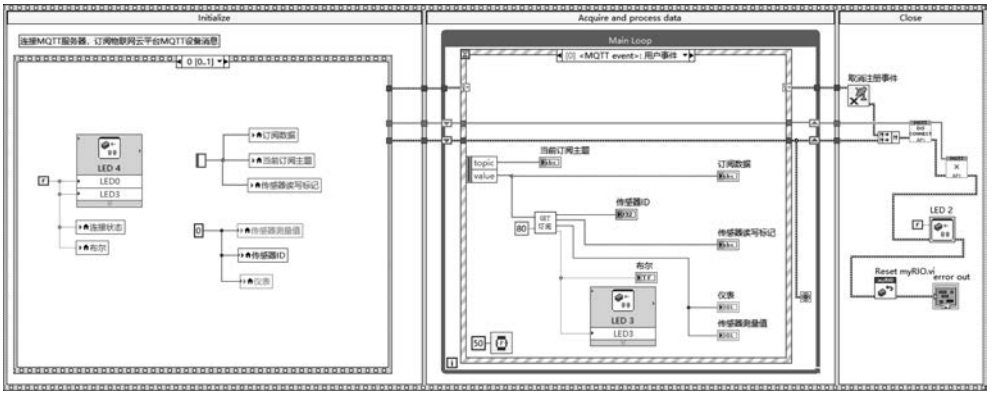

图 5-81　myRIO 订阅云端 MQTT 设备消息的完整程序实现

运行程序，myRIO 建立与 TLink 物联网平台 MQTT 服务器连接，并订阅指定主题的消息。打开 TLink 物联网平台，进入对应的 MQTT 设备链接协议页面，选择传感器，输入传感器最新取值 55，单击“写入指令”按钮，模拟 TLink 物联网平台中 MQTT 设备传感器消息发布，如图 5-82 所示。

此时打开 myRIO 应用程序界面，可见 myRIO 终端成功连接 MQTT 服务器，并且捕获到订阅主题的 JSON 格式消息及主题名称。程序对 JSON 格式消息进行解析，获取消息体中传感器 ID、测量值、读写标记等信息，根据设定的阈值判断是否报警。myRIO 订阅云端 MQTT 设备消息程序执行结果如图 5-83 所示。

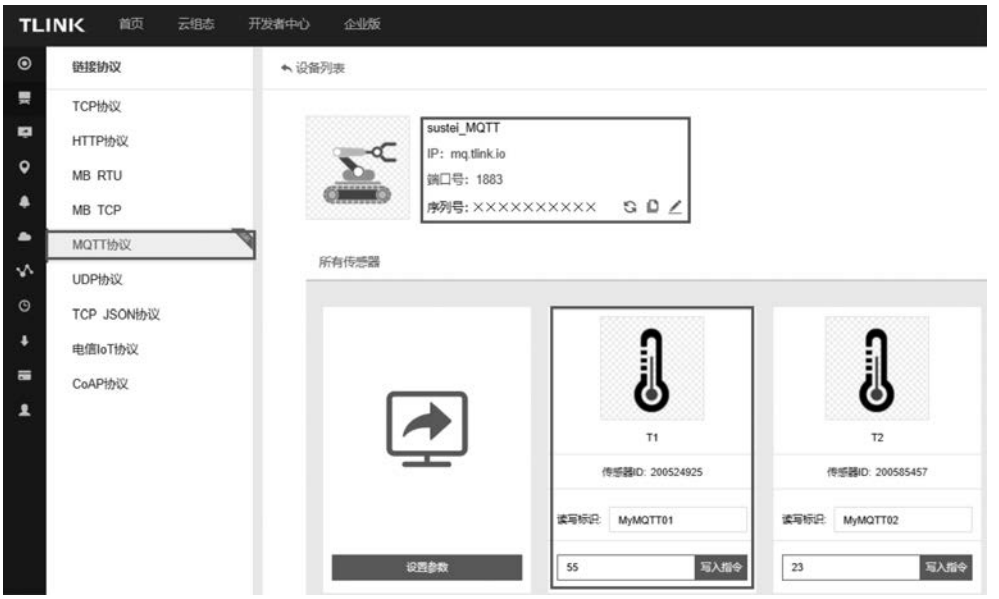

图 5-82　模拟 TLink 物联网平台中 MQTT 设备传感器消息发布

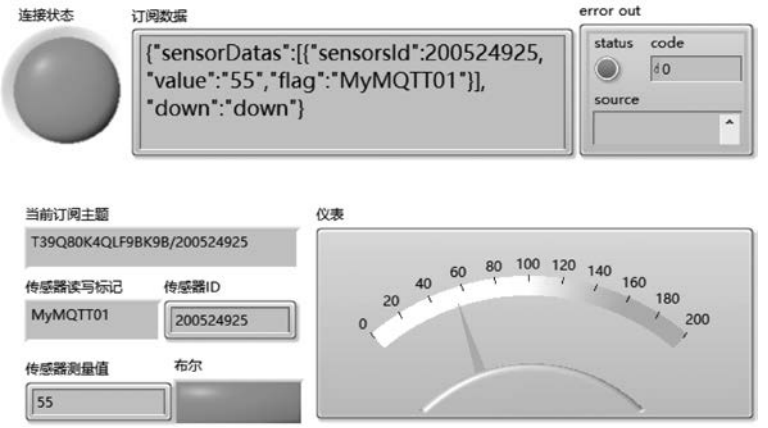

图 5-83　myRIO 订阅云端 MQTT 设备消息程序执行结果