

第 3 章



myRIO 模拟信号采集与输出

主要内容

- 模拟信号数据采集技术概述；
- myRIO 模拟信号采集与输出有关引脚分布；
- myRIO 工具包中模拟信号采集 ExpressVI、底层 VI 配置和使用方法；
- 模拟信号采集技术相关工程项目开发实战；
- myRIO 工具包中模拟信号输出 ExpressVI、底层 VI 配置和使用方法；
- 模拟信号输出技术相关工程项目开发实战。

3.1 模拟信号采集技术及应用

本节在模拟信号数据采集基本原理介绍的基础上,给出 myRIO 模拟输入端口资源配置,模拟信号采集的 ExpressVI 及其调用方法,模拟信号采集相关的若干底层 VI 及其应用的一般流程,并结合电压采集实例介绍实现模拟信号数据采集程序编写的基本方法。

3.1.1 模拟信号采集概述

模拟信号采集(Analog Input, AI)是指对于连续变化的物理量所表示的信号进行数据采集。这里所谓的连续,既指时间上的连续性,也指数值上的连续性。模拟信号一般指的是连续的电信号。常见的模拟信号采集多为电压信号的采集,当然模拟信号还包括温度、压力、流量、位移、电阻、频率、位置、声音、加速度等物理信号经传感器变换所得的电信号。一般情况下,物理信号经传感器变换所得的电信号需要经过信号调理才能进行数据采集。

根据传感器输出信号的特征不同,测量系统接入模拟信号有单端接入和差分接入两种方式。

单端接入方式是指传感器信号线连接测量系统有关 AI 端口,传感器地线与测量系统共地连线。这种连接方式简单易用,但是存在容易引入共模干扰的问题。单端接入方式适合传感器输出信号电平比较高、信号传输线比较短且存在屏蔽或信号可以共享公共参考点的场景。

差分接入方式是指传感器信号的+、-两极分别连接测量系统的两个不同的通道。差

分接入是一种比较理想的接线方式,可以有效抑制接地回路产生的误差,还可以在一定程度上抑制传输线拾取的环境噪声,特别适合传感器输出电平比较低(如小于 1V)、信号传输线路长、环境噪声比较大的场景。但是差分接入方式占用的 I/O 通道数量是单端接入方式的 2 倍,这是其无法克服的缺点。

模拟信号采集除了上述连接模式需要考虑,还必须在项目实施前考虑信号输入范围(输入信号的最大值和最小值)、采样速率(数据采集的时间间隔,需满足 Nyquist 定理要求)、精度(AD 转换的位数决定)等因素。

myRIO 提供了 8 个单端模拟输入通道、2 个差分模拟输入通道,总计 10 个模拟信号输入通道,其中 A 口提供 4 个单端模拟输入通道,分别是 AI0、AI1、AI2、AI3,对应 Pin3、Pin5、Pin7、Pin9; B 口同样提供 4 个单端模拟输入通道,其引脚编号与 A 口一致。A、B 口提供的模拟输入通道引脚分布如图 3-1 所示。

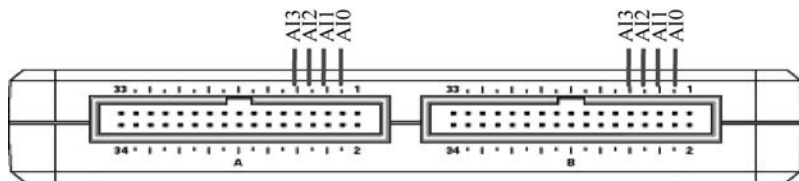


图 3-1 A、B 口提供的模拟输入通道引脚分布

C 口提供 2 个模拟输入通道,属于差分输入通道,为 AI0+ (Pin7)、AI0- (Pin8)、AI1+ (Pin9)、AI1- (Pin10),C 口提供的模拟输入通道引脚分布如图 3-2 所示。

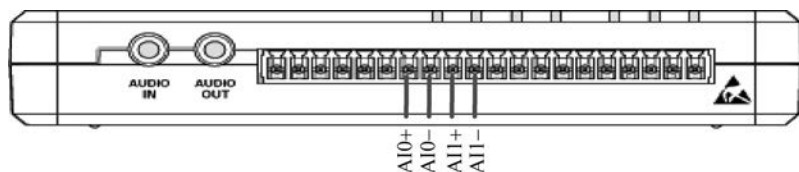


图 3-2 C 口提供的模拟输入通道引脚分布

myRIO 中的模拟输入一般为 0~5V 单端电压类型数据,连接传感器时需要注意传感器输出电压范围,必要时设计针对连接传感器输出特性的调理变换电路,使之适应 myRIO 模拟输入的要求。

3.1.2 主要函数节点

myRIO 中模拟信号采集的实现有两种方法。第一种方法是使用 myRIO 工具包中提供的模拟输入 ExpressVI,如图 3-3 所示。

将该节点拖曳至程序框图,即可弹出模拟输入 Express VI 配置窗口,如图 3-4 所示。

配置操作完成后,程序框图中的模拟输入 Express VI 节点图标如图 3-5 所示。

按照当前配置,该节点每次调用可以读取 A 口 AI0 模拟输入通道一个数据。配置多个模拟输入通道,则可以同时读取多个模拟信号数值。

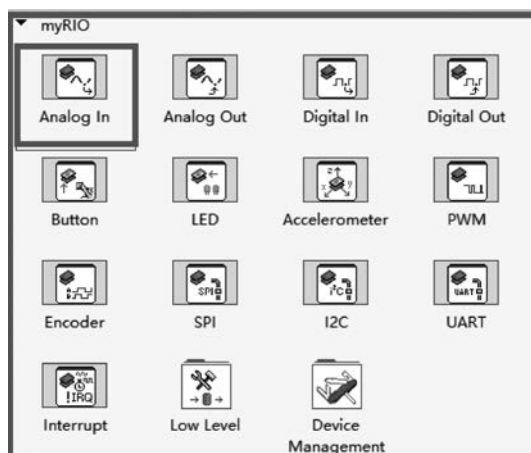


图 3-3 myRIO 工具包中提供的模拟输入 ExpressVI

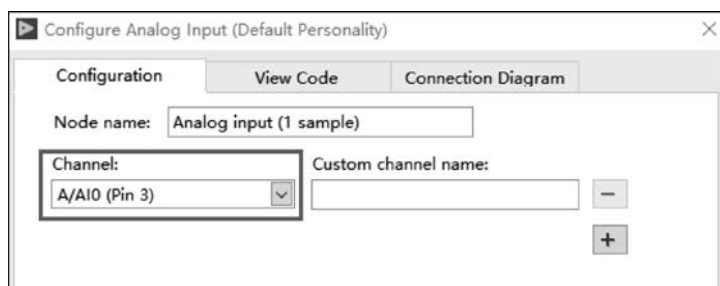


图 3-4 模拟输入 Express VI 配置窗口

第二种方法是基于 myRIO 提供的底层函数(myRIO→Low Level→Analog Input 1 Sample)实现模拟信号采集,底层函数子选板中的模拟输入相关 VI 如图 3-6 所示。

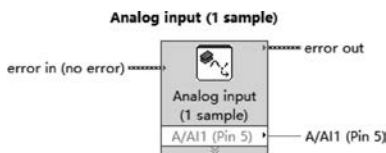


图 3-5 模拟输入 Express VI 节点图标

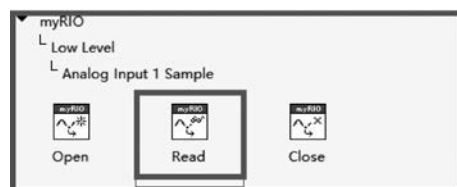


图 3-6 底层函数子选板中的模拟输入 VI

基于底层函数进行模拟信号采集时,基本流程为“打开 AI 通道(Open)→AI 通道读取(Read)→关闭 AI 通道(Close)”。如果需要连续读取 AI 通道数据,则将 AI 通道读取操作(Read)置于 While 循环结构之中即可。

3.1.3 模拟信号采集技术应用实例

模拟信号采集技术是电子系统感知外部世界连续变化数据或者状态的重要手段,具有极其重要的实践价值。



微课视频

旋转电位器是一种阻值可调的电子元件,由一个电阻体和一个转动或滑动系统组成。当电阻体的两个固定触点之间外加一个电压时,通过转动系统改变触点在电阻体上的位置,在动触点与固定触点之间便可得到一个与动触点位置成一定关系的电压。旋转电位器主要用于通信产品、对讲机、汽车功放、多媒体音响、智能家居、计算机周边等,其功能主要为音量调节、光线强弱调节、速度调节、温度调节等。

1) 设计目标

本节案例使用旋转电位器形成分压电路,产生跟随电位器旋转角度变化的模拟电压值,进而通过 myRIO 模拟信号采集端口获取对应的电压值,并根据电压值范围驱动板载不同 LED 灯显示。

2) 硬件连线

旋转电位器 A、B、C 三个接线端分别连接 myRIO 开发平台 A 口 Pin1(+5V)、Pin3(AI0)、Pin6(AGND),实现旋转电位器分压电路信号采集功能,基于旋转电位器的分压信号采集硬件连线如图 3-7 所示。

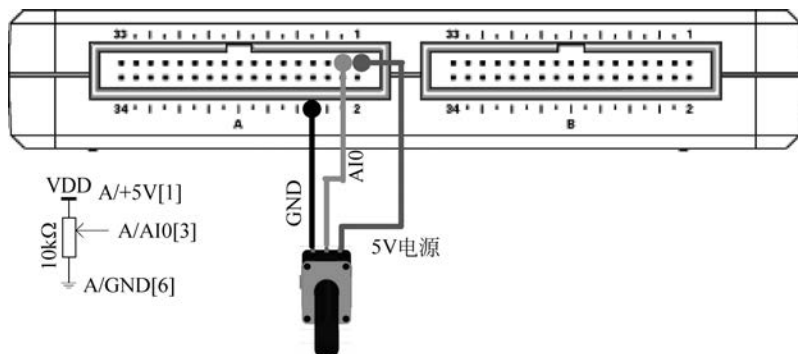


图 3-7 基于旋转电位器的分压信号采集硬件连线

3) 设计思路

利用 myRIO 新建项目的程序模板,在 While 循环结构内,采集 A 口 AI0(Pin3)数据,即旋转电位器分压所产生的模拟电压值。由于旋转电位器从一个极端转至另一个极端时,电压值相应地从 0V 连续变化至 5V,所以可以将采集的电压值设置为 4 个电压值区间,当采集的电压值处于 $[0, 1.25)$ 时,驱动 LED0 显示;当采集的电压值处于 $[1.25, 2.5)$ 时,驱动 LED1 显示,当采集的电压值处于 $[2.5, 3.75)$ 时,驱动 LED2 显示,当采集的电压值处于 $[3.75, 5]$ 时,驱动 LED3 显示。

4) 程序实现

程序实现可分解为程序总体结构设计、旋转电位器电压检测、板载 LED 驱动显示、myRIO 重置等功能设计。

(1) 程序总体结构设计。利用 myRIO 项目模板自动生成的 3 帧程序结构,删除三轴加速度计数据采集相关函数节点。第一帧保持不变;第二帧 While 循环中采集旋转电位器电压值,借助条件结构进行采集数据所在区间判断,并驱动板载 LED 显示;第三帧重置 myRIO。

(2) 旋转电位器电压检测。调用 ExpressVI Analog In(函数→myRIO→Analog In), 错误输入连接第一帧簇常量; 由于 A 口 Pin3(AI0)连接外部信号, 所以配置模拟信号采集通道如图 3-8 所示。

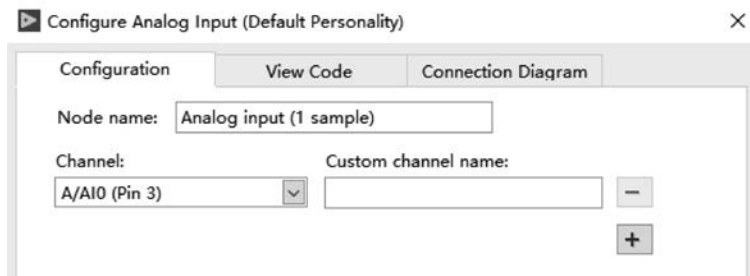


图 3-8 配置模拟信号采集通道

配置完毕后, 右击 ExpressVI Analog In 图标, 选择“图标样式→显示为子 VI”, 完成模拟信号采集函数节点配置和调用, 实现旋转电位器分压值的检测。

(3) 板载 LED 驱动显示。由于旋转电位器分压结果为 0~5V, 根据检测值将其进行四分段处理, 即分压值处于 $[0, 1.25)$ 、 $[1.25, 2.5)$ 、 $[2.5, 3.75)$ 、 $[3.75, 5]$ 某个区间时, 驱动对应的板载 LED0、LED1、LED2、LED3 分别显示。

调用函数节点“判定范围并强制转换”(函数→编程→比较→判定范围并强制转换)。其输入参数 x 连接模拟信号采集结果; 其输入参数“上限”设置为常量 1.25, 其输入参数“下限”设置为常量 0, 完成采集结果是否属于区间 $[0, 1.25)$ 的判断。同样方法, 完成采集数据是否属于另外三个区间的判断。

调用条件结构判断上一步采集数据区间判定结果。当判定结果为真时, 对应的条件结构“真分支”内调用 ExpressVI“LED”(函数→myRIO→LED), 配置板载 LED 控制对象为 LED0, 如图 3-9 所示。

为了进一步验证程序功能, 程序前面板中添加 4 个布尔类型显示控件“圆形指示灯”(函数→编程→布尔→圆形指示灯), 与板载 LED 功能对应一致。当板载 LED0 亮时, 前面板对应的“圆形指示灯”同时取值为真。

条件结构“假分支”内调用 ExpressVI LED, 设置常量“逻辑假”熄灭板载 LED0, 同时驱动前面板对应的“圆形指示灯”取值为假。

其他 3 种情况处置办法与此相同, 这里不再赘述。

(4) myRIO 重置。调用函数节点 Reset myRIO(函数→myRIO→Device management→Reset)实现程序结束后的 myRIO 设备复位功能。

最终完成的程序中, 当检测到的分压值满足某一区间判定条件时, 驱动对应的板载 LED 发光, 对应的模拟信号采集与 LED 发光驱动程序实现如图 3-10 所示。

当检测到的分压值不满足某一区间判定条件时, 驱动板载 LED 熄灭, 对应的模拟信号采集与 LED 熄灭驱动程序实现如图 3-11 所示。

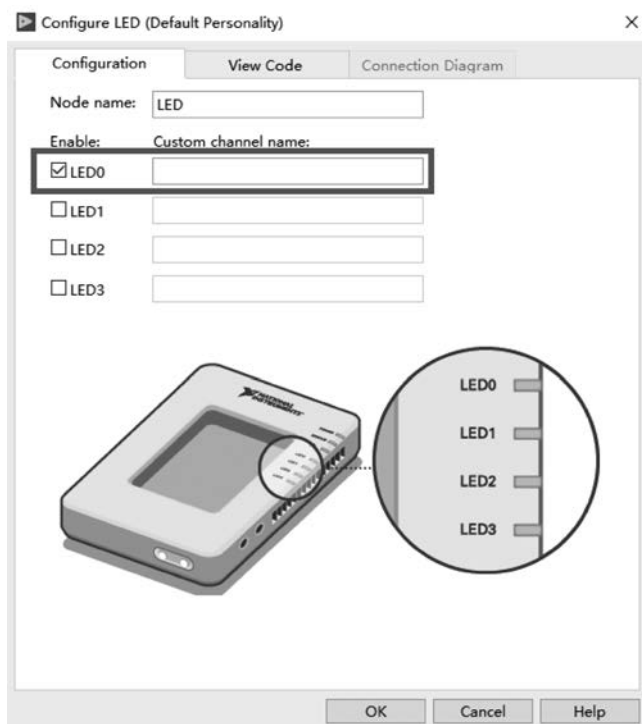


图 3-9 配置板载 LED 控制对象为 LED0

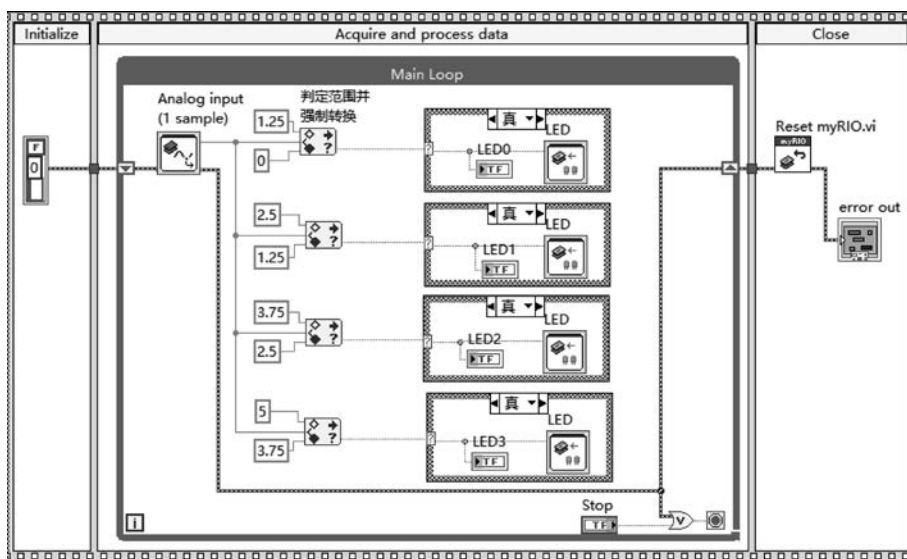


图 3-10 模拟信号采集与 LED 发光驱动程序实现

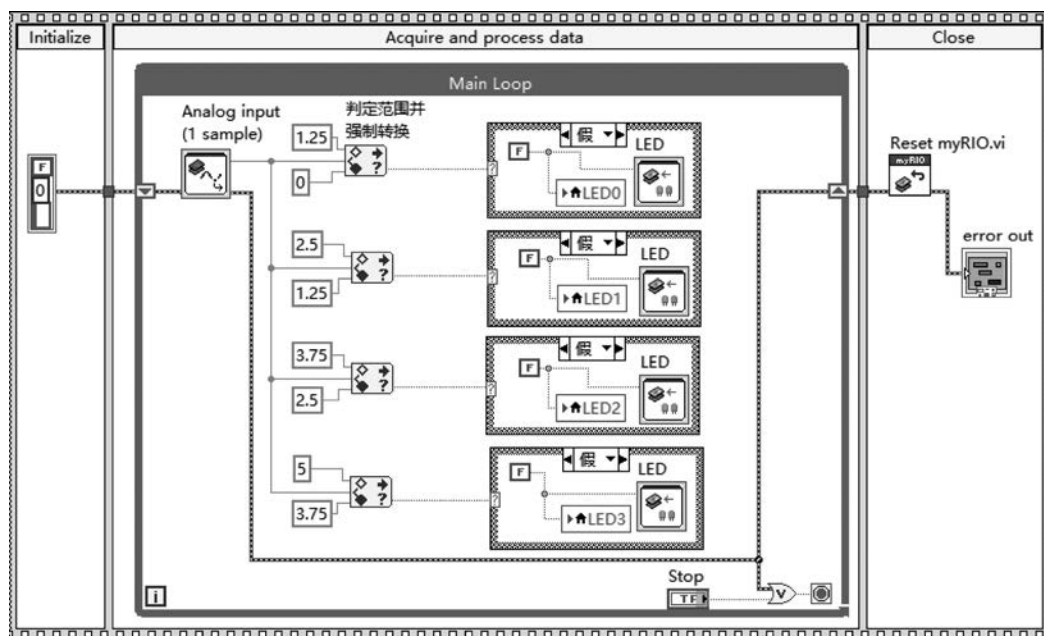


图 3-11 模拟信号采集与 LED 熄灭驱动程序实现

同样功能,亦可采用底层函数实现,基于底层 VI 的模拟信号采集与板载 LED 驱动程序实现如图 3-12 所示。

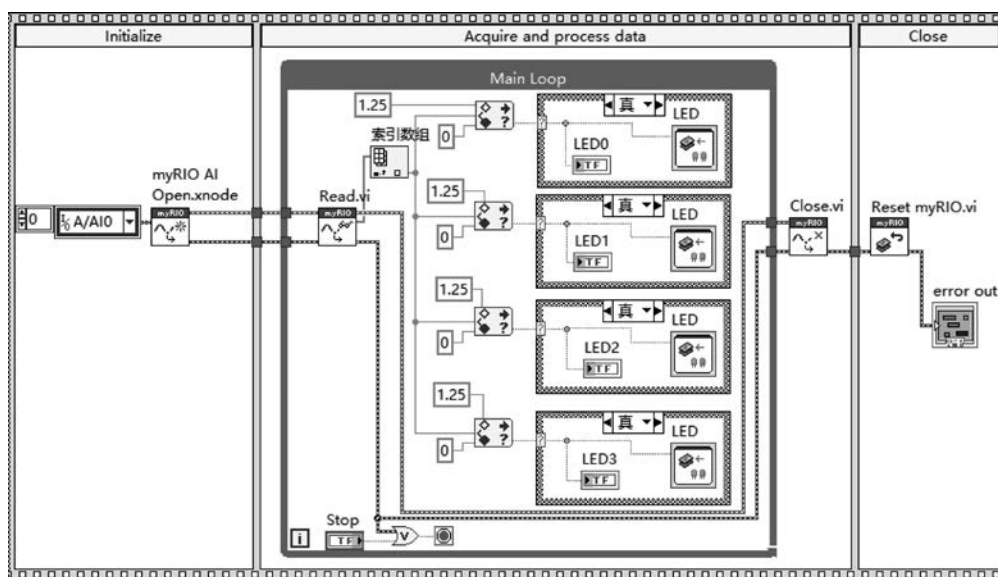


图 3-12 基于底层 VI 的模拟信号采集与板载 LED 驱动程序实现

3.2 模拟信号输出技术及应用

本节在模拟信号输出概念介绍的基础上,介绍 myRIO 模拟信号输出通道资源配置情况,模拟信号输出的 ExpressVI 及其调用方法,模拟信号输出相关的若干底层 VI 及其应用一般流程,并结合 LED 发光亮度连续调节实例介绍实现模拟信号输出程序编写的基本方法。

3.2.1 模拟信号输出技术概述

模拟信号输出(Analog Output,AO)可视为模拟信号采集的逆过程。模拟信号采集是获取外部模拟量数据,经采集装置 A/D 模块转换为计算机/嵌入式可识别的数字信号。而模拟信号输出则是指由计算机/嵌入式产生的数字信号经 D/A 转换成为模拟信号,经由模拟输出通道向外输出。

myRIO 提供了 4 个单端模拟输出通道和 2 个对地参考模拟输出通道。其中 A 口提供 2 个单端模拟输出通道,分别是 AO0(Pin2)、AO1(Pin4);B 口同样提供 2 个单端模拟输出通道,其引脚编号与 A 口一致。A、B 口模拟输出通道引脚分布如图 3-13 所示。

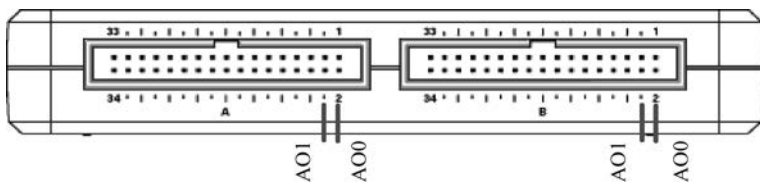


图 3-13 A、B 口模拟输出通道引脚分布

C 口提供 2 个对地参考模拟输出通道,为 AO0(Pin4)、AO1(Pin5),C 口模拟输出通道引脚分布如图 3-14 所示。

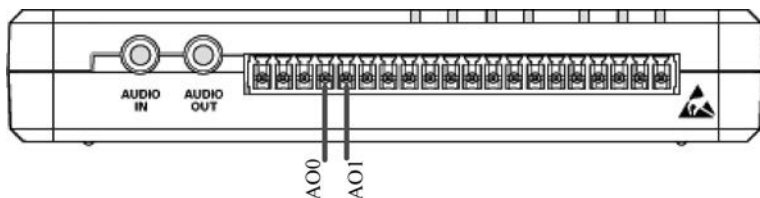


图 3-14 C 口模拟输出通道引脚分布

3.2.2 主要函数节点

myRIO 中模拟信号输出的实现有 2 种方法。第一种方法是使用 myRIO 工具包中的模拟输出 Express VI,如图 3-15 所示。

将该节点拖曳至程序框图,即可弹出模拟输出 ExpressVI 参数配置窗口,如图 3-16 所示。

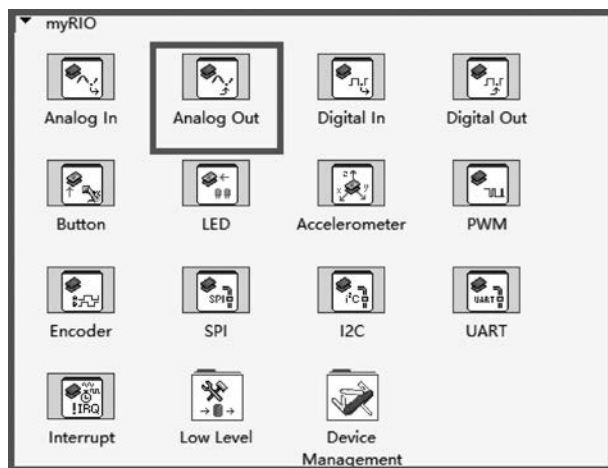


图 3-15 myRIO 工具包中的模拟输出 Express VI

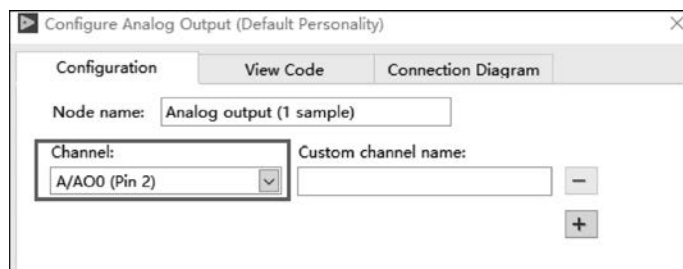


图 3-16 模拟输出 ExpressVI 参数配置窗口

配置操作完成后,程序框图中的模拟输出 ExpressVI 节点图标如图 3-17 所示。

按照当前配置,该节点每次调用可以向 A 口 AO0 输出一个数据。配置多个端口,则可以同时写出多个模拟量数值。

第二种方法是基于 myRIO 提供的底层函数(myRIO→Low Level→Analog Output 1 Sample)实现模拟信号输出,底层函数子选板中的模拟输出相关节点如图 3-18 所示。

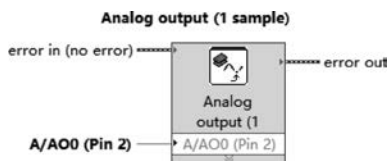


图 3-17 模拟输出 ExpressVI 节点图标

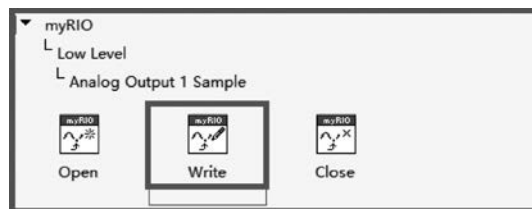


图 3-18 底层函数子选板中的模拟输出节点

基于底层函数进行模拟信号输出时,基本流程为“打开 AO 通道(Open)→AO 通道写入(Write)→关闭 AO 通道(Close)”。如果需要连续输出 AO 通道,则将 AO 通道写入操作(Write)置于 While 循环结构中即可。



微课视频

3.2.3 模拟信号输出技术应用实例

模拟输出通道提供的信号经常作为电子系统提供激励信号,或者为自动化系统提供控制信号,具有很重要的实践价值。

1) 设计目标

本节案例使用旋转电位器形成分压电路,产生跟随电位计旋转角度变化的模拟电压值,进而通过 myRIO 中 AI 通道采集对应的电压值,并将采集的电压值通过 AO 通道输出,用以实现人工可控的模拟电压调节 LED 发光亮度。

2) 硬件连线

旋转电位器 A、B、C 三个接线端分别连接 myRIO 开发平台 A 口 Pin1(+5V)、Pin3(AI0)、Pin6(AGND)。LED 阳极连接 B 口 Pin2(AO0),阴极连接 B 口 Pin6(AGND)。模拟输出驱动 LED 显示的硬件连线如图 3-19 所示。

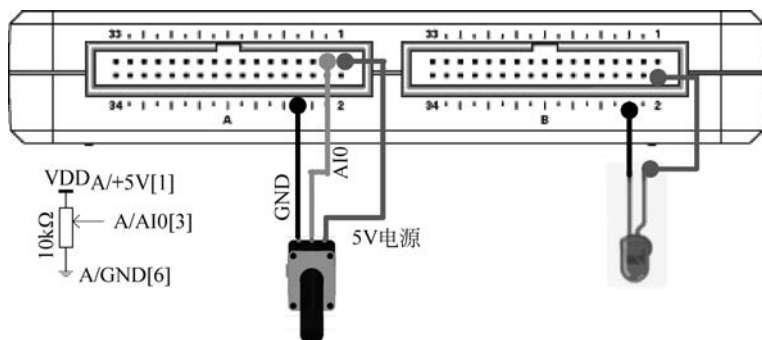


图 3-19 模拟输出驱动 LED 显示的硬件连线

3) 设计思路

利用 myRIO 新建项目的程序模板,在 While 循环结构内,采集 A 口 AI0(Pin3)连接的旋转电位器分压所产生数据。由于旋转电位器从一个极端转至另一个极端时,电压值相应地从 0V 连续变化至 5V,所以将采集的模拟电压值通过 AO0 端口输出,作为 LED 显示驱动电压,实现基于调节电位器产生连续变化的驱动电压,达到连续调节 LED 发光亮度的目的。

4) 程序实现

程序实现可分解为程序总体结构设计、旋转电位器电压检测、LED 驱动电压输出控制等步骤。

(1) 程序总体结构设计。利用 myRIO 项目模板自动生成的 3 帧程序结构,删除三轴加速度计数据采集相关函数节点。第一帧保持不变,传递初始状态无错误信息;第二帧 While 循环中采集旋转电位器电压值,向 AO0 通道输出采集结果实现 LED 亮度控制;第三帧重置 myRIO。

(2) 旋转电位器电压检测。调用 ExpressVI Analog In(函数→myRIO→Analog In),错

误输入连接第一帧簇常量；配置旋转电位器分压电路信号采集通道为 A 口 Pin3(AI0)，如图 3-20 所示。

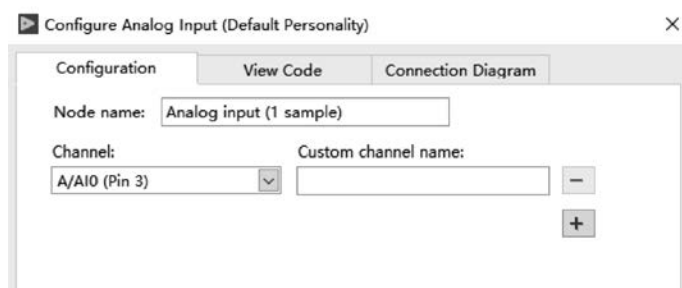


图 3-20 配置旋转电位器分压电路信号采集通道

(3) LED 驱动电压输出控制。LED 连接 B 口 AO0, 将采集的旋转电位器分压结果通过 ExpressVI Analog Out(函数→myRIO→Analog Out)输出, 控制 B 口 AO0 通道电压值, 达到调控 LED 发光亮度的目的。对应的 LED 发光亮度调控信号输出通道配置如图 3-21 所示。

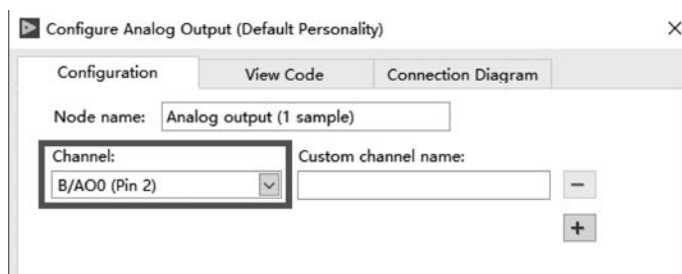


图 3-21 LED 发光亮度调控信号输出通道配置

基于模拟输出 Express VI 实现 LED 发光亮度调控的完整程序如图 3-22 所示。

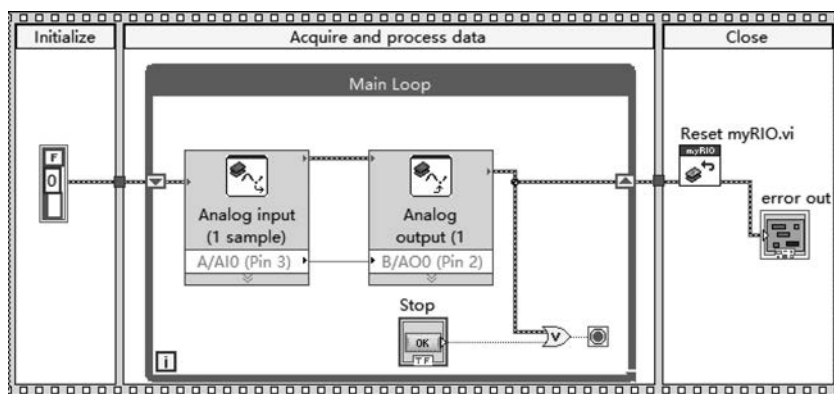


图 3-22 基于模拟输出 Express VI 实现 LED 发光亮度调控的完整程序

同样功能亦可采用底层函数实现,基于模拟输出底层 VI 实现 LED 发光亮度调控的完整程序如图 3-23 所示。

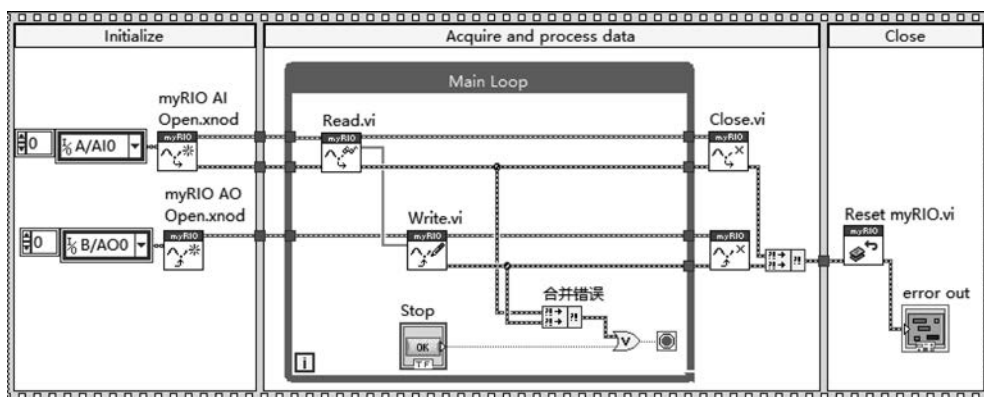


图 3-23 基于模拟输出底层 VI 实现 LED 发光亮度调控的完整程序