ZLG 致远电子

微文摘

ZLG MICRO DIGEST

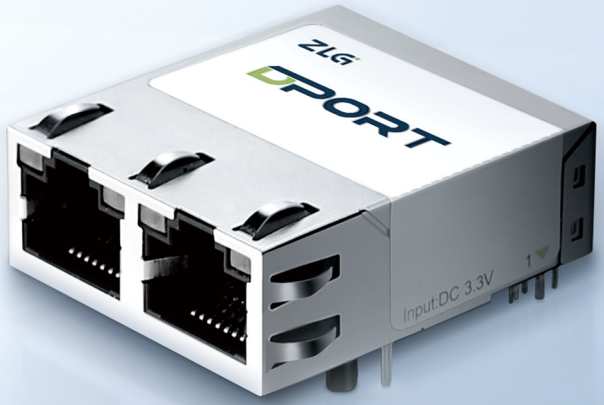
2023/12 第12期

月刊

ZLG 致远电子

国内首创集成式 EtherCAT 从站模块

DPort-ECT



为快速设计开发EtherCAT从站而生

高度集成　高实时性　高可靠性　小体积　易于使用

规格参数

产品型号	DPort-ECT
网口数量	2
主控接口	SPI
接口电平	3.3V
速率	10/100Mbps 自适应
供电电压	3.3V±3%
DC 抖动	≤ 15ns
工作温度	-40°C ~+85°C
存储温度	-40°C ~+85°C
相对湿度	5%~90%（非冷凝）
电磁兼容	静电放电抗扰度试验：4 级 雷击（浪涌）抗扰度试验：3 级 电快速瞬变群脉冲抗扰度试验：3 级 射频场感应的传导骚扰看扰度试验：10V
评估板	EPC6450-DP（不含 DPort 模块）

CONTENTS
目录

【技术分享】AWTK 开源智能串口屏方案

ZLG 致远电子 2023-12-01 11:33:31

AWTK 开源智能串口屏方案发布，旨在解决传统串口屏诸多痛点，为用户提供更开放、更易用、更强大的开源串口屏方案。

基于 AWTK 和 AWTK-MVVM 实现的串口屏方案。

界面修改数据，自动通知 MCU。

MCU 修改数据，自动更新界面。

主要特色

1. 开发

- 强大的界面设计器 AWStudio；
- 基于 AWTK 实现强大的 GUI 功能（多窗口、输入法、动画和各种控件）；
- 基于 AWTK-MVVM 实现低代码开发（编写绑定规则即可实现常见应用程序）；
- 支持在 PC 上模拟运行，并提供 MCU 模拟器模拟与串口屏的交互；
- 开发时支持通过串口更新 UI 资源，无需插拔 USB (TODO)；
- 开放源码，免费商用，从底层到应用程序全程可控；
- 轻松集成第三方开源库，比如 sqlite3 和各种网络协议。

2. 通信

- 支持串口；
- 支持 TCP；
- 开发者无需了解通信协议；
- 可以方便的移植到任何基于流的通信协议。

3. MCU 端

提供简单易用的 API，无需了解通信协议，无需记忆变量地址，一般使用下面 4 个函数即可：

- 通过名称设置数据的值；
- 通过名称获取数据的值；
- 处理数据变化的事件；
- 在主循环中分发事件。

对于高级用户，也提供了一些直接操作 GUI 的函数。

也可以集成 TKC，TKC 中提供大量实用函数，可以加快 MCU 端嵌入式软件的开发。

配套硬件

本方案不限制硬件，能运行 AWTK-MVVM 即可。后面的例子可以 PC 上运行，同时也提供了基于 ZDP1440 HMI 显示驱动芯片 环境。

1. 使用方法

1.1 下载并编译 AWTK

将 AWTK 下载到当前目录。AWStudio 带的 AWTK 不是最新的，所以需要自己下载。

```
git clone https://github.com/zlgopen/awtk.git
cd awtk
scons
```

AWTK 的编译方法和环境配置请参考 AWTK 编译指南：
https://gitee.com/zlgopen/awtk/blob/master/README_zh.md

1.2 下载并编译 AWTK-MVVM

将 AWTK-MVVM 下载到当前目录。

```
git clone https://github.com/zlgopen/awtk-mvvm.git
cd awtk-mvvm
scons WITH_JERRYSCRIPT=false
```

用 WITH_JERRYSCRIPT=false 参数禁止编译 jerryscript，因为 jerryscript 不是必须的。

1.3 编译 MCU 模拟器

编译 MCU 模拟器

```
cd mcu/simulator
scons
python scripts/update_res.py all
```

运行 MCU 模拟器

```
./bin/mcu_sim
```

1.4 编译 HMI Demo

编译 HMI Demo

```
cd hmi/demo_app3
scons
python scripts/update_res.py all
```

运行 HMI Demo

```
./bin/demo
```

【技术分享】AWTK 串口屏开发(1) - Hello World

ZLG 致远电子 2023-12-08 11:34:37

本文以一个简单的温度设置程序为例，介绍 AWTK 串口屏的开发流程和 MVVM 数据绑定的基本方法。

功能

这个例子很简单，制作一个调节温度的界面。在这里例子中，模型（也就是数据）里只有一个温度变量：

变量名	数据类型	功能说明
温度	整数	温度。范围 (0-100) 摄氏度

创建项目

从模板创建项目，将 hmi/template_app 拷贝 hmi/hello_word 即可。

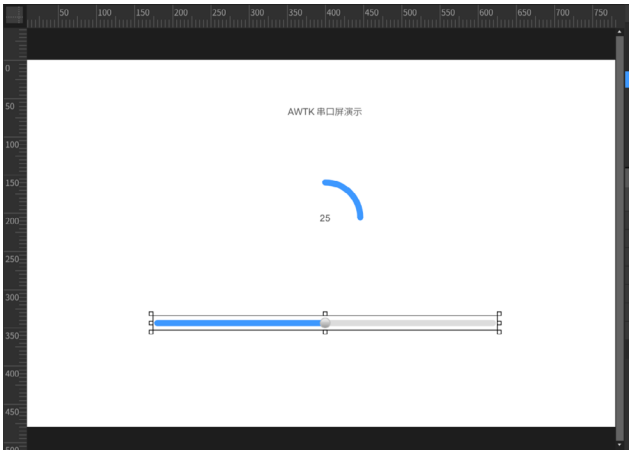
第一个项目最好不要放到其它目录，因为放到其它目录需要修改配置文件中的路径，等熟悉之后再考虑放到其它目录。路径中也不要中文和空格，避免不必要的麻烦。

制作界面

用 AWStudio 打开上面 hello_world 目录下的 project.json 文件。里面有一个空的窗口，在上面加入下面的控件：

- 静态文本
- 环形进度条
- 滑动条

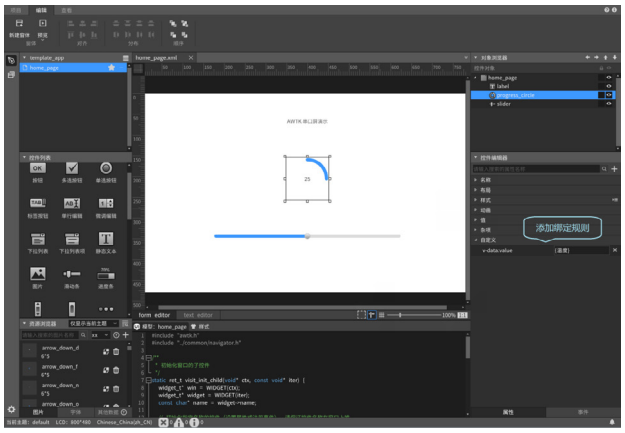
并调节位置和大 小，做出类似下面的界面。



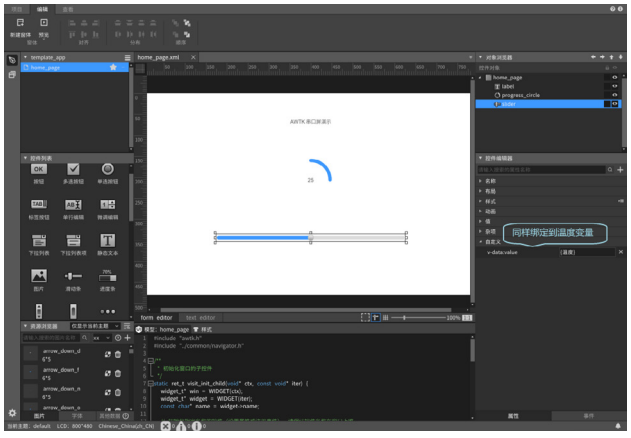
添加绑定规则

1. 将 环形进度条 绑定到 温度 变量。添加自定义的属性 v-data:value，将值设置为 { 温度 }，如下图所示：

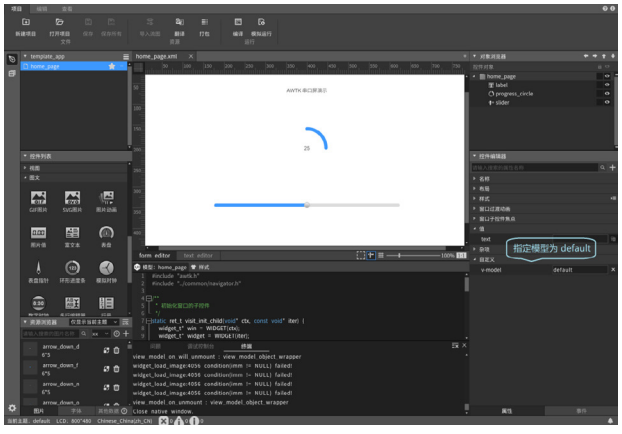
v-data:value 表示控件的值，后面会经常用到，建议记住它。



2. 将 滑动条 绑定到 温度 变量。添加自定义的属性 v-data:value，将值设置为 { 温度 }，如下图所示：



3. 指定窗口的模型为 default。如下图所示：



严格的意义上说，绑定规则也是一种代码，不过相比于 C 语言，它有下面的优势：

- 无需编译，直接运行
- 简单，通常只有一行。
- 易懂，声明式的语法。

初始化数据

修改资源文件 design/default/data/default_model.json, 将其内容改为：

```
{
  " 温度 ":25
}
```

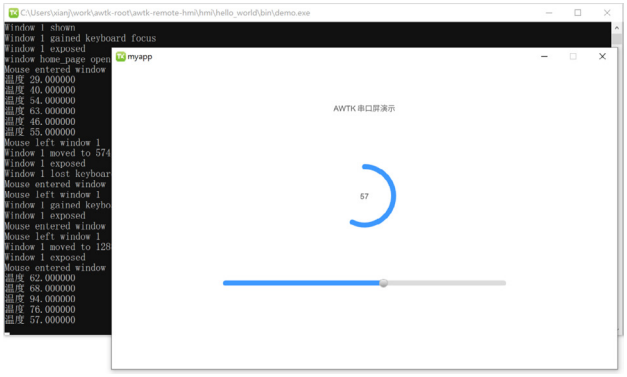
注意：

如果文件内容有中文 (非 ASCII 字符)，一定要保存为 UTF-8 格式。

重新打包资源才能生效。

编译运行

运行 bin 目录下的 demo 程序，拖动滑动条上的滑块，滑动条的数据也会跟随改变。



使用 MCU 模拟器与之进行交互

运行 mcu/simulator 目录下的 mcu_sim 程序，连接到 Localhost:2233。

- 拖动滑动条上的滑块，会看到模拟器上收到了对应的事件；
- 在模拟器中设置变量温度的数据，HMI 端的界面也会自动更新。

注意

本项目并没有编写界面相关的代码，AWStudio 在 src/pages 目录下生成了一些代码框架，这些代码并没有用到，可以删除也可以不用管它，但是不能加入编译。

【技术分享】AWTK 串口屏开发(2) - 家居控制

ZLG 致远电子 2023-12-15 11:43:42

本文以一个家居控制应用程序为例，介绍 AWTK 串口屏的开发流程和 MVVM 数据绑定的高级用法。

1. 功能

这个例子稍微复杂一点，重点关注数据绑定。在这个例子中，模型（也就是数据）里包括一台空调和一台咖啡机：

变量名	数据类型	功能说明
空调 _ 开关	布尔	空调开关
空调 _ 模式	整数	空调模式 (0: 制冷；1: 制热；2: 送风；2: 除湿；4: 自动)
空调 _ 风速	整数	0-4 共五档
空调 _ 垂直风向	整数	垂直风向 (0: 自动; 1: 上; 2: 中; 3: 下)
空调 _ 水平风向	整数	水平风向 (0: 自动; 1: 左; 2: 中; 3: 右)
空调 _ 温度	布尔	温度（0-40）
咖啡 _ 开关	整数	咖啡开关
咖啡 _ 类型	整数	类型 (0: 卡布奇诺；1: 拿铁；2: 美式；3: 意式)
咖啡 _ 温度	整数	温度（0-100）
咖啡 _ 口味	整数	口味 (0: 浓郁；1: 丝滑；2: 清淡；3: 平衡；4: 温和)
咖啡 _ 热奶	整数	热奶 (0: 少量; 1: 较少; 2: 较多; 3: 大量)
咖啡 _ 奶泡	整数	奶泡 (0: 少量; 1: 较少; 2: 较多; 3: 大量)
咖啡 _ 水量	整数	水量 (50-350ml)
咖啡 _ 剩余时间	整数	制作时间（格式：分钟：秒）
咖啡 _ 开始制作	布尔	开始制作

2. 创建项目

从模板创建项目，将 hmi/template_app 拷贝 hmi/home_automation 即可。

项目最好不要放到其它目录，因为放到其它目录需要修改配置文件中的路径，等熟悉之后再考虑放到其它目录。路径中也不要中文和空格，避免不必要的麻烦。

3. 制作界面

界面和资源就直接用了 ZDP1440 显示驱动芯片例子：

主界面

空调界面

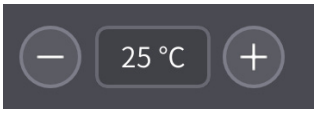
咖啡机界面

4. 添加绑定规则

里面的控件太多，为了不至于太累赘，不同类型的绑定只举一个例子：完整示例可以参考 hmi/demo_app3

4.1 温度设置

这种两个按钮带一个静态文本的组合很常见。



中间的静态文本

绑定属性	绑定规则	说明
v-data:value	{ 空调 _ 温度 + ‘℃’ }	

右边的按钮（增加）

绑定属性	绑定规则	说明
v-on:click	{fscript, Args=set(空调 _ 温度，min(空调 _ 温度 +1, 40))}	这里用函数 set 将变量空调 _ 温度增加 1 度。min 函数保证变量的值不会超出 40。

左边的按钮（减少）

绑定属性	绑定规则	说明
v-on:click	{fscript, Args=set(空调 _ 温度，max(空调 _ 温度 -1, 40))}	这里用函数 set 将变量空调 _ 温度减少 1 度。max 函数保证变量的值不会小于 0。

v-on:click 是一个常见的事件，最好记住，可以提高效率。

4.2 模式选择

这个用一组单选按钮实现，将多个单选按钮放到 group_box 里（将多个单选按钮放到 view 里也可以，只是需要为每个单选按钮编写绑定规则）。

如果绑定规则写到 group_box 上。这样写即可：

绑定属性	绑定规则	说明
v-data:value	{ 空调 _ 模式 }	

如果使用 view 作为容器，则麻烦一点。需要为每个单选按钮编写两条绑定规则：

绑定属性	绑定规则	说明
v-data:value	{ 空调 _ 模式 == 0}	这里的 0 是单选按钮的序数，根据实际情况调整
v-on:click	{fscript, Args=set(空调 _ 模式，0)}	这里的 0 是单选按钮的序数，根据实际情况调整

这种方式虽然麻烦，但是可以处理变量的值不是从 0 开始或者不连续的情况。

4.3 模式显示

模式用一个静态文本显示。问题在于，模式在内部用一个正整数表示，而显示的是一个用户可以理解的字符串。所以需要一个转换函数 one_of：

绑定属性	绑定规则	说明
v-data:value	v-data:value="{one_of(‘制冷 ; 制热 ; 送风 ; 除湿 ; 自动’ , 空调 _ 模式)}	这里的 one_of 的功能是从指定的字符串数组中取出对应的子串。

4.4 剩余时间

剩余时间用一个静态文本显示。问题在于，剩余时间在内部用一个正整数表示 (秒数)，而显示的是“分钟 : 秒”。所以需要一个转换表达式：

绑定属性	绑定规则	说明
v-data:value	int(咖啡 _ 剩余时间 /60) + ‘:’ + 咖啡 _ 剩余时间 %60}	表达是按浮点数计算的，这里的 int 将结果转换为正数。

4.5 隐藏视图

在点击开关按钮时，会自动显示或隐藏右边的设置视图。这是通过将视图的可见性 (visible) 绑定到开关的状态实现的：

绑定属性	绑定规则	说明
v-data:visible	{ 空调 _ 开关 }	

4.6 指定窗口的模型为 default

这是最简单也是最关键的一步：

绑定属性	绑定规则	说明
v-model	default	

严格的意义上说，绑定规则也是一种代码，不过相比于 C 语言，它有下面的优势：

- 无需编译，直接运行。
- 简单，通常只有一行。
- 易懂，声明式的语法。

5. 初始化数据

修改资源文件 design/default/data/default_model.json, 将其内容改为：

```
{
  " 空调 _ 开关 " : false,
  " 空调 _ 模式 " : 3,
  " 空调 _ 风速 " : 3,
  " 空调 _ 垂直风向 " : 1,
  " 空调 _ 水平风向 " : 1,
  " 空调 _ 温度 ": 25,
  " 咖啡 _ 开关 " : false,
  " 咖啡 _ 类型 " : 1,
  " 咖啡 _ 温度 " : 60,
  " 咖啡 _ 口味 " : 1,
  " 咖啡 _ 热奶 " : 1,
  " 咖啡 _ 奶泡 " : 1,
  " 咖啡 _ 水量 " : 150,
  " 咖啡 _ 剩余时间 " : 200,
  " 咖啡 _ 开始制作 " : false
}
```

注意：

如果文件内容有中文 (非 ASCII 字符)，一定要保存为 UTF-8 格式。

重新打包资源才能生效。

6. 编译运行

运行 bin 目录下的 demo 程序。

7. 使用 MCU 模拟器与之进行交互

运行 mcu/simulator 目录下的 mcu_sim 程序，连接到 Localhost:2233。

在界面上修改参数，会看到模拟器上收到了对应的事件：

在模拟器中设置变量 咖啡 _ 类型 的数据，HMI 端的界面也会自动更新。

8. 注意

完整示例可以参考 hmi/demo_home2。

本项目并没有编写界面相关的代码，AWStudio 在 src/pages 目录下生成了一些代码框架，这些代码并没有用到，可以删除也可以不用管它，但是不能加入编译。

【技术分享】
AWTK 串口屏开发(3) - 告警信息

ZLG 致远电子 2023-12-22 11:37:38

告警信息是一个常用的功能。在 AWTK 开源串口屏中，内置告警信息模型，只需设计用户界面即可实现告警信息的显示和管理。

1. 功能

告警信息是一个常用的功能，MCU 在设备异常时，会发送告警信息到串口屏，串口屏可以显示告警信息，也可以对告警信息进行管理（保存或清除）。

基本工作原理：

1.MCU 端设置属性名为 log_message，数据类型为字符串，数据格式为 " 告警级别 | 时间 | 设备 | 告警信息 " 数据。

2. 串口屏收到数据后，会把告警信息放到一个名为 ** log_message ** 的模型（数据）中。

3. 界面通过绑定规则将 log_message 模型中的数据关联到控件上。

告警级别可以是：调试 (0); 信息 (1); 警告 (2); 错误 (3)。

告警信息中如果出现 “|” 字符，则整个告警信息需用用英文双引号引起来。

下面演示一下具体的实现方法。

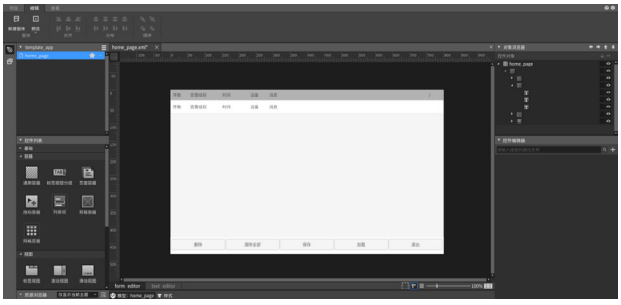
2. 创建项目

从模板创建项目，将 hmi/template_app 拷贝 hmi/log_message 即可。

第一个项目最好不要放到其它目录，因为放到其它目录需要修改配置文件中的路径，等熟悉之后再考虑放到其它目录。路径中也不要中文和空格，避免不必要的麻烦。

3. 制作界面

用 AWStudio 打开上面 log_message 目录下的 project.json 文件。里面有一个空的窗口，在上面设计类似下面的界面：



中间是一个列表视图，列表视图中放一个列表项，列表项中放 5 个文本控件，分别用来显示序数、告警级别、时间、设备、告警信息。

4. 添加绑定规则

第一次用到列表视图，有几点需要特别说明一下：

列表视图中的滚动视图需要指定 v-for-items 属性：

属性	值	说明
v-for-items	true	它保证其下的列表项，会根据数据自动生成 |

4.0.1 几个特殊的变量

- index 特指序数。
- item 特指当前的数据。[0] 表示第一个数据，[1] 表示第二个数据，以此类推。比如在这里 item.[0] 表示告警级别，item.[1] 表示时间，item.[2] 表示设备，item.[3] 表示告警信息。
- selected_index 表示当前选中的序数（可在列表视图之外绑定）。
- items 表示当前列表视图中的数据个数（可在列表视图之外绑定）。

4.0.2 几个特殊的命令

- set_selected 设置当前选中的序数（在列表项中使用）。
- save 保存数据到文件（在列表视图之外的按钮上绑定）。
- reload 重新加载数据（在列表视图之外的按钮上绑定）。
- clear 清除所有数据（在列表视图之外的按钮上绑定）。
- remove 删除指定序数的数据（在列表视图之外的按钮上绑定）。

4.1 序数

绑定属性	绑定规则	说明
v-data:value	{index}	index 特指序数。

4.2 告警级别

前面提到告警级别是正数，可以通过 item.[0] 来获取它。它的意义对应为：debug(0)、info(1)、warning(2)、error(3)，我们需要用 one_of 函数将它转换为对应的字符串。

绑定属性	绑定规则	说明
v-data:value	{one_of(‘调试; 信息; 警告; 错误’ , item.[0])}	这里的 one_of 的功能是从指定的字符串数组中取出对应的子串。

4.3 时间

时间是一个字符串，可以通过 item.[1] 来获取。

绑定属性	绑定规则	说明
v-data:value	{item.[3]}	

4.4 设备

设备是一个字符串，可以通过 item.[2] 来获取。

绑定属性	绑定规则	说明
v-data:value	{item.[2]}	

4.5 告警信息

告警信息是一个字符串，可以通过 item.[3] 来获取。

绑定属性	绑定规则	说明
v-data:value	{item.[3]}	

4.6 列表项

为了配合删除选中的告警信息，需要在列表项加两个绑定规则。

绑定属性	绑定规则	说明
v-on:click	{set_selected}	点击时将当前项目设置为选中
v-data:focused	{index==selected_index}	当前项目选中时高亮

4.7 删除当前选择的告警信息

绑定属性	绑定规则	说明
v-on:click	{remove, Args=selected_index}	selected_index 表示当前选中的项目

4.8 清除所有告警信息

绑定属性	绑定规则	说明
v-on:click	{clear}	

4.9 保存告警信息

绑定属性	绑定规则	说明
v-on:click	{save}	

4.10 重新加载告警信息

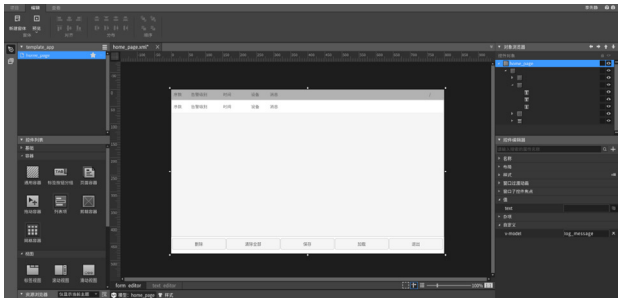
绑定属性	绑定规则	说明
v-on:click	{reload}	

4.11 退出应用程序

绑定属性	绑定规则	说明
v-on:click	{nothing, QuitApp=true}	

4.12 指定窗口的模型

指定窗口的模型为 log_message。



5. 添加告警信息

修改 design/default/data/settings.json 文件，启用告警信息：

```
{
  "name": "hmi_log_message1",
  "log_message": {
    "enable": true, /* 是否启用告警信息 */
    "fields": [
      "level" /* 告警级别 */,
      "time" /* 日期时间 */,
      "device" /* 设备 */,
      "message" /* 信息 */
    ],
    "fields_seperator": "|", /* 字段之间的分隔符 */
    "max_rows": 1000 /* 告警信息最大行数 */
  }
}
```

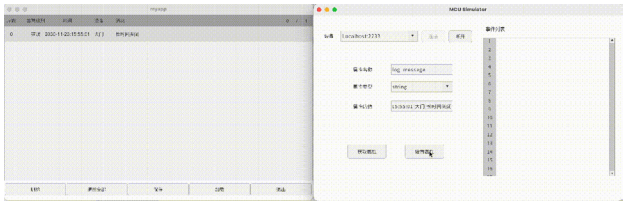
6. 编译运行

运行 bin 目录下的 demo 程序。

7. 使用 MCU 模拟器与之进行交互

运行 mcu/simulator 目录下的 mcu_sim 程序，连接到 Localhost:2233。

- 通过模拟器发送数据，可以看到串口屏界面，自动添加告警信息。



测试数据：

- 3|2030-11-23:15:55:01| 大门 | 长时间关闭

8. 注意

本项目并没有编写界面相关的代码，AWStudio 在 src/pages 目录下生成了一些代码框架，这些代码并没有用到，可以删除也可以不用管它，但是不能加入编译。

实际使用时，在 demo_app4/design/default/ui/home_page.xml 基础上进行调整即可，无需重复上面的过程，但是最好了解其中的原理。

【技术分享】AWTK 串口屏开发(4) - 数据采集

ZLG 致远电子 2023-12-29 11:42:48

数据采用是一个常用的功能。在 AWTK 开源串口屏中，内置数据采样模型，只需设计用户界面即可实现采样数据的显示和管理。

1. 功能

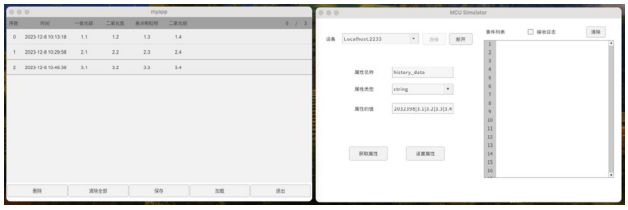
数据采集是一个常用的功能，MCU 定时采集数据（如环保设备定时采样空气中的污染物），并发送采样数据到串口屏，串口屏可以显示采样数据，也可以对采样数据进行管理（保存或清除）。

基本工作原理：

MCU 端设置属性名为 history_data，数据类型为字符串，数据格式为用 | 分隔的多个字段的数据。

串口屏收到数据后，会把采样数据放到一个名为 history_data 的模型（数据）中。

界面通过绑定规则将 history_data 模型中的数据关联到控件上。



时间为 epoch 时间，方便内部存储和查询。

下面演示一下具体的实现方法。

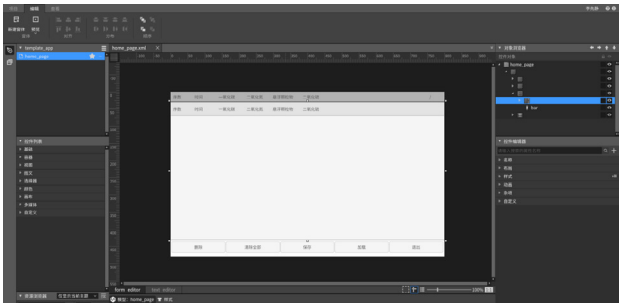
2. 创建项目

从模板创建项目，将 hmi/template_app 拷贝 hmi/history_data 即可。

第一个项目最好不要放到其它目录，因为放到其它目录需要修改配置文件中的路径，等熟悉之后再考虑放到其它目录。路径中也不要中文和空格，避免不必要的麻烦。

3. 制作界面

用 AWStudio 打开上面 history_data 目录下的 project.json 文件。里面有一个空的窗口，在上面设计类似下面的界面：



中间是一个列表视图，列表视图中放一个列表项，列表项中放 6 个文本控件，分别用来显示序数、时间、一氧化碳、二氧化氮、悬浮颗粒物、二氧化硫。

4. 添加绑定规则

第一次用到列表视图，有几点需要特别说明一下：

列表视图中的滚动视图需要指定 v-for-items 属性：

属性	值	说明
v-for-items	true	它保证其下的列表项，会根据数据自动生成

4.0.1 几个特殊的变量

- index 特指序数。
- item 特指当前的数据。比如在这里 ‘item.time’ 表示时间，‘item.一氧化碳’表示一氧化碳，‘item. 二氧化氮’表示二氧化氮，‘item. 悬浮颗粒物’ 表示悬浮颗粒物。
- selected_index 表示当前选中的序数（可在列表视图之外绑定）。
- items 表示当前列表视图中的数据个数（可在列表视图之外绑定）。

4.0.2 几个特殊的命令

- set_selected 设置当前选中的序数（在列表项中使用）。
- save 保存数据到文件（在列表视图之外的按钮上绑定）。
- reload 重新加载数据（在列表视图之外的按钮上绑定）。
- clear 清除所有数据（在列表视图之外的按钮上绑定）。
- remove 删除指定序数的数据（在列表视图之外的按钮上绑定）。

4.1 序数

绑定属性	绑定规则	说明
v-data:value	{index}	index 特指序数。

4.2 时间

时间是整数 (秒数)，可以通过 item.time 来获取。

绑定属性	绑定规则	说明
v-data:value	{date_time_format(item.time, 'Y-M-D h : m : s')}	需要用 date_time_format 将 epoch 时间转换成人类可读的时间。

4.3 一氧化碳

可以通过 item. 一氧化碳 来获取。

绑定属性	绑定规则	说明
v-data:value	{item. 一氧化碳 }	无

4.4 二氧化氮

可以通过 item. 二氧化氮 来获取。

绑定属性	绑定规则	说明
v-data:value	{item. 二氧化氮 }	无

4.5 悬浮颗粒物

可以通过 item. 悬浮颗粒物 来获取。

绑定属性	绑定规则	说明
v-data:value	{item. 悬浮颗粒物 }	无

4.6 二氧化硫

可以通过 item. 二氧化硫 来获取。

绑定属性	绑定规则	说明
v-data:value	{item. 二氧化硫 }	无

4.7 列表项

为了配合删除选中的采样数据，需要在列表项加两个绑定规则。

绑定属性	绑定规则	说明
v-on:click	{set_selected}	点击时将当前项目设置为选中
v-data:focused	{index==selected_index}	当前项目选中时高亮

4.8 删除当前选择的采样数据

绑定属性	绑定规则	说明
v-on:click	{remove, Args=selected_index}	selected_index 表示当前选中的项目

4.9 清除所有采样数据

绑定属性	绑定规则	说明
v-on:click	{clear}	无

4.10 保存采样数据

绑定属性	绑定规则	说明
v-on:click	{save}	无

4.11 重新加载采样数据

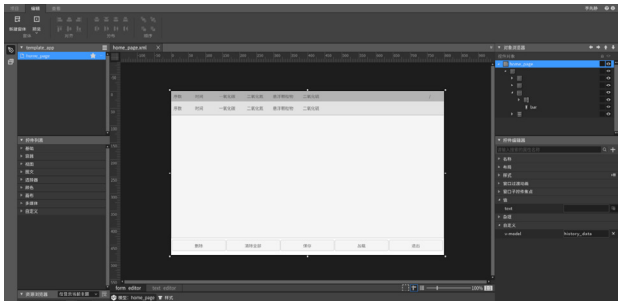
绑定属性	绑定规则	说明
v-on:click	{reload}	无

4.12 退出应用程序

绑定属性	绑定规则	说明
v-on:click	{nothing, QuitApp=true}	无

4.13 指定窗口的模型

指定窗口的模型为 history_data。



5. 启用数据采样

修改 design/default/data/settings.json 文件，启用数据采样：

```
{
  "name": "hmi_histroy_data1",
  "history_data": {
    "enable": true, /* 是否启用数据采集 */
    "fields": {
      "time": {}, /* 时间必须用 'time'，放在第一位 */
      " 一氧化碳 " : {
        "min": 0,
        "max": 100,
        "unit": "mg/m³"
      },
      " 二氧化氮 " : {
        "min": 0,
        "max": 110,
        "unit": "mg/m³"
      },
      " 悬浮颗粒物 " : {
        "min": 0,
        "max": 120,
        "unit": "mg/m³"
      },
      " 二氧化硫 ": {
        "min": 0,
        "max": 130,
        "unit": "mg/m³"
      }
    },
    "fields_seperator": "|", /* 字段之间的分隔符 */
    "max_rows": 1000 /* 数据采集最大行数 */,
    "auto_save_interval": 60000
  }
}
```

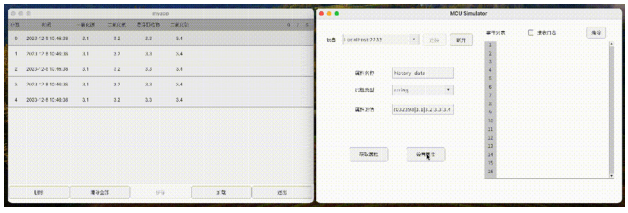
6. 编译运行

运行 bin 目录下的 demo 程序。

7. 使用 MCU 模拟器与之进行交互

运行 mcu/simulator 目录下的 mcu_sim 程序，连接到 Localhost:2233。

通过模拟器发送数据，可以看到串口屏界面，自动添加采样数据。



测试数据：

1702032398|3.1|3.2|3.3|3.4

8. 注意

本项目并没有编写界面相关的代码，AWStudio 在 src/pages 目录下生成了一些代码框架，这些代码并没有用到，可以删除也可以不用管它，但是不能加入编译。

实际使用时，在 demo_history_data1/design/default/ui/home_page.xml 基础上进行调整即可，无需重复上面的过程，但是最好了解其中的原理。

【EsDA 应用】如何利用AWFlow搭建HTTP服务器

ZLG 致远电子 2023-12-13 11:44:13

本文将基于 EsDA 开发套件快速在开发板上搭建 HTTP 服务器，为 HTTP 客户端提供了访问板子的外设资源，数据以及文件读写等接口，实现设备与网络服务的互联互通。

项目简介

在物联网应用中，在开发板上搭建 HTTP 服务器是一项很重要的技术，可以实现设备间的数据通信和远程控制。本文将介绍如何使用 EPC6450-AWI 开发板 ，以及图形化设计工具 AWFlow Designer 搭建 HTTP 服务器，对设备数据进行远程访问和控制。

项目概述

当客户端发送 HTTP 请求时，服务器接收请求，并根据请求的内容做出相应的处理，如读写开发板资源的配置和数据，读写文件等操作。开发板搭建的 HTTP 服务器，通过 HTTP 协议与客户端进行通信。本文 HTTP 服务器的项目可为客户端提供以下五个接口：

1. 写内容到开发板的文件中；
2. 读开发板中的文件内容；
3. 提供读取 ADC 通道的电压值；
4. 提供读取温湿度传感器的温度值和湿度值；
5. 提供设置 Modbus 从站的寄存器值。本项目采用的 Modbus 从站是模拟 Modbus 从站设备的上位机 Modbus Slave（工具使用请见【EsDA 应用】Modbus RTU Master 转 MQTT 的“3.2.4、Modbus Slave 安装并使用”章节）。

前期准备

读者可以阅读以下文章对 HTTP 通信和 EsDA 的一些基础项目操作进行熟络：

- 【EsDA 应用】快速实现串口转 HTTP 请求
- 【EsDA 应用】Modbus 应用详解
- 【EsDA 应用】5 分钟实现一个串口通信业务
- 【EsDA 应用】常用 IO 设备节点详解

1. 硬件准备

在标有丝印为 TF Card 丝印的卡槽处，插入 SD 卡。

在标有丝印为 DUART 的调试串口模块上，将 TTL 转 USB 串口模块的 TX 与板子丝印为 RXD 连接，TTL 转 USB 串口模块的 RX 与板子丝印为 TXD 相连；并将 TTL 转 USB 串口模块的 USB 端口接入电脑。

在标有丝印为 Type-C 的接口处，插上 Type-C 线，并将 Type-C 线的另一端 USB 口插入电脑。

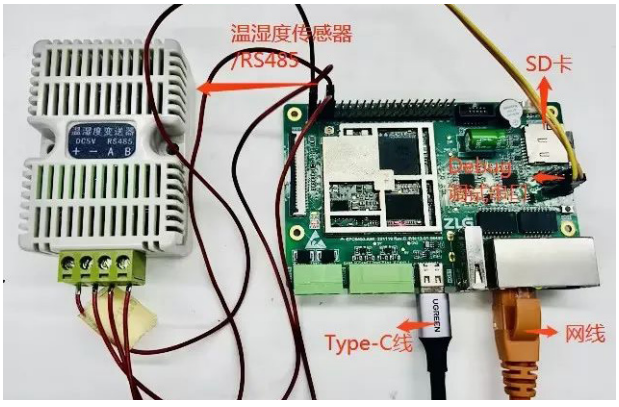
在标有丝印为 NET0 或 NET1 的 RJ45 插座处接上网线的水晶头，网线另一端的水晶头插在 PC 的网络接口处。

1.1 提供温湿度传感器值的接口项目

在标有丝印为 5V 的排针针脚处接上温湿度变送器的 DC5V_+ 引脚；在标有丝印为 GND 的排针针脚处接上温湿度变送器的 DC5V_- 引脚；在标有丝印为 485A 的排针针脚处接上温湿度变送器的 RS485_A 的引脚；在标有丝印为 485B 的排针针脚处接上温湿度变送器的 RS485_B 的引脚上。

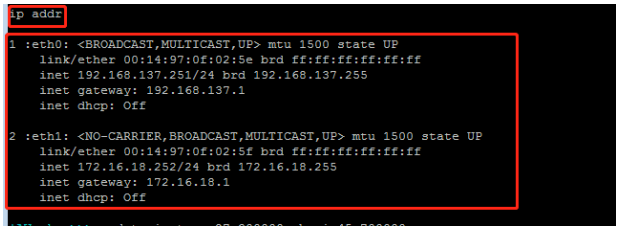
1.2 设置Modbus从站寄存器值的接口项目

在标有丝印位 5V 的排针针脚处接上 USB 转 485 串口设备的 +5V 电源引脚；在标有丝印位 GND 的排针针脚处接上 USB 转 485 串口设备的 GND 接地引脚；在标有丝印位 485A 的排针针脚处接上 USB 转 485 串口设备的 RS485A 引脚；在标有丝印位 485B 的排针针脚处接上 USB 转 485 串口设备的 RS485B 引脚。

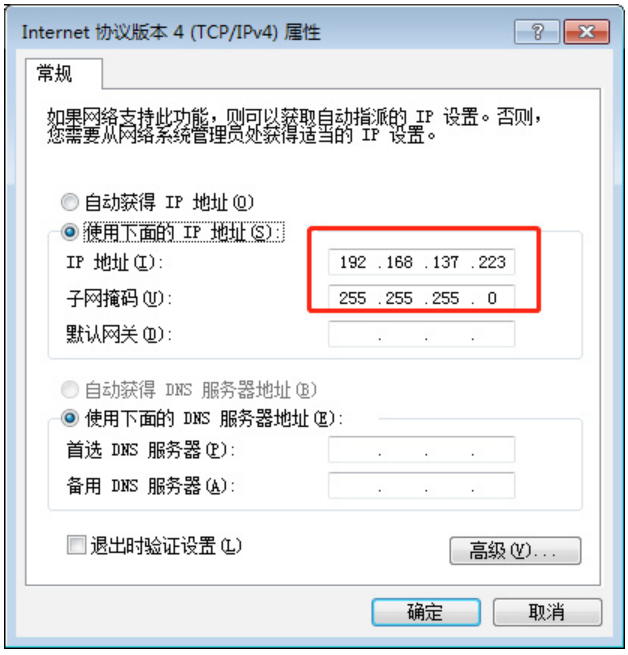


2. 网络搭建

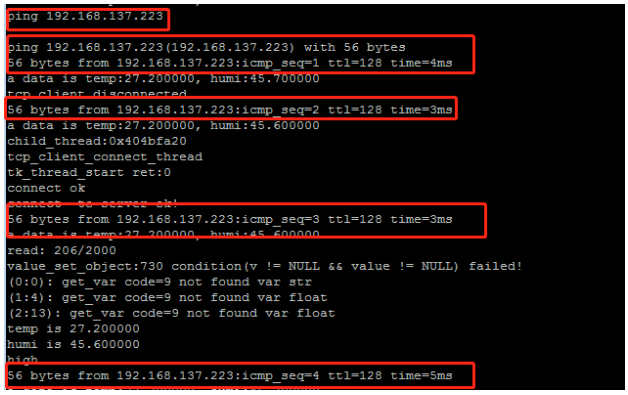
打开串口调试助手，检索并打开 TTL 转 USB 串口模块的设备端口号后，使用 shell 命令 ip addr，查看网口的 IP 地址，根据下图可知，本文使用的网口设备 IP 地址是 192.168.137.251。



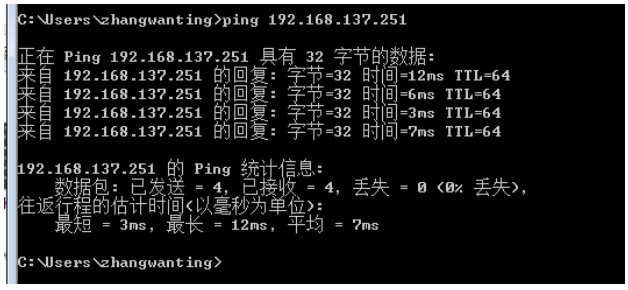
配置 PC 上的以太网的 IP 与开发板的 IP 地址在同一局域网下。



在串口调试助手输入 shell 指令 ping 192.168.137.223，ping 成功即开发板可以单方面 ping 成功 PC，若 ping 失败则可以关闭 PC 端的防火墙之后再尝试。



电脑端执行 WIN+R 输入 cmd 回车即可打开 CMD 命令提示符窗口，执行 ping 192.168.137.251，ping 成功则电脑可以单方面 ping 成功开发板。当开发板和 PC 可以互 ping 成功则联网成功。



项目实施

1. 写文件接口

本接口业务主要是将写入的文件内容通过 HTTP 客户端发送 POST 请求上传到 HTTP 服务器，服务器收到请求后做出相应处理。可分为以下四个部分：

用户在 HTTP 客户端的 body 中编辑需要写入的文件内容；

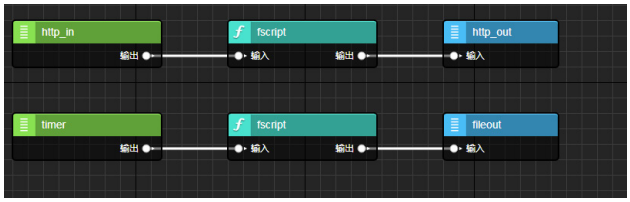
HTTP 客户端设置服务器对应的 url 接口和请求方式，向开发板搭建的 HTTP 服务器发送 HTTP 请求；

HTTP 服务器监听指定端口，处理对应接口的请求并响应；

查看写入到文件的内容，检验是否成功通过开发板搭建的 HTTP 服务器提供的接口，实现远程文件内容的输入。

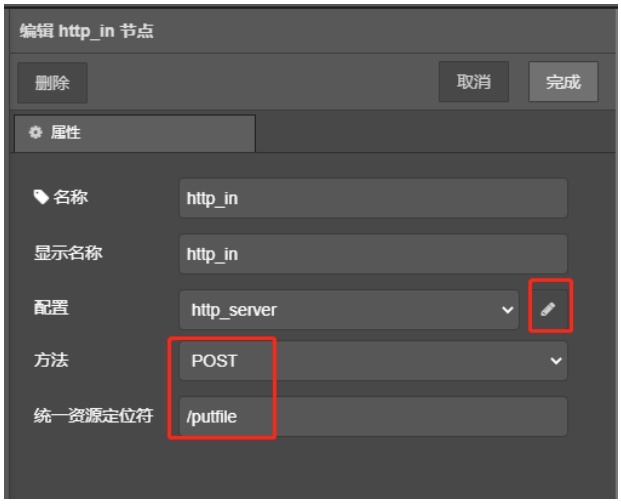
1.1 流图绘制

添加 http_in，fscript，http_out，timer，fscript 和 fileout 节点到画布中并连线如下图。



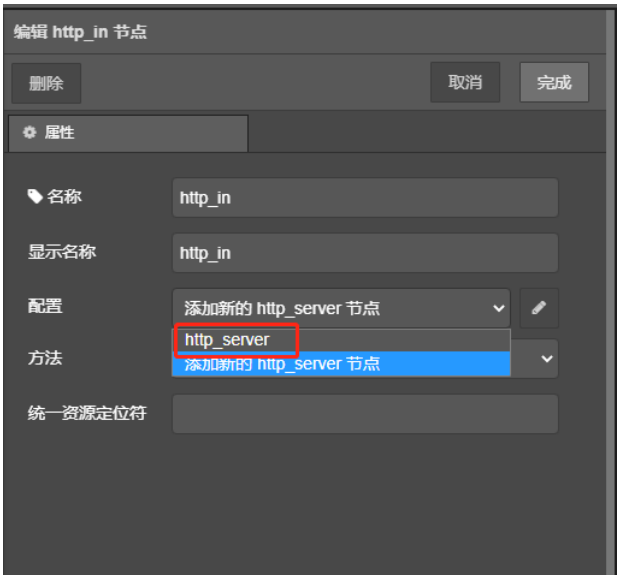
1.2 节点配置

双击 http_in 节点，该节点主要是给 HTTP 客户端提供一个 POST 方法的写文件内容的接口，统一资源定位符可以根据用户需要更改。点击完成按钮即可保存配置。



点击 http_server 旁边的铅笔图标对 http_server 节点进行配置，通常情况下修改访问的端口号即可，点击更新按钮即可保存配置。

本项目其他的接口都是在同一个 HTTP 服务器上处理，所以后续的接口添加新的 http_in 节点，直接选中之前配置好的 http_server，不用再重新添加新的 http_server 节点（不用重新创建一个 http_server）。后面的接口项目不再赘述。



双击 http_in 节点的消费者节点 fscript，该节点主要是接收 HTTP 客户端上传消息体的内容，并将接收写文件接口请求的标志位置一。点击完成按钮即可保存配置。

```
global.put_file_status = 1
global.putfile = str(msg.payload,true)
```

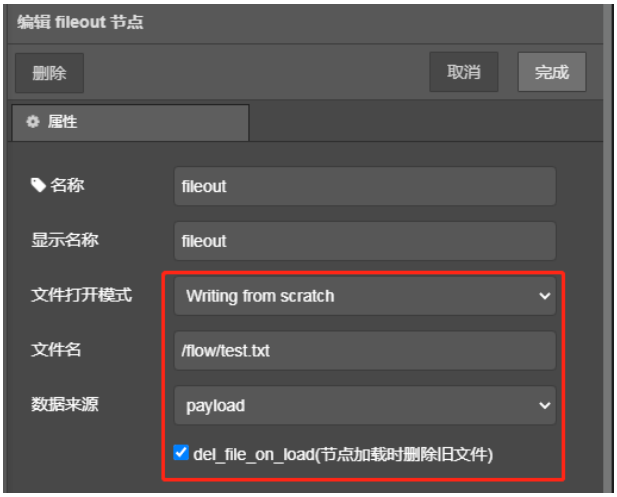
双击 http_out 节点，可以根据用户实际需要设置响应码和消息头，本项目使用默认的就可以了。点击完成按钮即可保存配置。

双击 timer 节点，设置定时写入文件的周期时间。

双击 timer 的消费者节点 fscript，该节点主要是处理当写文件接口的 http_in 节点被触发时那么 put_file_status（写文件标志位）置一，之后才将收到的请求体的内容写入文件中。

```
if(global.put_file_status == 1) {
    wb = wbuffer_create()
    wbuffer_write_string(wb,global.putfile)

    output.payload = wbuffer_get_data(wb)
    output.payloadLength = len(global.putfile)
    global.put_file_status = 0
}
```

双击 fileout 节点，可根据用户需要配置属性参数。因为本项目的数据来源是从前节点 fscript 的 payload 中获取的，所以属性数据来源选择 payload 选项。



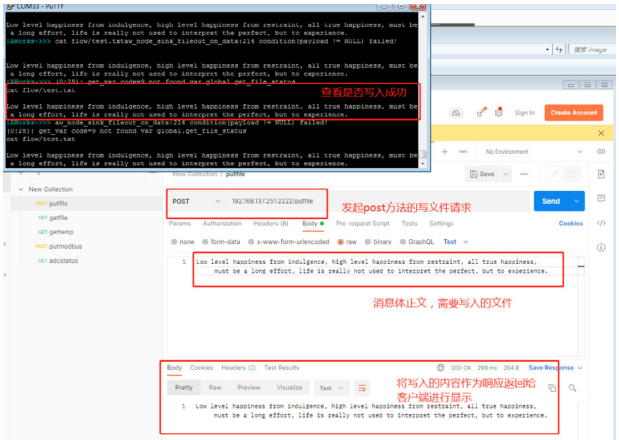
1.3 流图下载

绘制完流图后，点击 CTRL+S 即可保存流图，点击下载流图。

在 HTTP 客户端 postman 上发起写文件请求，HTTP 服务器收到请求后将写入的内容作为响应返回到客户端进行显示，并执行写文件的操作，在串口调试助手上输入

cat flow/test.txt

即可看到写入文件的内容。结果如下图所示则表示 HTTP 服务器搭建写文件的接口已基本实现。



2. 读文件接口

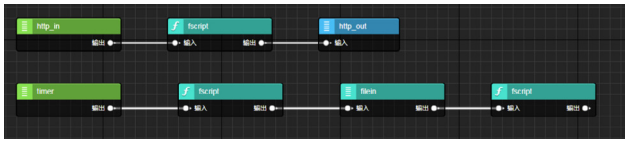
本接口业务主要是 HTTP 客户端发送 GET 方法的读文件请求，HTTP 服务器接收到请求之后将文件内容作为响应返回到客户端进行显示。可分为以下两个部分：

HTTP 客户端设置读文件的 url 接口和请求方式，向开发板搭建的 HTTP 服务器发送 HTTP 请求；

HTTP 服务器监听指定端口，接收到读文件的请求后将读文件标志位置一，等待读取文件内容，并将读到的内容作为响应发送到 HTTP 客户端。

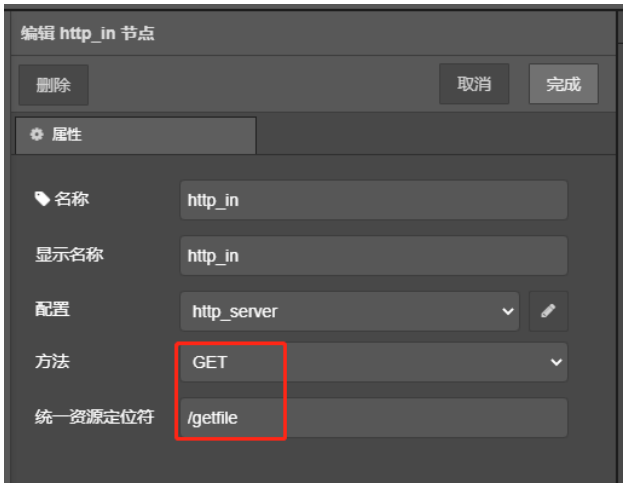
2.1 流图绘制

添加 http_in，fscript，http_out，timer 和 filein 节点到画布中并连线如下图。



2.2 节点配置

双击 http_in 节点，该节点主要是给 HTTP 客户端提供一个 GET 方法的读文件内容的接口，统一资源定位符可以根据用户需要更改。点击完成按钮即可保存配置。



双击 http_in 的消费者节点 fscript，该节点的主要功能是当 HTTP 服务器接收到了来自客户端的读文件接口的请求，那么将读文件标志位（get_file_status）置一，延时等待读取文件的业务，最后将读取到的数据作为响应发送到 HTTP 客户端。

```
global.get_file_status = 1
sleep_ms(500)
msg.payload = "you get file data is "+global.getfile
```

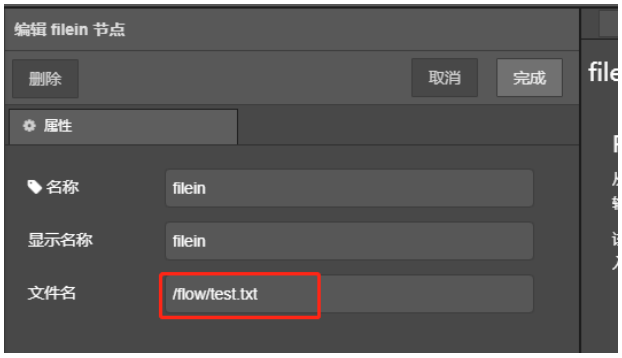
http_out 节点在本项目中使用默认参数值即可。

双击 timer 配置定时读文件的周期时间。

双击 timer 的消费者节点 fscript，该节点主要是检测读文件标志位（get_file_status）置一时，设置 filein 节点的配置参数。点击完成即可保存配置。

```
if(global.get_file_status == 1) {
  set(msg.topic, "exec:read_all")
  set(msg.payload, 4096)
  global.get_file_status = 0
}
```

双击 filein 节点，配置需要读取的文件名。点击完成即可保存配置。



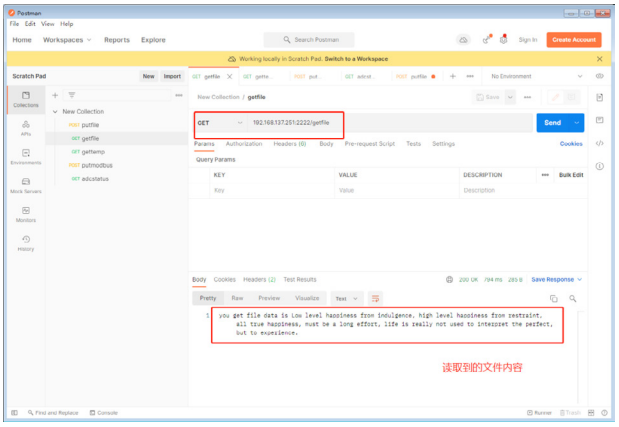
双击 filein 的消费者节点 fscript，该节点主要是将 filein 节点读取出来的文件内容，赋值给可供其他节点访问的全局参数。点击完成即可保存配置。

```
global.getfile = str(msg.payload, true)
```

2.3 流图下载

绘制完流图后，点击 CTRL+S 即可保存流图，点击下载流图。

在 postman 中发送 HTTP 客户端 GET 方法的读文件请求，后续可以在响应报文部分看到由 HTTP 服务器发出的读取文件内容的响应正文。结果如下图所示则表示 HTTP 服务器搭建读文件的接口已基本实现。



3. 读取ADC通道值的接口

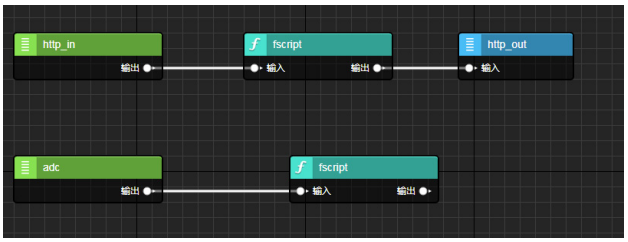
本接口业务主要是 HTTP 服务器提供 ADC 通道的电压值的接口，当 HTTP 客户端发起该请求时，HTTP 服务器将采集到的 ADC 通道电压值作为响应返回到 HTTP 客户端进行显示。可分为以下两个部分：

HTTP 客户端设置服务器获取 ADC 通道值的 url 接口和请求方式，向开发板搭建的 HTTP 服务器发送 HTTP 请求；

HTTP 服务器监听指定端口，接收到对应接口的请求后将采集到的 ADC 通道的电压值作为响应并发送到 HTTP 客户端。

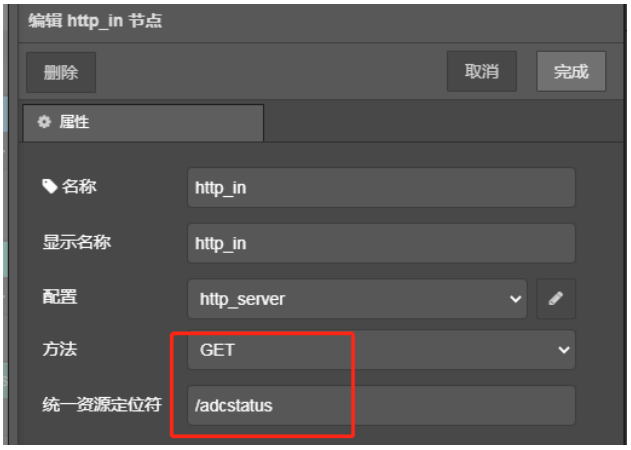
3.1 流图绘制

添加 http_in，fscript，http_out 和 adc 节点到画布中并连线如下图。



3.2 节点配置

双击 http_in 节点，该节点主要是给 HTTP 客户端提供一个 GET 方法的获取 ADC 通道电压值的接口，统一资源定位符可以根据用户需要更改。点击完成按钮即可保存配置。

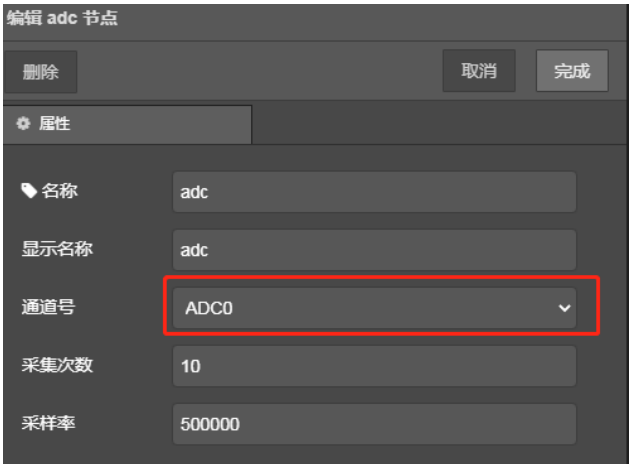


双击 http_in 的消费者节点 fscript，该节点主要目的是将采集到的 ADC 通道的电压值作为响应发送到 HTTP 客户端。点击完成即可保存配置。

```
var a = "adc status is "+ global.adc_value
wb = wbuffer_create()
wbuffer_write_string(wb,a)
msg.payload = wbuffer_get_data(wb)
msg.payloadLength = len(a)
```

本项目中 http_out 节点使用默认配置即可。

双击 adc 节点，配置需要采集的通道号。点击完成即可保存配置。



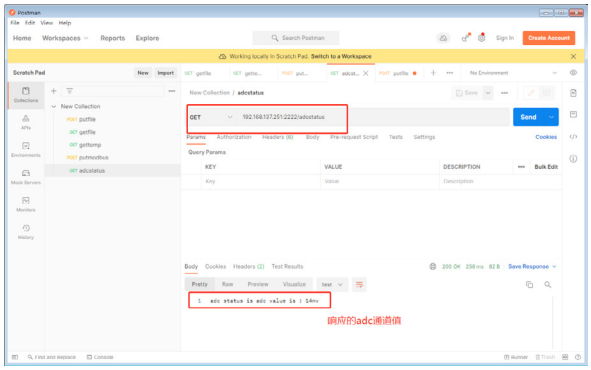
双击 adc 的消费者节点 fscript，该节点主要是将采集到的 ADC 电压值赋值给全局变量，供其他节点使用。点击完成即可保存配置。

```
global.adc_value="adc value is :"+" "+msg.payload + "mv"
```

3.3 流图下载

绘制完流图后，点击 CTRL+S 即可保存流图，点击下载流图。

在 postman 中发送 HTTP 服务器提供的采集 ADC 通道值的接口请求，后续可以在响应部分看到 HTTP 服务器返回的实时采集到的 ADC 电压值。结果如下图所示则表示 HTTP 服务器搭建的获取 ADC 通道电压值的接口已基本实现。



4. 提供读取传感器值的接口

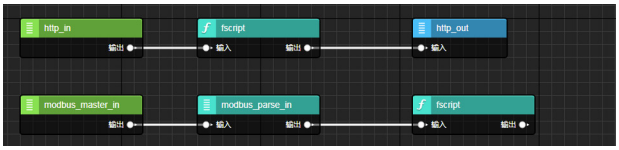
本接口业务主要是 HTTP 服务器提供温湿度传感器采集的温度和湿度值的接口，当 HTTP 客户端发起该请求时，HTTP 服务器将采集的温湿度值作为响应返回到 HTTP 客户端进行显示。可分为以下两个部分：

HTTP 客户端设置获取温湿度传感器值的 url 接口和请求方式，向开发板搭建的 HTTP 服务器发送 HTTP 请求；

HTTP 服务器监听指定端口，接收到对应接口的请求后将采集到的温湿度值作为响应并发送到 HTTP 客户端。

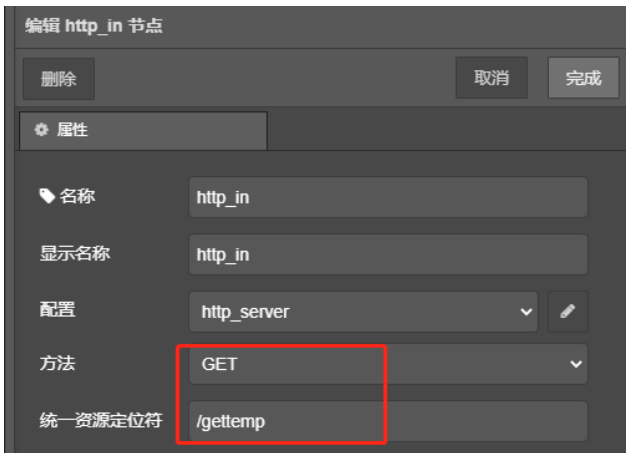
4.1 流图绘制

添 加 http_in，fscript，http_out，modbus_master_in 和 modbus_parse_in 节点到画布中并连线如下图。



4.2 节点配置

双击 http_in 节点，该节点主要是给 HTTP 客户端提供一个 POST 方法的接收温湿度传感器数据的接口，统一资源定位符可以根据用户需要更改。点击完成按钮即可保存配置。



双击 http_in 节点的消费者节点 fscript，该节点主要处理是将采集到的温湿度传感器的值作为响应发送给 HTTP 客户端。点击完成按钮即可保存配置。

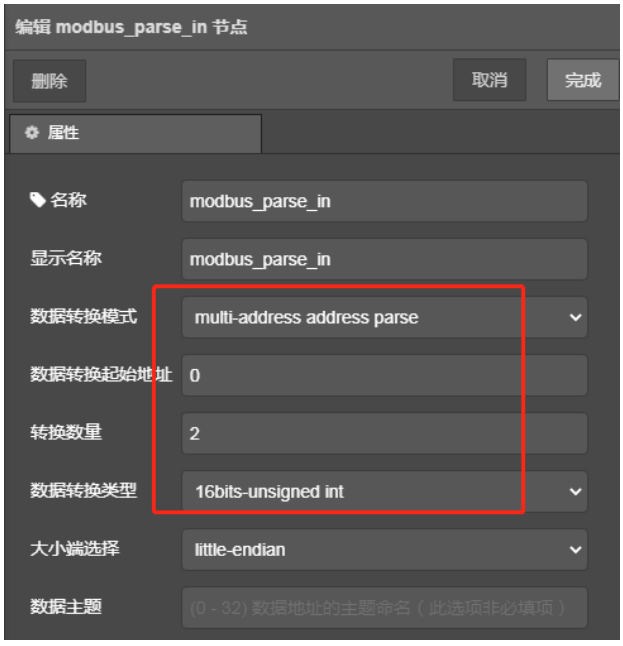
```
msg.payload = global.temp_value
```

http_out 节点配置在本项目中不需要任何更改，使用默认配置即可。

双击 modbus_master_in 节点，配置温湿度传感器从站的相关配置信息如下。点击完成即可保存配置。

点击 modbus_master_in 节点的属性主机参数配置旁边的铅笔图标，因为是通过 RS485 进行 Modbus 通信，所以选择 rtu 传输模式并选择对应的串口设备名，其他串口参数配置根据实际需要进行配置，点击更新即可保存配置。

双击 modbus_parse_in 节点，因为地址 0 是温度值，1 是湿度值，所以配置节点参数如下所示。点击完成即可保存配置。



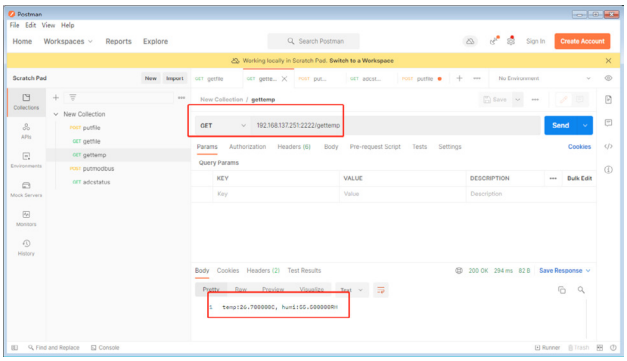
双击 modbus_parse_in 的消费者节点 fscript，本节点主要是解析处理采集到的数据，并赋值给可供其他节点使用的全局变量。点击完成即可保存配置。

```
global.temp_value = "temp:"+array_get(msg.payload,0)/10+"C,
"+"humi:"+array_get(msg.payload,1)/10+"RH"
```

4.3 流图下载

绘制完流图后，点击 CTRL+S 即可保存流图，点击下载流图。

在 postman 中发送 HTTP 服务器提供的采集温湿度传感器值的接口请求，后续可以在响应部分看到 HTTP 服务器返回的温度值和湿度值。结果如下图所示则表示 HTTP 服务器搭建的获取温湿度传感器值的接口已基本实现。



5. 设置Modbus从站寄存器值的接口

本接口业务主要是将需要写入的 Modbus 从站寄存器值通过 HTTP 客户端发送 POST 请求上传到 HTTP 服务器，服务器收到请求后做出相应处理。可分为以下四个部分：

用户在 HTTP 客户端的 body 中编辑需要写入的 Modbus 从站寄存器值内容；

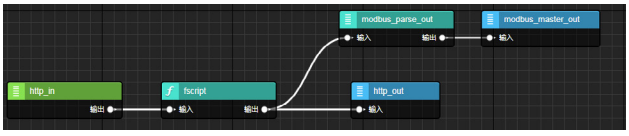
HTTP 客户端设置对应的 url 接口和请求方式，向开发板搭建的 HTTP 服务器发送 HTTP 请求；

HTTP 服务器监听指定端口，处理对应接口的请求并响应；

查看写入到 Modbus 从站寄存器的内容，检验是否成功通过开发板搭建的 HTTP 服务器提供的接口，实现 Modbus 从站寄存器值的远程输入。

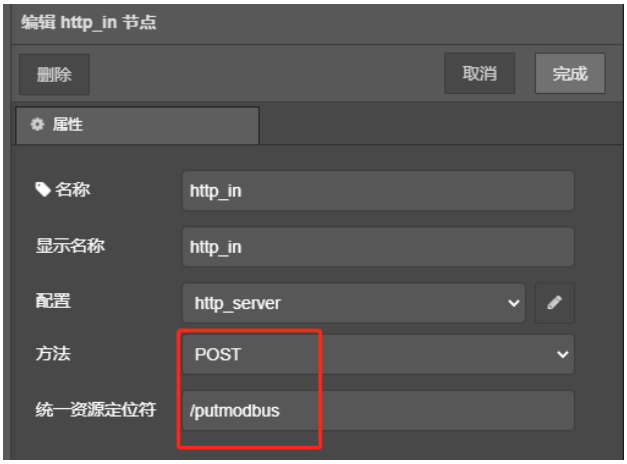
5.1 流图绘制

添 加 http_in，fscript，http_out，modbus_parse_out 和 modbus_master_out 节点到画布中并连线如下图。



5.2 节点配置

双击 http_in 节点，该节点主要是给 HTTP 客户端提供一个 POST 方法的设置 Modbus 从站寄存器值的接口，统一资源定位符可以根据用户需要更改。点击完成按钮即可保存配置。

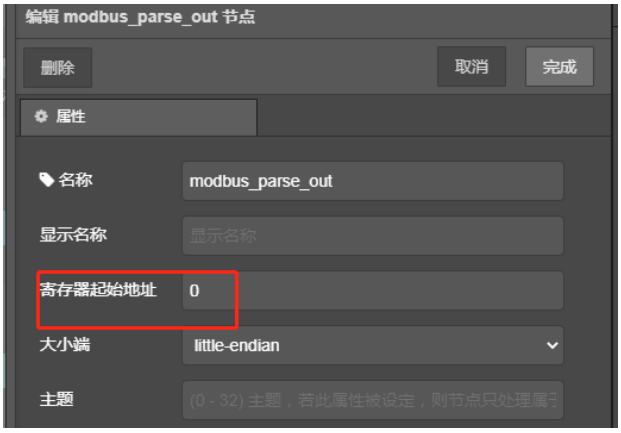


点击 http_in 的消费者节点 fscript，该节点主要是处理将客户端发送的请求消息体解析为 modbus_parse_out 节点可以接收的类型。内容如下：

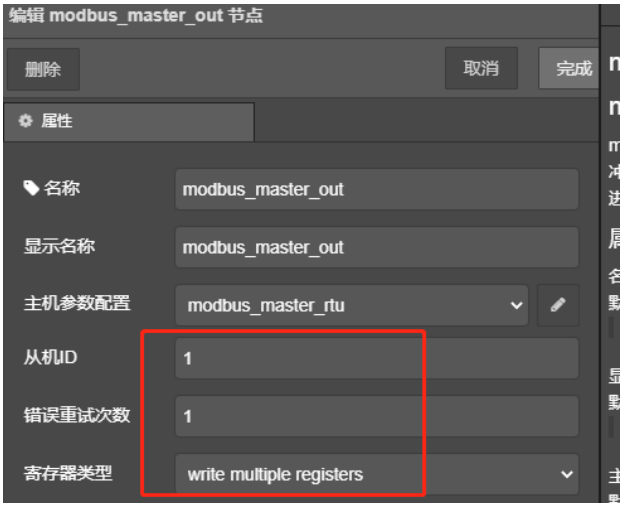
```
a = array_create();

str b = str(msg.payload,true)
b = replace(b,',',":")
data1 = one_of(b,1,":")
data2 = one_of(b,3,":")
data3 = one_of(b,5,":")
array_insert(a, 0, u16(data1))
array_insert(a, 1, u16(data2))
array_insert(a, 2, u16(data3))

output.payload = a;
output.slaveID = 1;
output.address = 0;
```

本项目中 http_out 节点使用默认配置即可。

双击 modbus_parse_out 节点，配置需要从哪个寄存器开始写值。点击完成即可保存配置。



双击 modbus_master_out 节点，配置从机 ID 和写入的寄存器类型。点击完成即可保存配置。



双击 modbus_master_out 节点的属性主机参数配置旁边的铅笔图标，因为是通过 RS485 进行 Modbus 通信，所以选择 rtu 传输模式并选择对应的串口设备名，其他串口参数配置根据实际需要进行配置，点击更新即可保存配置。

5.3 流图下载

绘制完流图后，点击 CTRL+S 即可保存流图，点击下载流图。

在 postman 中发送 HTTP 服务器提供的设置 Modbus 从站寄存器值的接口请求，后续可以在 Modbus Slave 上位机中查看写入的寄存器的值。结果如下图所示则表示 HTTP 服务器搭建的设置 Modbus 从站寄存器值的接口已基本实现。

【从0开始创建AWTK应用程序】编译应用到嵌入式Linux平台运行

ZLG 致远电子 2023-12-06 11:37:16

AWTK 是基于 C 语言开发的跨平台 GUI 框架。本系列文章介绍如何从 0 开始创建 AWTK 应用程序，包括搭建开发调试环境、使用 AWTK 创建 Hello 工程并在模拟器上运行、将 AWTK 应用程序移植到其它平台。

搭建Linux平台交叉编译环境

在上一篇文章我们介绍了使用 AWTK 开发简单的应用并在 PC 上模拟运行，本篇文章就来介绍一下怎么让应用程序运行在嵌入式 Linux 平台上。

1. 准备交叉编译环境

交叉编译通常在 PC 的 Ubuntu 上进行，准备对应板子的交叉编译工具链、AWTK SDK（awtk 与 awtk-linux-fb）、应用工程（**应用需提前在 AWTK Designer 上打包资源**）。

SDK 版本**需要配套，否则会出现编译不通过或运行异常问题**，具体根据 AWTK 应用程序所依赖的 SDK 版本选择，获取 SDK 有三种方式：

1.AWStudio 安装目录下的 Installer 文件夹自带的 SDK；**（推荐）**

2. 板子供应商提供的 SDK，可能与应用依赖版本不兼容，需找供应商更新 SDK；

3.Github 上最新版的 SDK，要求应用使用最新版 awtk 开发，但最新版 SDK 未经测试，可能存在 BUG。

第一种和第三种方式需要自行配置 awtk_config.py，配置方法见下面小节；第二种供应商一般会配置好，或者有专门的配置教程可以参考。

将工具链、SDK 和应用拷贝到 Ubuntu 以下路径，SDK 与应用需放在同一级目录：

工具链：/opt/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/

SDK 与应用：

```
/home/user/
  |-- awtk/
  |-- awtk-linux-fb/
  |-- awtk-hello-demo/
```

2.配置交叉编译工具链路径

将交叉编译工具解压到 Ubuntu 环境内，接着打开 awtk-linux-fb 目录的 awtk_config.py，找到 TOOLS_PREFIX 参数配置工具链的路径。配置 tslib 则需修改 TSLIB_LIB_DIR 与 TSLIB_INC_DIR 参数。

```
# awtk_config.py
TSLIB_LIB_DIR='/opt/tslib/lib'
TSLIB_INC_DIR='/opt/tslib/include'
TOOLS_PREFIX='/opt/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/bin/arm-linux-gnueabihf-'
```

配置设备可以到 awtk-linux-fb/awtk-port/main_loop_linux.c 修改，下图分别对应着 Framebuffer、触摸屏、键盘和鼠标等设备配置。

我们可以通过 "hexdump /dev/input/xx" 命令识别正确的触摸或鼠标设备文件名。

```
/* framebuffer */
#define FB_DEVICE_FILENAME "/dev/fb0"

/* tslib */
#define TS_DEVICE_FILENAME "/dev/input/event0"

/* 键盘 */
#define KB_DEVICE_FILENAME "/dev/input/event1"

/* 鼠标 */
#define MICE_DEVICE_FILENAME "/dev/input/mouse0"
```

图1 配置设备名称

编译打包并运行应用程序

1. 编译awtk

在 Ubuntu 打开 awtk-linux-fb 目录并输入 scons 命令编译 awtk。

2. 编译应用程序

同样在 awtk-linux-fb 目录输入命令 scons APP=../awtk-hello-demo，“APP=”后面为应用程序相对路径。

提示“scons: done building targets.”代表编译成功，若编译失败考虑 SDK 版本不配套或 awtk_config.py 中工具链路径设置不正确等情况。

3. 发布并运行应用程序

在提示编译成功之后，还需要执行下面命令生成 realease 文件夹。

./release.sh ../awtk-hello-demo/res demo

将 release 文件夹通过 scp 或其它工具将其拷贝到板子上，并在板子上运行命令：

./release/bin/demo

若提示 can't load library 'xxx.so'，需添加动态库查找路径，在板子控制台输入 export LD_LIBRARY_PATH=so 所在文件夹的路径，再运行应用即可。

新版AWTK SDK配置与编译方式

若使用 Github 最新版的 SDK，可以使用另外一种配置与编译方式，这种方式可以避免了在一堆配置里面寻找配置项的麻烦。

1. 配置交叉编译工具链路径

在 awtk-linux-fb 目录创建名为 awtk_config_define.py 的文件，将 TOOLS_PREFIX 等参数及对应的路径写入该文件中。

awtk_config_define.py 还可以由脚本生成并支持修改其它参数，具体可以看最新版 awtk-linux-fb 的 README 文档。

2. 编译 awtk

在 Ubuntu 打开 awtk-linux-fb 目录并输入 scons 命令编译 awtk。

3. 编译与发布应用

去到应用程序目录下，使用 scons LINUX_FB=true 命令编译应用。接着输入 python ./scripts/release.py 生成 release 文件夹。

将 release 文件夹通过 scp 或其它工具将其拷贝到板子上，并在板子上运行命令：

./release/bin/demo

链接错误解决思路

若在编译过程中遇到‘undefined symbol’错误提示，需检查 awtk_config.py 脚本 OS_LIBS 是否添加链接库，OS_CPPPATH 与 OS_LIBPATH 是否包含头文件与库文件搜索路径。

到这里，我们就完成了把 AWTK 应用程序应用部署到嵌入式 Linux 的操作，大家如果在移植过程中遇到了什么错误，可以在评论区或是加入官方交流群进行讨论。

系列往期文章

1.【从 0 开始创建 AWTK 应用程序】开发及调试环境搭建

2.【从 0 开始创建 AWTK 应用程序 】 创建应用程序并在模拟器运行

【AWTK开源智能串口屏方案】方案介绍和工作原理

ZLG 致远电子 2023-12-20 11:38:26

本篇文章介绍一下 AWTK 开源智能串口屏方案的基本原理和实际使用效果，包括主要特点、应用场景、工作原理以及简单的 Demo 演示。

引言：AWTK-HMI 是基于 AWTK 与 AWTK-MVVM 开发的低代码智能串口屏方案，本系列文章介绍如何从零开发 HMI 程序，包括搭建开发环境、创建 HMI 运行时工程、修改应用界面以及开发 MCU 程序。

AWTK开源智能串口屏方案简介

AWTK 开源智能串口屏方案（Gitee 仓库地址：gitee.com/zlgopen/awtk-hmi），仓库包含了 MCU 端工程（含 PC 模拟器）、MCU SDK、HMI 端工程（含 PC 模拟器）以及多个串口屏应用示例，并配有说明文档。

其中 HMI 端工程（又称为运行时）运行在串口屏硬件上，负责界面显示和人机交互，用户可替换里面的资源文件切换为不同的应用。MCU 端工程则是用户自己的主控程序，通过调用 MCU SDK 的 API，可以与 HMI 端通过串口或网口通信。MCU 可以主动获取、设置串口屏的变量值，或接收串口屏的按键事件，实现双向交互。

图 1 MCU端与串口屏端连接方式

1. 主要特点

1.1 HMI端

- 开放源码，免费商用，从底层到应用程序全程可控；
- 基于 AWTK 和 AWTK-MVVM 实现强大的 GUI 功能；
- 强大的界面设计器 AWStudio，拖拽式开发所见即所得；
- 界面设计与绑定数据（变量）简单，无需编写任何代码；
- 支持通过串口更新 UI 资源，无需重新编译工程；
- 支持在 PC 上模拟运行。

1.2 MCU端

- 提供 MCU SDK 与 HMI 端通信，使用简单无需了解通信协议；
- 只需使用三组函数：获取变量、设置变量、监视事件；
- 无需指定物理地址，变量可用中文名；
- 支持串口和 TCP 通信；
- 提供 MCU 模拟器模拟与串口屏的交互。

2. 应用场景

本方案假设有两类用户：串口屏厂商与普通用户。厂商把 HMI 端工程（运行时）烧写固化到串口屏硬件上；用户买到串口屏后，通过 AWStudio 开发自己的界面应用并把生成的资源文件替换到串口屏上，再开发自己的主控 MCU 程序，最后通过 MCU SDK 控制串口屏。

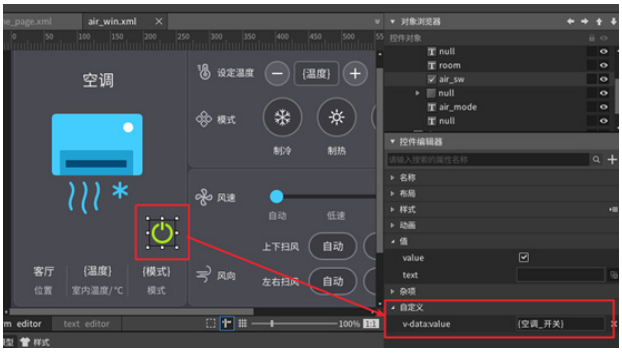


图 2 使用AWStudio开发界面应用

3. 工作原理

串口屏 HMI 工程实际上是一个 AWTK+MVVM 的程序（运行时），这个程序可以加载 res 文件夹的 UI 资源文件（UI-XML、字体、图片等），并显示在屏幕上。由于使用 MVVM 框架，界面上的可变内容可以绑定到特定的变量名。

串口屏作为通信服务端启动，而 MCU 端程序使用我们封装好的通信协议（MCU SDK），作为通信客户端与串口屏连接，之后就可以直接通过变量名进行交互。

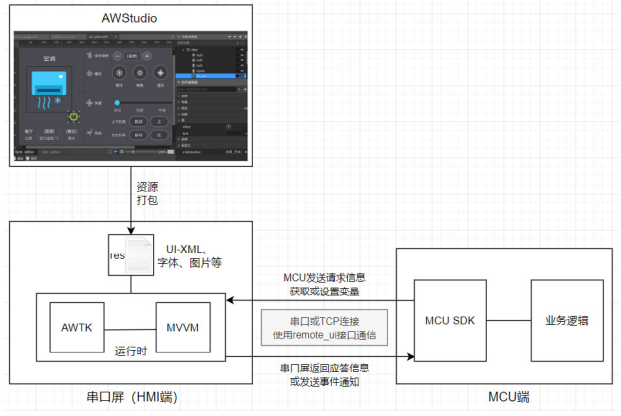


图 3 工作原理

后期用户如果想更新串口屏上的应用，直接用 AWStudio 设计新界面，打包替换 res 文件夹的资源文件然后重启串口屏即可，无需重新编译 HMI 工程。

Demo演示

下面使用 PC 上的 MCU 模拟器和串口屏模拟程序来对 AWTK-HMI 的一些功能进行演示。

1. MCU获取串口屏事件

用户在串口屏上改变界面数据时，会自动将事件通知 MCU 端，如下图所示：

图 4 串口屏界面改变自动通知MCU

2. MCU设置串口屏变量属性

在 MCU 中输入属性名称、属性类型和值再点击“设置属性”按钮即可看到串口屏界面中对应的属性改变，如下图所示：

图 5 MCU修改数据自动更新串口屏界面

3. 串口屏实机效果

下面是串口屏在实际板子上的运行效果，包括使用 MCU 操控串口屏界面以及手动操作串口屏界面两种方式：

图 6 串口屏实际上板演示

【产品应用】
如何用“搭积木”方式快速搭建智慧工厂大屏

ZLG 致远电子 2023-12-05 11:37:06

传统的工厂管理方式里，管理层缺乏一个可视化的工具来实现对工厂生产的整体管理。通过 ZWS 云低代码平台，可以使用“搭积木“的方式快速搭建出一个智慧工厂可视化大屏，实现工厂管理的实时高效。

ZWS 云低代码平台是一款以图形化、拖拽方式并辅以少量代码就能快速搭建企业数字化应用的工具平台。我们提供了常用的基础组件、工业组件（用于工业生产场景）、大屏组件（用于展示大量数据）以及业务组件（物联网相关业务），无需深入了解编程语言和开发技术，您只需简单地拖拽组件即可快速搭建物联网 Web 应用。

CATCOM-100 是一款工业级智能网联终端，通过 RS-485 等接口，可快速实现设备联网。终端采用 Cat.1 通信标准，最高支持上行 5Mbps、下行 10Mbps 的通信速率，与 Cat.4（LTE 4G）网络共用基站，保障信号覆盖范围。CATCOM-100 适用于环境监测、电力配电、智慧工厂、智慧农业等多种应用场景。

点击购买

场景应用

在工厂管理里，管理层经常需要关注工厂的生产情况、设备运行状态等数据；传统的管理方式，管理层需要通过一摞摞、一叠叠、堆成小山的生产报告、纸质表单才能了解到工厂情况，存在着数据滞后、对工厂情况了解不够全面直观等问题。使用 ZWS 云低代码平台使用“搭积木”方式搭建智慧工厂大屏，能够让工厂管理层能够基于工厂管理的实际需求，通过“拖拉拽”组件的方式自定义工厂大屏，实现管理的直观高效，更能依托大屏数据进行下一步生产决策。

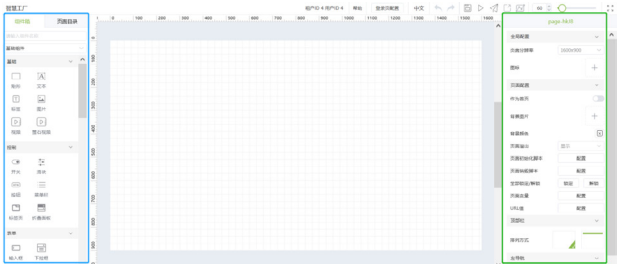
智慧工厂大屏搭建流程

1. 了解ZWS云低代码页面布局

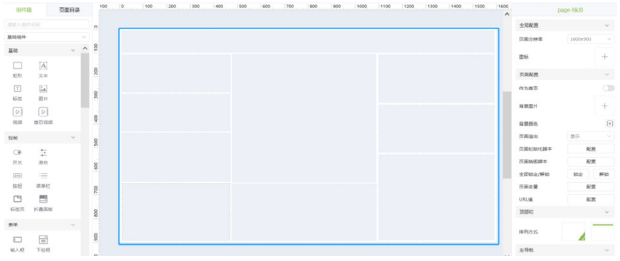
中间是一块空白画布，搭建大屏和页面的主流程都在画布上进行；

左侧是组件栏，提供了各类常用组件，例如基础组件、控制组件、表单组件等，用户可以直接从组件栏中选择所需组件拖拽到画布中；

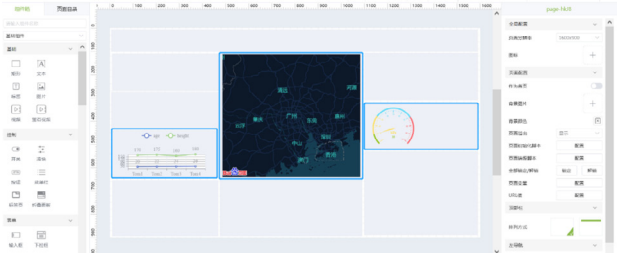
右侧是配置栏，既能实现全局配置，也能对组件的样式和交互进行配置。



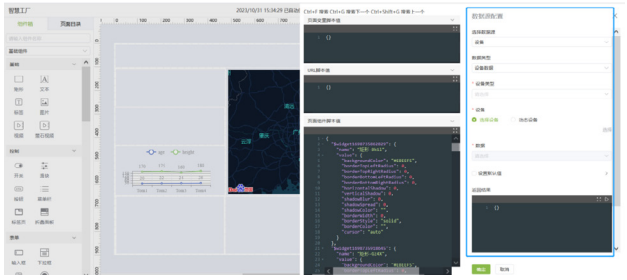
2. 搭建智慧工厂大屏时，可以先在画布区进行分区排版，确定组件间的位置关系。



3. 拖拽合适的组件来搭建大屏。例如，可以用折线图来表达工厂能耗的变化情况，使用仪表盘来表示当前总功率，使用地图来展示工厂的地理位置。



4. 双击组件，对组件进行数据源的配置，例如仪表盘组件，绑定数据源为“设备”，配置数据类型为“设备数据”，或者通过写脚本方式处理数据源数据。配置好数据源后，就能够在大屏上看到一个关联到真实场景数据的组件。



5. 依照此流程，我们能够依次搭建出【地图】、【酷炫文字】、【仪表盘】、【折线图】、【柱状图】等组件。

6. 最终搭建出来的智慧大屏效果。

小结

以上是通过 ZWS 云低代码平台使用“搭积木”方式快速搭建出一个智慧工厂大屏的全流程。

使用这种方式搭建大屏，让即使没有代码基础的管理人员也能快速搭建出一个满足需求的可视化大屏，同时让管理人员可以更专注于从业务角度去设计大屏的内容，围绕工厂生产的重点任务和核心指标去进行整体效果的呈现。

【产品应用】基于ZWS云对LoRa网关与节点的通信统计

ZLG 致远电子 2023-12-12 11:34:03

通过 LoRa 网关，可以将各种传感器节点设备的数据采集，并上传到物联网云平台，实现智能化管理与可视化监控。本文将介绍在 ZWS 物联网云平台如何分析 LoRa 网关与节点的通信情况。

应用场景

点击购买

GLCOM-NET 是 ZLG 致远电子开发的一款智能无线数据通信网关，采用 LoRa 无线技术，实现了自组网、无线数据透明传输至 TCP、UDP、MQTT、HTTP 及 RS485/232 等功能，大幅简化无线产品的开发过程，使您的产品快速投入市场，广泛用于工业物联网领域。

在工业自动化、环境监测、智慧消防等领域，LoRa 网关是连接物联网终端设备与云平台的桥梁角色，LoRa 网关与节点的通信，是实现工业设备远程监控和智能化管理的关键。ZWS 物联网云平台支持分析 LoRa 网关（GLCOM-NET）与节点的通信情况，可以从数据帧统计、信道负载率、空闲信道扫描以及数据异常日志四个方面监测分析。

前期准备

1.LoRa 网关 GLCOM-NET；

2. 注册 / 登录 ZWS 物联网云平台。

将网关接入 ZWS 云平台，就可以在云平台中远程查看分析 LoRa 网关与节点的通信情况。

基于ZWS物联网云平台对LoRa网关与节点的通信分析

1. 统计数据帧

可以远程统计分析上下行数据帧的帧类型、帧速率的占比。

2. 统计信道负载率

可以实时统计 LoRa 网关的信道负载率，分析上下行负载情况。

3. 空闲信道扫描

可以远程扫描空闲信道，方便设置合适的频段。

4. 监测数据异常日志

可以远程监测 LoRa 网关与节点数据和行为的异常，比如：节点 MIC 错误、非法地址错误、DEVNOCE 错误、节点 join 过于频繁、节点 confirm 帧上报过于频繁、节点帧计数器不连续等异常日志监测。

基于 ZWS 物联网云平台，可以远程分析 LoRa 网关与节点的通信情况，还可以对业务终端设备进行远程监测管理，实现方便高效的运营管理。

GLCOM-NET

点击购买

【产品应用】储能EMS网关如何快速接入智慧储能云平台

ZLG 致远电子 2023-12-14 11:38:46

数据采用是一个常用的功能。在 AWTK 开源串口屏中，内置数据采样模型，只需设计用户界面即可实现采样数据的显示和管理。

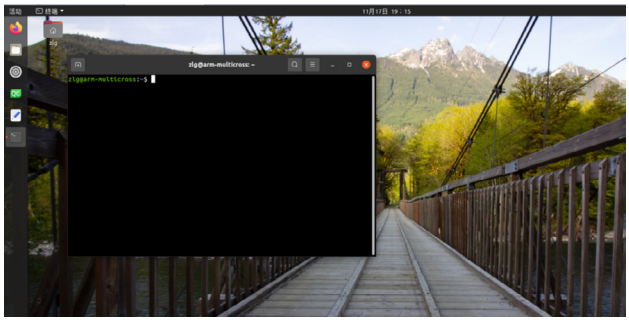
无需手动配置交叉编译器工具链，只需一个命令就能完成储能 EMS 网关接入云示例程序编译，快速接入 ZWS 智慧储能云平台。

储能EMS网关

EM 系列储能边缘智能网关是致远电子专为新能源储能系统设计的高性能、多接口通讯管理设备，拥有 EM500、EM1000 等多个型号，既能满足储能系统的本地能源管理需求，同时可以实现上云服务，适配工商储能等不同场景。

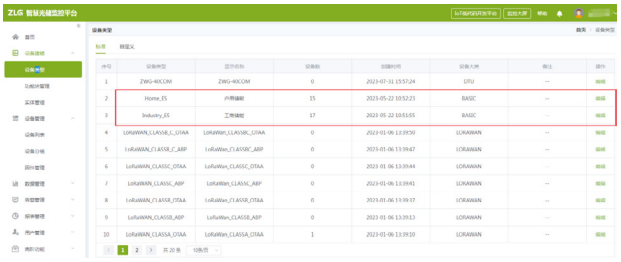
储能EMS网关交叉编译环境

交叉编译环境，即安装、配置交叉编译工具链。储能 EMS 网关提供了虚拟机镜像，已经预安装了配套的交叉编译环境，用户开机后即可使用。

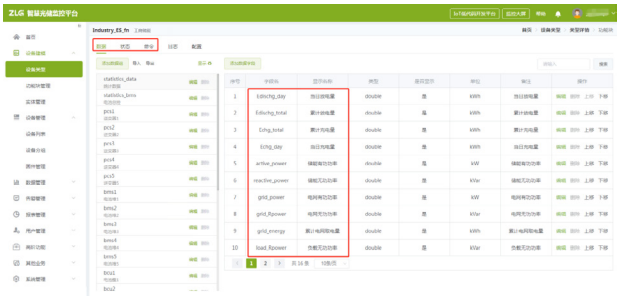


储能网关设备建模

致远电子智慧储能云平台预定义了工商储能和户用储能的设备模型 Industry_ES、Home_ES。



储能设备模型中还定义了储能设备的通用数据字段、状态字段、命令字段。



一键交叉编译储能设备端示例程序

ZWS MQTT SDK 对储能 EMS 网关做了专属化支持，ZWS MQTT SDK 提供了储能设备示例程序，用户无需手动配置交叉编译器工具链，只需一个命令就能完成储能示例程序的编译。网关 EM500/EM1000 交叉编译 mqtt SDK 的步骤如下：

1. 安装 cmake 工具。
2. 执行 cmake 命令。

先切换到 SDK 的目录，然后在开发环境中执行 cmake 命令。

如果储能网关是 EM500 型号，执行命令：

o

cmake -DEM500=1 .

如果储能网关是 EM1000 型号，执行命令：

o

ocmake -DEM1000=1 .

3. 使用 make 命令进行编译。

编译成功后，可在 bin 目录中看到生成的多个二进制文件。

```
drwxr-xr-x  2 root root          4096 Nov 16 19:19 ./
drwxr-xr-x 16 root root          4096 Nov 16 19:15 ../
-rwxr-xr-x  1 root root        219496 Nov 16 19:11 appuser_example*
-rwxrw-rw-  1 root lvjianchao   55296 Aug 30 14:32 appuser_example.exe*
-rwxrw-rw-  1 root lvjianchao   64284 Aug 30 14:31 client.lib*
-rwxrw-r--  1 root lvjianchao  110592 Aug 30 13:44 client.pdb*
-rwxr-xr-x  1 root root       1839472 Sep 13 16:43 client_test*
-rwxr-xr-x  1 root root        225448 Nov 16 19:16 device_example*
-rwxrw-rw-  1 root lvjianchao   64512 Aug 30 14:32 device_example.exe*
-rwxr-xr-x  1 root root        230224 Nov 16 19:19 device_example_Home_ES*
-rwxr-xr-x  1 root root        218624 Nov 16 19:11 device_example_Industry_ES*
-rwxr-xr-x  1 root root        192648 Nov 16 19:16 download_firmware_example*
-rwxrw-rw-  1 root lvjianchao   41984 Aug 30 14:32 download_firmware_example.exe*
-rwxr-xr-x  1 root root        198784 Nov 16 19:16 download_package_example*
-rwxrw-rw-  1 root lvjianchao   45568 Aug 30 14:32 download_package_example.exe*
-rwxr-xr-x  1 root root         32712 Nov 16 13:41 echoc*
-rwxr-xr-x  1 root root        226096 Nov 16 19:11 gateway_example*
-rwxrw-rw-  1 root lvjianchao   64000 Aug 30 14:32 gateway_example.exe*
-rwxr-xr-x  1 root root         62912 Nov 16 13:41 http_auth*
-rwxr-xr-x  1 root root         48376 Nov 16 13:41 http_download*
-rwxr-xr-x  1 root root        190200 Nov 16 19:11 http_transfer_chunk*
-rwxrw-rw-  1 root lvjianchao   43520 Aug 30 14:32 http_transfer_chunk.exe*
-rwxrw-rw-  1 root lvjianchao   16928 Aug 30 14:31 jsmn.lib*
-rwxrw-r--  1 root lvjianchao   86016 Aug 30 13:44 jsmn.pdb*
-rwxrw-rw-  1 root lvjianchao   16960 Aug 30 14:31 jsmn_wrapper.lib*
-rwxrw-r--  1 root lvjianchao   86016 Aug 30 13:44 jsmn_wrapper.pdb*
-rw-r--r--  1 root root         81882 Nov 16 19:11 libclient.a
-rw-r--r--  1 root root         25064 Nov 16 13:41 libjsmn.a
-rw-r--r--  1 root root         25064 Nov 16 13:41 libjsmn_wrapper.a
-rw-r--r--  1 root root        303524 Nov 16 13:41 libnetwork.a
-rwxr-xr-x  1 root root        105240 Nov 16 13:41 mqtt_demo*
-rwxrw-rw-  1 root lvjianchao  178854 Aug 30 14:32 network.lib*
-rwxrw-r--  1 root lvjianchao  159744 Aug 30 13:48 network.pdb*
-rwxr-xr-x  1 root root       1795792 Sep 13 16:43 params_test*
-rwxr-xr-x  1 root root        190088 Nov 16 19:16 upload_file_example*
-rwxrw-rw-  1 root lvjianchao   41472 Aug 30 14:32 upload_file_example.exe*
```

其中，

device_example_Industry_ES 是工商储能的示例程序

device_example_Home_ES 是户用储能的示例程序

启动工商储能示例程序

./bin/device_example_Industry_ES Industry_ES your_dev_id your_dev_password

智慧储能云平台收到数据后，对数据进行统计和分析。

示例程序代码概述

设备连接云端服务器。

```
static IoT_Error_t connect_server(mqtt_client_t *mq, ZLG_IoT_Client *mqclient, dev_meta_info_t* me
{
  IoT_Client_Init_Params* p_mqttInitParams = &iotClientInitParamsDefault;
  IoT_Client_Connect_Params* p_connectParams = &iotClientConnectParamsDefault;
  IoT_Error_t rc = FAILURE;
  mq->ctx = mqclient;
  mq->publish = mqtt_publish;

  zlg_mqtt_info_init(&mqtt_info);
  if (!get_mqtt_info(&mqtt_info, ZLG_IOT_AUTH_HOST, ZLG_IOT_TERMINAL_AUTH_PATH, ZLG_IOT_AUTH_PORT,
  {
    IOT_ERROR("get mqtt info error");
    return FAILURE;
  }
  IOT_INFO("\nZLG IoT SDK Version %d.%d.%d-%s\n", VERSION_MAJOR, VERSION_MINOR, VERSION_PATCH, VER
  p_mqttInitParams->enableAutoReconnect = false;
  p_mqttInitParams->pHostURL = mqtt_info.host;
#ifdef WITH_MQTT_SSL
  p_mqttInitParams->port = mqtt_info.sslport;
  //p_mqttInitParams->port = 4883;
#else
  p_mqttInitParams->port = mqtt_info.port;
#endif
  p_mqttInitParams->mqttCommandTimeout_ms = 20000;
  p_mqttInitParams->tlsHandshakeTimeout_ms = 50;
  p_mqttInitParams->isSSLHostnameVerify = false;
  p_mqttInitParams->disconnectHandlerData = NULL;
```

示例程序中模拟生成了储能设备数据。

```
BOOL Industry_ES_dev_get_statistics_data(params_t* params) {
    params->size = 1; /*保留版本号*/

    // 每条记录的产生时间，云端要求时间精度为ms;
    // 可省略该字段，此时以云端收到设备数据的时间为准
    params_append_uint(params, STR_KEY_TIME, time(0)*1000);
    /*TODO: 请在这里设置真实数据*/
    // 以下数据字段对应云端Industry_ES类型中statistics_data分组下面定义的字段
    params_append_float(params, "Edischg_day", 100);
    params_append_float(params, "Edischg_total", 110);
    params_append_float(params, "Echg_day", 120);
    params_append_float(params, "Echg_total", 110);
    params_append_float(params, "active_power", rand()%1000);
    params_append_float(params, "reactive_power", rand()%1000);
    params_append_float(params, "grid_power", rand()%1000);
    params_append_float(params, "grid_Rpower", rand()%1000);
    params_append_float(params, "total_power", rand()%1000);
    params_append_float(params, "grid_energy", rand()%1000);
    params_append_float(params, "load_Rpower", rand()%1000);
    params_append_float(params, "load_power", rand()%1000);
    params_append_float(params, "dump_energy", rand()%1000);
    params_append_float(params, "feed_energy", rand()%1000);
    params_append_float(params, "photovoltaic_energy", rand()%1000);
    params_append_float(params, "photovoltaic_power", rand()%1000);

    return TRUE;
}
```

储能设备接收云端下发的命令如削峰填谷策略。

```
// 云端下发(云端配置的)[削峰填谷]策略给设备
static BOOL Industry_ES_peak_shaving_exec(command_t* cmd, struct _client_t* client, params_t* para
  int cmdid = 0;
  BOOL ret = TRUE;
  const char* message = "ok";
  const char* cfg_info = NULL;

  goto_done_if_fail(cmd != NULL && client != NULL && params != NULL, BAD_PARAMETERS);
  goto_done_if_fail(params_get_string(params, "cfg_info", &cfg_info), BAD_PARAMETERS);
  printf("Industry_ES_peak_shaving_exec cfg_info:%s\n",cfg_info);
  // 把云端下发的[削峰填谷]策略保存到本地(这里模拟保存在内存变量Industry_ES_peak_shaving_buff中)
  memset(Industry_ES_peak_shaving_buff, 0, sizeof(Industry_ES_peak_shaving_buff));
  memcpy(Industry_ES_peak_shaving_buff, cfg_info ,strlen(cfg_info));
  (void)cmd;
  printf("%s\n", __func__);
  params_dump(params);

done:
  if (params_get_int(params, STR_KEY_CMDID, &cmdid)) {
    client_reply_cmd(client, cmdid, ret, message, STR_NULL);
  }
  return ret;
}
```

储能设备上报数据、设备状态给智慧储能云平台。

```
  // 获取模拟的工商储能的统计数据(statistics_data)然后上报到云端，数据格式是key1\0vluae1\0key2\0vluae
  request_init(&params, buff, sizeof(buff));    // 初始化buff内容为空
  Industry_ES_dev_get_statistics_data(&params);
  const char* group_statistics_data = "statistics_data";  // 数据分组对应Industry_ES定义中的数据分
  if (!client_report_data_multi(&a_device, group_statistics_data, &params)) {
    //发送失败重发，这里只是简单示例，具体重发机制由调用者自行实现（也可以在mqtt_client.c文件处理重发）。
    //client_report_data_multi(&a_device, group_statistics_data, &params);
  }
```

【产品应用】如何一键将EPCM3568边缘网关接入ZWS云

ZLG 致远电子 2023-12-19 11:37:51

EPCM3568 智能边缘网关支持专属化 ZWS MQTT SDK，提高研发效率，快速接入物联网云，实现数字化智能化管理。

EPCM3568工业智能边缘网关

EPCM3568B-LI/EPCM3568C-LI 5G 智能边缘计算网关 配置双千兆以太网口，并支持 WiFi，Bluetooth，4G/5G 无线传输；拥有丰富灵活的接口扩展，包括 USB2.0、USB3.0、HDMI、LVDS、RS485、CAN、GPIO、ADC 等，广泛适用于工业控制、储能、环境监测等领域。

点击购买

EPCM3568开发环境虚拟机镜像

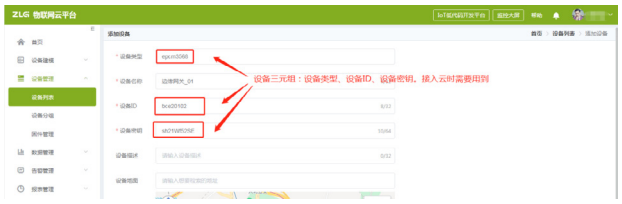
为节约用户重复配置开发环境的时间、避免因开发环境设置不当影响研发进度，EPCM3568-LI 提供了开箱即用的基于 Ubuntu 虚拟机镜像的交叉编译开发环境。

一键交叉编译EPCM3568智能边缘网关示例程序

arm linux 设备要接入 ZWS 云，要先手动配置 cmake 依赖的编译器工具链，将对应工具链安装路径配置到 cross-linux.mk，这个手工配置过程繁琐还容易出错。

为了提高用户的研发效率，ZWS MQTT SDK 对 EPCM3568 智能边缘网关做了专属化支持，用户无需手动配置交叉编译器工具链，只需一个命令就能完成边缘网关示例程序的编译。EPCM3568 智能边缘网关交叉编译 mqtt SDK 的步骤如下：

1. 在 ZWS 物联网云平台建模，创建设备类型，定义设备类型的数据、状态、命令等字段，并创建设备。



2. arm 版本的 ubuntu 上安装 cmake，然后执行 EPCM3568 边缘网关的 cmake 命令。

cmake -DEPCM3568=1 .

3. 执行编译 make 命令。

make

编译成功后，可在 bin 目录中看到生成的多个二进制文件。

```
zlg@arm-multicross:~/ws/zws_sdk_r2/mqttproto_V2.0.1.230913/bin$ ll
总用量 2448
drwxrwxr-x 2 zlg zlg   4096 11月 10 16:41 ./
drwxrwxrwx 8 zlg zlg   4096 11月 10 17:33 ../
-rwxrwxr-x 1 zlg zlg 257792 11月 10 16:41 appuser_example*
-rwxrwxr-x 1 zlg zlg 259744 11月 10 16:41 device_example*
-rwxrwxr-x 1 zlg zlg 225072 11月 10 16:41 download_firmware_example*
-rwxrwxr-x 1 zlg zlg 232832 11月 10 16:41 download_package_example*
-rwxrwxr-x 1 zlg zlg  36512 11月 10 16:41 echoc*
-rwxrwxr-x 1 zlg zlg 264624 11月 10 16:41 gateway_example*
-rwxrwxr-x 1 zlg zlg  70472 11月 10 16:41 http_auth*
-rwxrwxr-x 1 zlg zlg  52616 11月 10 16:41 http_download*
-rwxrwxr-x 1 zlg zlg 227216 11月 10 16:41 http_transfer_chunk*
-rw-rw-r-- 1 zlg zlg 100218 11月 10 16:41 libclient.a
-rw-rw-r-- 1 zlg zlg  28440 11月 10 16:41 libjsmn.a
-rw-rw-r-- 1 zlg zlg  28440 11月 10 16:41 libjsmn_wrapper.a
-rw-rw-r-- 1 zlg zlg 354436 11月 10 16:41 libnetwork.a
-rwxrwxr-x 1 zlg zlg 119744 11月 10 16:41 mqtt_demo*
-rwxrwxr-x 1 zlg zlg 218208 11月 10 16:41 upload_file_example*
```

4. 启动接入示例程序。

将网关三元组作为参数启动 device_example 程序：

./bin/device_example dev_type dev_id dev_password

三元组是指设备类型、设备 ID、设备密钥，启动程序时，要根据实际定义的值进行替换。比如：设备类型定义为 EPCM3568 时，将 dev_type 替换成 EPCM3568。（设备 ID 和设备密钥同理）

边缘网关成功上传数据后，根据 demo 示例，开发具体的业务数据，就可以在 ZWS 物联网云平台进一步远程管理和数据分析。

EPCM3568系列

点击购买

【产品应用】基于ZWS云和LoRa网关的环境监测“1+1>2”方案

ZLG 致远电子 2023-12-21 12:39:46

环保部门在治理环境污染问题时，面临的一个重要挑战是如何实时掌握不同地区的环境情况。为了解决这一问题，本文将介绍一种基于 ZWS 云和 LoRa 网关的环境监测“1+1>2”方案。

传统的环境监测方法需要人工现场采集数据，不仅监测点分散、分布面积广，而且有些监测点位于环境恶劣地区，无法在短时间内获取准确的污染排放情况。环保部门需要一个智能化的环境监测系统，能够实时监测不同地区的环境情况，以便及时处理环境污染问题。传统的环境监测方法无法满足这一需求，因此需要一种新型的环境监测方案，能对环境情况实现动态数据展现和污染物预警，为环保部门的环境管理和污染防止提供数据支撑。

在新的方案我们采用了 GLCOM-NET 智能无线数据通信网关，该网关采用 LoRa 无线技术，实现了自组网、无线数据透明传输至 TCP、UDP、MQTT、HTTP 及 RS485/232 等功能，拥有低功耗、长距离等优点。

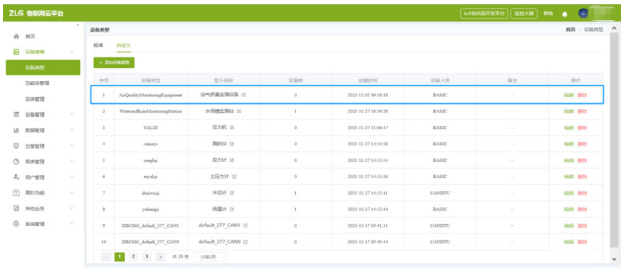
而 ZWS 云则是一款通用的物联网云平台，能够提供数据采集、存储、分析和可视化等功能，广泛应用于环境监测、智慧农业、智慧家居等领域，在 ZWS 云平台上，能对环境监测数据实现动态展示、动态预警和数据分析等。

将 ZWS 云平台和 GLCOM-NET 网关结合起来，连接各种环境监测设备，实时监测各种环境数据，如 PM2.5、温度、湿度、一氧化碳、水质情况、土壤情况等。通过 GLCOM-NET 网关将这些数据传输到 ZWS 云平台，再通过电脑、小程序等终端实时查看环境数据。这样，环保部门就可以方便地了解不同地区的环境情况，监测环境污染情况，及时治理环境污染问题。

功能介绍

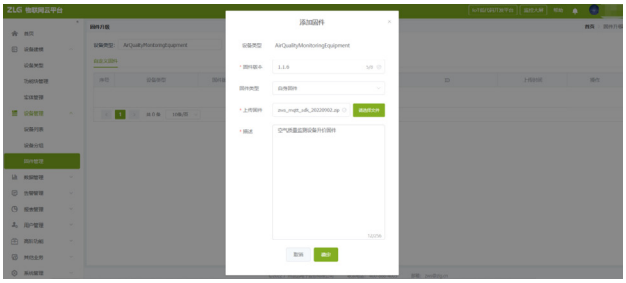
1. 设备管理

ZWS 物联网云平台能够快速接入多种协议的环境监测设备，如空气质量监测设备，实现环境数据的动态展示。



2. 远程升级/配置

支持云端固件升级功能，能对设备进行远程升级和配置。环境监测设备分布区域广，有些分布在远离城区、位置偏远地区或高海拔地区地区，人工现场升级面临诸多不便，使用远程升级 / 配置功能，能实现“足不出户”即可完成环境监测设备和 LoRa 网关升级和配置。



3. 数据统计分析

ZWS 云平台提供数据大盘和报表功能，数据大盘由多个看板组成，环保部门可以根据实际需求自行创建曲线图、折线图、柱状图等多形态看板，基于环境监测数据进行环境质量趋势分析，报表功能能对监测数据形成多维度的报表。ZWS 云平台数据统计分析功能能够帮助环保部门挖掘数据价值，为环境治理提供数据支撑。

4. 告警管理

ZWS 云平台告警管理功能，能够基于 LoRa 网关实时上传的监测数据，对超标污染物发出数值超标告警，提醒环保部门对相关污染物进行治理。

【产品应用】LoRa & ZWS云应用(1)-智能抄表方案篇

ZLG 致远电子 2023-12-28 11:36:53

智能远程抄表是能耗监测和管理的重要手段之一，可以通过采集水电气表计设备数据，并将这些数据传输到云端进行处理分析，本文将介绍基于 LoRa 网关节点和 ZWS 物联网云的智能远程抄表应用方案。

远程抄表如何实现？智能远程抄表是应用了物联网技术，采集 - 传输 - 解析 - 应用。采集设备数据，然后再将数据传输到智能系统，系统再根据指令自动信息反馈回来。实现远程抄表过程，其实就是一个数据传输的过程。当然由于不同场景不同场合，适用的数据传输方式不同，就需要不同的组网方案。

LoRa组网方案

LoRa 节点模块可内置于水电气表计中，LoRa 节点传输水电气表数据给 LoRa 网关，再通过 LoRa 网关上传数据至云平台的组网方案，无需布线，只要采集设备位置有信号就可以实现数据传输，适用分散安装的环境。

致远电子的 LoRa 节点模组 ZSL420，可以接入表计设备中，实现采集总用量、电能、水量等数据，并通过 LoRa 网关 GLCOM-NET 将数据上传到 ZWS 物联网云平台进行监测管理。

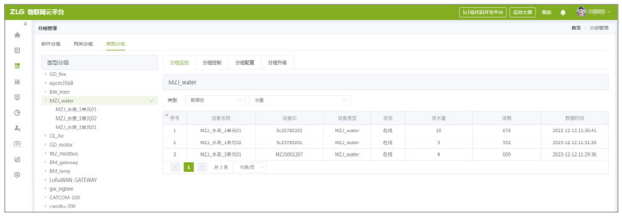
点击购买

点击购买

ZWS物联网云平台功能

1. 支持多厂家、多协议的水电气表计设备接入。

2. 支持远程自动抄表，查看抄表数据，节省人力提高效率。



3. 支持数据可视化，统计包括日用水量、月用水量、季度用水量等，以便于用户和水务公司进行用水分析。

ZWS 物联网云平台加强了适用性，可以用在多种品牌的水电气表计。节省了人员费用，不需要一个个到现场去查看，只需要远程自动抄表，就可以看到数据。能够在 PC 端和手机端看到抄表数据，清楚的了解费用使用的过程，把抄表人员从低端重复性劳动中解放出来。

【产品应用】Coral3568如何软硬件过滤can帧及优化?

ZLG 致远电子 2023-12-07 11:37:52

CAN 总线调试时，根据数据的重要性，接收端可以专注于接收重要消息，提高效率。CAN 报文帧是用一个标识符或一系列标识符来过滤的。可以使用 CAN 总线软件和芯片硬件过滤规则实现过滤功能。

软件过滤

```
struct can_filter filter[1];
/* 规则：可以接收 ID 为 0x200 数据帧和错误帧 */

filter[0].can_mask = CAN_SFF_MASK;
filter[0].can_id  = 0x200 & CAN_SFF_MASK;
filter[0].can_mask |= CAN_EFF_FLAG;

if (setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, &filter,
sizeof(filter)))
{
perror("setsockopt failed");
exit(EXIT_FAILURE);
}
```

软件的配置可以在系统中检查：

```
root@host:/root# ls /proc/net/can/
rcvlist_all  rcvlist_err  rcvlist_inv  reset_stats
rcvlist_eff  rcvlist_fil  rcvlist_sff  stats

root@host:/root# cat /proc/net/can/rcvlist_fil
receive list 'rx_fil':
 (any: no entry)
 (can0: no entry)
 device   can_id   can_mask  function  userdata   matches  ident
  can1      200    800007ff  0000000095327ce0  00000000674196b1
0  raw
 (can2: no entry)
```

此处 rcvlist_* 为 CAN 协议中的接收器：struct receiver，包含软件的过滤规则。

硬件过滤

芯片手册：

24.2.2 ACCEPTANCE FILTER

The acceptance filter performs filtering using the acceptance ID register and the ID mask register. The acceptance filter uses multiple-time filtering. It includes the old version receiving filter and five addition ID filters. Each ID filter pair has a Filter Mask register and a Filter ID register. Each filter pair is controlled by corresponding FILTER CTRL bit in AFR register. For each pair using ID register and ID mask register(controlled by acceptance filter register)sampling all the bits of the ID, it is compared with the ID register. It is not a comparison every time a bit is sampled. And don't check the bit in the ID mask register that is 1. Once all the ID bits have passed the filtering, the controller considers the frame data as the desired ID, thereby performing the next operation.

硬件过滤方式在驱动：drivers/net/can/rockchip/rockchip_canfd.c 中实现，具体代码：

```
static int rockchip_canfd_start(struct net_device *ndev)
{
 ......
 rockchip_canfd_write(rcan, CAN_INT_MASK, 0);

 /* RECEIVING FILTER, accept all */
 rockchip_canfd_write(rcan, CAN_IDCODE, 0);
 rockchip_canfd_write(rcan, CAN_IDMASK, CAN_RX_FILTER_MASK);
 rockchip_canfd_write(rcan, CAN_IDCODE0, 0);
 rockchip_canfd_write(rcan, CAN_IDMASK0, CAN_RX_FILTER_
MASK);
 rockchip_canfd_write(rcan, CAN_IDCODE1, 0);
 rockchip_canfd_write(rcan, CAN_IDMASK1, CAN_RX_FILTER_
MASK);
 rockchip_canfd_write(rcan, CAN_IDCODE2, 0);
 rockchip_canfd_write(rcan, CAN_IDMASK2, CAN_RX_FILTER_
MASK);
 rockchip_canfd_write(rcan, CAN_IDCODE3, 0);
 rockchip_canfd_write(rcan, CAN_IDMASK3, CAN_RX_FILTER_
MASK);
 rockchip_canfd_write(rcan, CAN_IDCODE4, 0);
 rockchip_canfd_write(rcan, CAN_IDMASK4, CAN_RX_FILTER_
MASK);
 ......
}
```

从以上代码可以看出，每次执行 CAN 软件都会重新配置 CAN 控制器寄存器不过滤任何数据。

通过 2 个寄存器来实现硬件过滤：CAN_IDCODEn、CAN_RX_FILTER_MASK

RK3568 CAN 控制器有 6 个过滤器，其中第一个过滤器默认使能，不受开关控制，其余过滤器需要使能对应的过滤器。可以在 can 软件运行时，再去配置硬件过滤方式，如：

```
root@host:/root# ip link set can1 type can bitrate 1000000
root@host:/root# ip link set can1 up
root@host:/root# candump can1 &
# 只接收 0x123 id 的帧，不受开关控制
root@host:/root# io -4 0xfe58003c 0x123
root@host:/root# io -4 0xfe580040 0x0
# 只接收 0x124 id 的帧，需开关控制
root@host:/root# io -4 0xfe580120 0x124
root@host:/root# io -4 0xfe580124 0x0
root@host:/root# io -4 0xfe58011c 0x1
```

系统优化

1. 发送异常返回

系统 CAN 默认发送队列：txqueuelen:10，有线网口该数值为 1000。更小的数值，实时性更强。

在大量数据发送时，write 函数经常异常返回，大部分原因是由于系统发送队列内存不足，可以使用以下指令增大发送队列：

```
root@host:/root# ip link set txqueuelen 500 dev can1
```

2. 接收数据丢失

接收队列不足表现在 read 得到的数据不全，大部分情况为数据 skb 已经提供到 receiver 队列中，但是应用来不及取出，导致最终占满所有可支配的内存大小，数据更新到了队列中错误的位置。

可使用以下指令调整接收队列大小：

```
root@host:/root# echo 1000000 > /proc/sys/net/core/rmem_max
root@host:/root# echo 1000000 > /proc/sys/net/core/rmem_default
```

Coral-EVa 为致远电子推出的 Coral3568 配套评估底板，同样功能强大，接口丰富。Coral-EVa 评估底板采用适配器供电，更方便实验室和研发办公室使用，HDMI、DP、USB、CAN、RS485、RS232、TTL UART、3.5mm 四线耳麦接口、Micro SD 卡槽、SATA、M.2、LVDS LCD、MIPI_DSI、MIPI-CSI、RTC 时钟、蜂鸣器等功能一应俱全。

3568系列

点击购买

【新品发布】集成式EtherCAT从站模块DPort-ECT，你见过吗？

ZLG 致远电子 2023-12-25 12:20:36

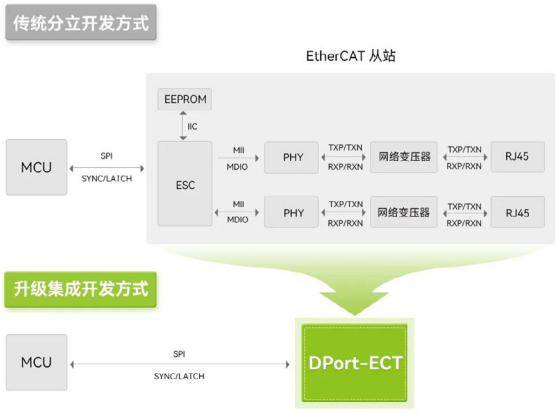
选型表

产品型号	DPort-ECT
网口数量	2
主控接口	SPI
接口电平	3.3V
速率	10/100Mbps 自适应
供电电压	3.3V±3%
DC 抖动	≤ 15ns
工作温度	-40°C ~+85°C
存储温度	-40°C ~+85°C
相对湿度	5%~90%(非冷凝)
电磁兼容	静电放电抗扰度试验：4 级 雷击（浪涌）抗扰度试验：3 级 电快速瞬变群脉冲抗扰度试验：3 级 射频场感应的传导骚扰看扰度试验：10V
评估板	EPC6450-DP（不含 DPort 模块）

立即选购

四合一集成式设计，快速开发EtherCAT从站

DPort -ECT 是四合一集成式 EtherCAT 从站模块，将 ESC 芯片、PHY 芯片、网络变压器、RJ45 插座高集成度封装。用户只需 MCU 控制 SPI 通信接口与从站配置引脚，轻松实现 EtherCAT 从站开发。



高速传输，低延迟通信

DPort-ECT 设计依照 EtherCAT 协议标准，数据传输快，通信延迟低，同步精度高，具有卓越的实时性能。

互联互通 ▼

2. 智慧医疗

目前健康医疗设备通常是可穿戴产品，通过蓝牙模块，可接收和分析接收到的健康数据，起到对数据实时监控的作用。典型产品如：手环、手表、健康体重秤、血糖计、血压计、脉搏血氧仪、心率带、活动传感器等。

图2 医疗保健产品

3. 智能照明

低功耗蓝牙智能 MESH 灯，利用低功耗蓝牙技术和 MESH 组网技术，只需在智能手机上安装一个 app，即可实现灯光的智能组网控制更加方便，灵活，同时支持遥控器设备。

图3 智能照明系统

4. HID设备的应用

采用低功耗蓝牙技术，相较传统蓝牙键盘鼠标，具有功耗更低、配对连接更加快速方便的特点。在客户现有的鼠标、键盘上可以增加 BLE 实现产品的快速升级。

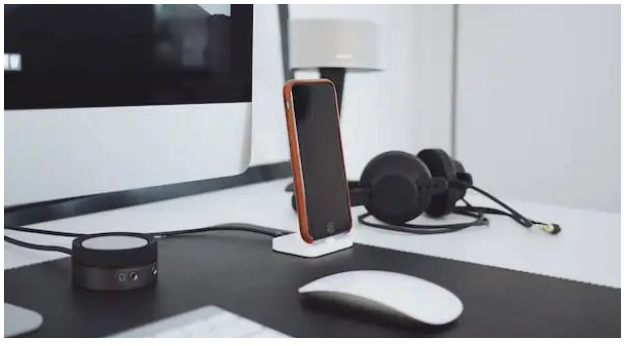


图4 HID设备

5. 智能家电

在传统家电上，例如豆浆机、电饭煲等，通过低功耗蓝牙控制，将其升级为更加智能的家电设备。通过智能手机对家电进行开关、预约等各种控制，还可以对家电设备进行故障自检、固件升级等功能。

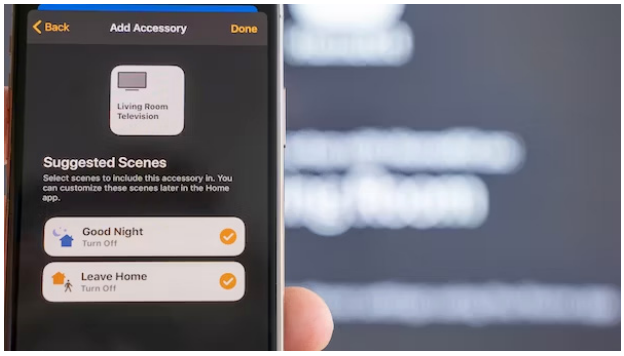


图5 智能家电产品

6. iBeacon应用

iBeacon 基于低功耗蓝牙广播给手机发送信息，能帮助商家优化经营策略，利用用户的地理位置提供服务。采用低功耗蓝牙性能更稳定可靠，功耗更低，让设备保持更长的工作时间。

图6 iBeacon应用场景

【技术分享】盘点几种RS-485方向切换方案

ZLG 致远电子 2023-12-04 11:36:40

RS-485 作为常见的总线之一，几乎每个工控设备都在用，其最大的特点就是需要进行方向控制，如果方向控制的时机不对，数据传输会出现错误。下面我们来讨论 RS-485 方向切换的几种方案。

使用软件控制方向

目前市面大部分的 RS-485 产品采用该方案，如图 1 所示，用致远电子的 SM4500 举例子，在空闲时，MCU 的 GPIO 脚输出低电平，此时 SM4500 处于接收状态，在准备发送数据之前，MCU 会拉高 GPIO 引脚，SM4500 处于发送状态，发送完毕之后，GPIO 脚重新处于低电平，SM4500 又处于接收状态。

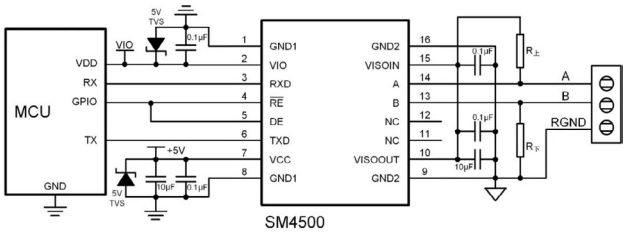


图1 软件控制方向方案

- 优点：无需外接多余的硬件外围，可以达到比较高的速率。
- 缺点：过于依赖软件的控制，要求软件对控制脚的反应要迅速，特别是在高波特率的情况下，不然会出现丢包的情况。

三极管自动切换方向

既然软件控制方向过于依赖软件，且还要多用一个 IO 口来控制方向，提高了软件成本，那有没有办法只用 TX 和 RX 就实现控制？其实只需要加个三极管即可，市面上很多低成本且速率要求不高的 485 转换器采用了该方案，电路如图 2 所示，依旧采用 SM4500 举例子，当 TX 引脚为 0 时，三极管不导通，DE 为高电平，进入发送模式，又因为 SM4500 的 TXD 引脚接地，那么此时 AB 之间的差分电平逻辑就为 0；当 TX 引脚为 1 时，三极管导通，RE 为低电平，进入接收模式。此时收发器的 A、B 引脚进入高阻态，因为上拉电阻 R 上、下拉电阻 R 下的作用，此时 AB 之间的差分电平逻辑为 1。

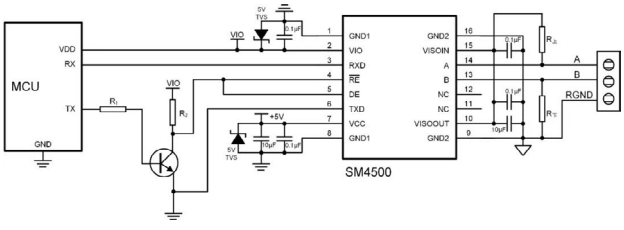


图2 三极管自动切换方向方案

- 优点：电路简单，只需一个三极管和两个电阻作为外围即可实现，无需软件介入。
- 缺点：三极管的开关延时过大，这会导致电路发送电平的延时时间较长，其次高电平的发送是通过外部上下拉电阻驱动的，电阻越大，上升沿越缓慢，驱动能力越弱，市面上类似电路的波特率一般不会超过 9600bps，如果要达到比较高的波特率，就要不断调试三极管和上下拉电阻参数，增大研发成本。

使用致远电子RSM485MG

那么有没有自带“自动收发切换”且能克服以上两种方案使用问题的产品呢?

致远电子最近推出的国产化自动流控型 RS-485 隔离收发器 RSM485MG 能完美解决以上问题，该隔离收发器将隔离 DC-DC 电路、信号隔离电路、RS-485 自动收发电路三合一封装起来，具有 100% 国产化、超小体积、带隔离输出电源脚、最多可连接 64 个节点、最大波特率 500kbps 等特点，应用电路如图 3 所示。

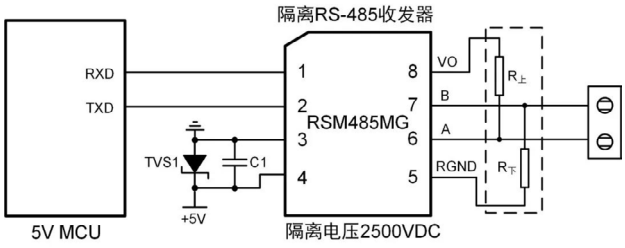


图3 RSM485MG经典电路连接图

- 优点：由模块自动切换方向，无需软件干预，波特率可达到 500kbps，体积小更容易嵌入板卡。与传统的设计相比，RSM485MG 产品内置完整的隔离 DC-DC 电路、信号隔离电路和 RS-485 总线收发电路，具备高集成度与可靠性。而且元器件 100% 国产化，满足国产化市场。
- 缺点：驱动能力较非自动流控产品弱。

RSM485MG的产品应用

气体探测和监控是当前比较普遍的应用，被广泛应用于工业生产、医学诊断、环境监测、国防等领域。气体检测探头将现场检测到的气体浓度转为标准信号，如 4-20mA 或者 RS-485 等信号，然后将信号输送到报警控制主机上进行统一显示，但是在工业领域，对通信有着抗干扰、传输距离及组网的要求，普遍采用 RS-485 通信方式，在一些场合还有隔离要求，我们的 RSM485MG 就完美契合该应用。其组网的解决方案如图 4 所示，主控

感知控制 ▼

方面推荐致远电子的 Cortex®-A7 平台的 M6Y2C 系列核心板，有着高达 8 路的 UART 接口和两路百兆的以太网接口，能外接更多设备，工业级品质使设备在户外工作更安全可靠，在 LVDS 接口显示方面我司也提供方案支持，在 RS-485 通信方面均采用 RSM485MG。

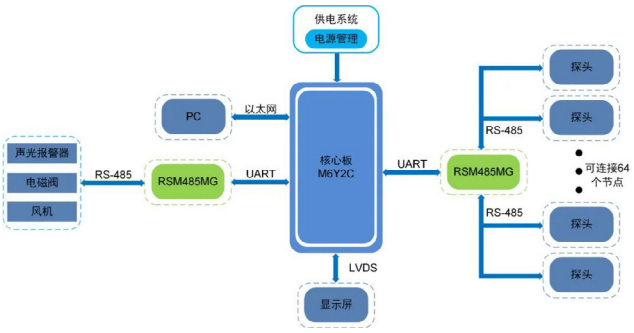


图4 气体检测应用的组网方案

【产品应用】CAN通信中的“过滤”是怎样实现的?

ZLG 致远电子 2023-12-26 11:41:01

一个 CAN 节点接入总线后，滤波不使能的情况下，总线上所有 CAN 数据帧都能被接收。若仅需接收一个固定范围 ID 的数据帧，可以通过配置屏蔽码和验收码，从而对数据帧进行过滤处理。

数据帧在 CAN 总线上传输时，CAN 节点通过验收滤波器来对比总线上的数据帧的 ID 与本节点的 ID 是否一致，若 ID 一致，CAN 控制器才允许将已接收信息存入对应的寄存器中；ID 不一致，该数据帧会被丢弃以此减轻 CAN 控制器的工作量。验收滤波器是由验收代码寄存器（ACR）和验收屏蔽寄存器 （AMR）共同组成。用户通过配置验收屏蔽寄存器和验收代码寄存器，进而得出对应的屏蔽码和验收码对 CAN 数据帧进行过滤处理。本文以 SJA1000 为例进行说明，其滤波方式分为单滤波和双滤波两种。由模式寄存器 AFM 位控制，AFM=1 时为单滤波；AFM=0 时为双滤波。

双滤波

双滤波模式下，定义了两个验收滤波器，接收的数据帧只要通过其中一个验收滤波器，就被认定为有效帧，即能被正常接收并存入寄存器中。ACR0、ACR1 和 AMR0、AMR1 组成第一个验收滤波器，ACR2、ACR3 和 AMR2、AMR3 组成第二个验收滤波器。以接收 CAN 拓展帧为例，该数据帧的 ID 段的前两个字节，即 29 位 ID 中的高 16 位参与滤波。ACR0 和 ACR1 控制 ID 位为 No.13-No.28，ACR2 和 ACR3 也控制 ID 位为 No.13-No.28，如表 1 所示。这意味着该 CAN 拓展帧 ID 位 No.13-No.28 这 16 位要与这两组验收代码中的其中一组一致才能被该节点正常接收并存入接收缓存区；否则该数据帧会被丢弃，这样达到了两次滤波的效果。

表1 AMR、ACR和控制ID位对应表

验收滤波器	验收屏蔽寄器（AMR）	验收代码寄存器（ACR）	控制 ID 位范围（位）
1#	AMR0	ACR0	ID.21 至 ID.28
	AMR1	ACR1	ID.13 至 ID.20
2#	AMR2	ACR2	ID.21 至 ID.28
	AMR3	ACR3	ID.13 至 ID.20

单滤波

单滤波模式下，所有 AMR 为“0”的位所对应的 ACR 位和参与单滤波数据对应的位需要一致才能被认定为验收滤波通过，如表 2 所示。接收 CAN 标准帧时，11 位 ID、RTR 位和 DATA 段前 16 位都参与滤波；接收 CAN 拓展帧时，29 位 ID 和 RTR 位参与滤波，如图 1 所示。

表2 AMR、ACR和滤波ID对应表

验收屏蔽码（AMR）	验收代码（ACR）	验收滤波通过 ID
0xFF	0x00	0x00-0xFF
0x00	0xFF	0xFF
0x00	0x08	0x08
0x08	0x08	0x00、0x08

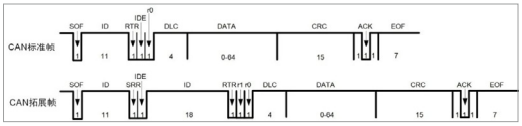


图1 CAN标准帧和CAN拓展帧的帧格式示意图

CSM330A

我司的隔离 SPI/UART 转 CAN 芯片 CSM330A 并未采用单滤波或者双滤波的方式进行接收滤波。CSM330A 的验收滤波器是由 1 个验收屏蔽寄存器和 6 个验收代码寄存器组成，用户可以通过配置一组屏蔽码“管理”6 组验收码。当屏蔽码某位值为 1 时，则该位对应的验收码会被“使能”，被“使能”的“验收码”和产品要接收的 CAN 帧的“帧 ID”相同，该 CAN 帧才会被接收到接收缓冲区。当“屏蔽码”的位值为 0 时，验收码不起作用，相应位的帧标识为任何值都可被接收，如表 3 所示。

表3 滤波、屏蔽码真值表

屏蔽位	验收位	帧 ID	接收或拒绝
0	X	X	接收
1	0	0	接收
1	0	1	拒绝
1	1	0	拒绝
1	1	1	接收

产品应用

CSM330A 适用于工业通讯、电池检测、充电桩、轨道交通、楼宇自动化等领域。CSM330A 协议转换芯片应用在智能汽车司机行为监测的应用方案，如图 2 所示。ZLG 面向 DMS 行业客户主要提供主控、供电、通讯防护等的模块化方案。主控方面为 Cortex-A35 M1808 AI 核心板，搭载自研图像传感技术搭配出车载行为检测系统解决方案。可实现驾驶员身份识别，对司机异常操作发出警报和提醒等功能。

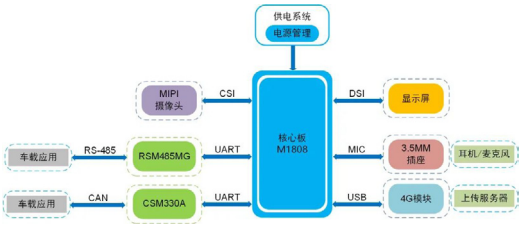


图2 司机行为检测应用方案

【技术分享】无源滤波设计分享，揭开测温滤波电路的神秘面纱

ZLG 致远电子 2023-12-11 11:34:07

在使用热电偶测温模块或设计热电偶测温电路中，每个采集通道往往需要一阶甚至多阶的无源滤波电路，关于热电偶测温无源滤波电路的设计需要考虑什么因素呢?

热电偶测温原理简述

两种不同材料的导体 A、B 与采样电路串接成一个闭合回路，当热端 1 和冷端 2 处于不同的温度 T、T0 时，回路中就会产生热电动势 EAB（T，T0）被 ADC 采集到。当热端和冷端的温度差发生变化时热电动势 EAB（T，T0）随之变化，用户可根据采集到的电压值查找热电偶温度对照表得出热端 T 与冷端 T0 的温度差值，进而得到热端温度值。

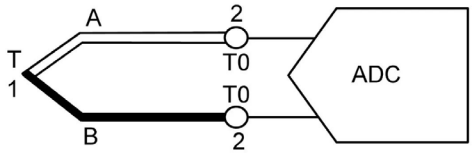


图1 热电偶测温电路简图

输入信号调理

在热电偶测温电路设计中，信号的调理是非常关键的。由于混叠效应，一般 ADC 采集电路前端都需要某些频段的滤波来减少输入噪声，以使模块具备更高的测温精度。

作为一个示例，考虑设计一个截止频率小于工频 50Hz 的低通滤波器电路，另外考虑到采样电压的建立时间，串联电阻值与滤波电容值不能太大，一般设定串联电阻值不高于 500Ω。根据一般经验差模电容值比共模电容值大 10 倍，设计出如图 2 所示一阶低通滤波电路，其截止频率 f=1/[2π(R3+R4)(C3+C2/2)]。根据该电路图 3 的波特图仿真结果可知其截止频率约为 48.85Hz。

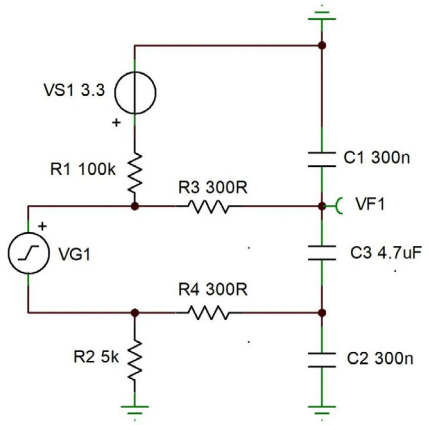


图2 一阶低通滤波电路图

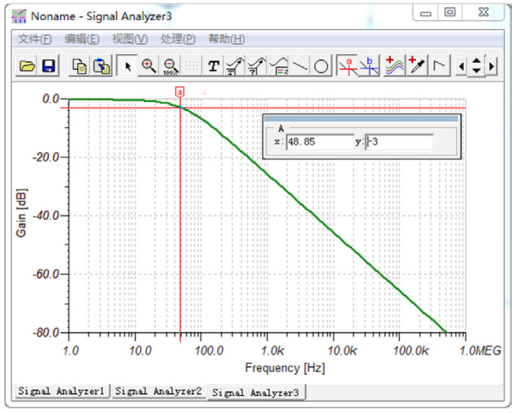


图3 一阶低通滤波电路波特图

根据工程设计经验，考虑到阻容在环境温度下的阻容值会发生变化及电容的压电效应等环境或器件因素影响，另外考虑到设计余量我们一般会将截止频率设计的尽可能低一些。例如可能将截止频率设计在 25Hz 左右，此时 C3 的容值可能接近 8uF。考虑到大容值电容的成本以及一阶滤波的滚降不够理想，因此我们采用二阶滤波的方式进行改善。

我们可以在图 2 的电路基础上再增加一阶电路改为无源二阶低通滤波器如图 4 所示，根据该电路图 5 中的波特图仿真结果，可知该电路截止频率的仿真值为 24.56Hz。

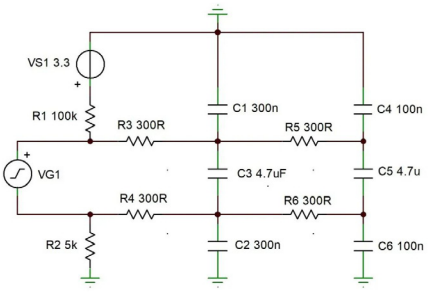


图4 二阶低通滤波电路图

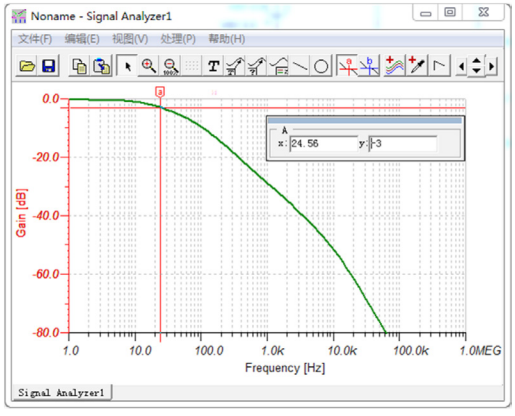


图5 二阶低通滤波电路波特图

总结一下

热电偶测温低通滤波电路设计的截止频率越低，对高频噪声的抑制能力越强，但是截止频率越低阻容值要求越大采样电压建立时间越长。截止频率设置过高，对工频或其它频段干扰的衰减不够理想会对测温精度产生影响。因此热电偶测温低通滤波电路截止频率设计的过高或过低对测试结果均会产生影响，根据经验值截止频率可设计在 30Hz~40Hz 左右。

热电偶测温模块 ZAM6218A 的推荐滤波电路采用二阶滤波截止频率约为 40Hz，该产品具备优异的采样精度和测温范围，通过对采集数据进行处理后 ZAM6218A 通过 IIC 协议直接输出热端温度数据，大大简化了用户的软件和硬件设计，是热电偶测温的不二选择。

图6 ZAM6218A简易说明图

【产品应用】雨量检测模块在智能家居中的应用

ZLG 致远电子 2023-12-18 11:44:40

随着科学技术的发展和生活节奏越来越快，以家居生活日常事务管理系统为主要特点的智能家居逐渐步入人们的生活。集成雨量传感器的智能家居系统能有效地对窗户进行智能化控制，提高整个系统的可靠性和用户体验。

智能家居的介绍及使用场景

智能家居又称智能住宅，是以住宅为平台，家居电器及家电设备为主要控制对象，利用综合布线技术、网络通信技术、安全防范技术、自动控制技术、音视频技术将家居生活有关的设施进行高效集成，构建高效的住宅设施与家庭日程事务的控制管理系统，提升家居智能、安全、便利、舒适，并实现环保节能的综合智能家居网络控制系统平台。衡量一个智能家居系统的成功与否，并非仅仅取决于智能化系统的多少、系统的先进性或集成度，而是取决于系统的设计和配置是否经济合理，并且系统能否成功运行，系统的使用、管理和维护是否方便，系统或产品的技术是否成熟适用，换句话说，就是如何以最少的投入、最简便的实现途径来换取最好的用户体验，实现便捷高质量的生活。

图1 智能家居应用场景概念图

智能家居是接收传感设备的各类传感信号，并触发控制命令或提醒人手动触发对应的智能设备来收回控制命令，例如：智能窗户系统主要通过雨量传感器、温湿度传感器检测温湿度、雨滴等周围环境，利用控制核心处理器驱动预置关窗程序，实现窗体的自动闭合。

方案设计

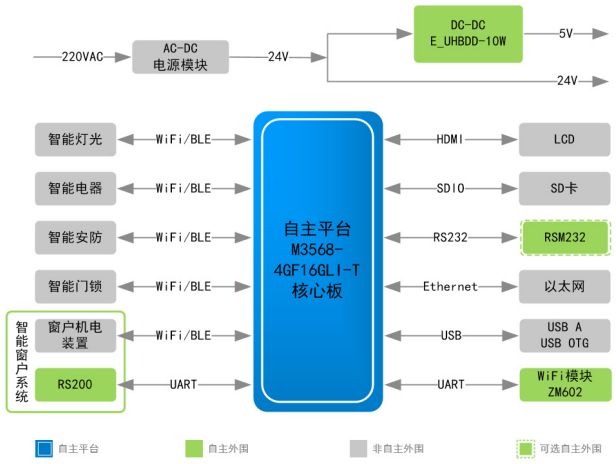


图2 智能家居应用方案框图

该方案使用了致远的 M3568-4GF16GLI-T 核心板、RS200 雨量检测模块、ZM602 WiFi 模块、DC-DC 隔离电源系列 E_UHBDD-10W 和 RS232 隔离接口模块 RSM232，智能家居应用方案框图见图 2。

方案使用模块及性能优点如下：

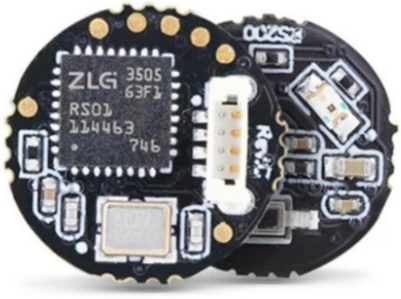
模块	性能优点	工作温度
M3568-4GF16GLI-T 核心板	核心板采用四核 64 位 Cortex®-A55 架构，高性能、低功耗，支持 Buildroot，Ubuntu，Debian 等操作系统。	-40°C ~+85°C
RS200	基于光学系统，能准确检测玻璃表面的雨滴状况，将雨滴状况分为无雨、小雨、中雨、大雨反馈给主机。	-40°C ~+85°C
ZM602	ZM602 支持 IEEE 802.11b/g/n 协议，支持 UART 串口透明传输，用户可实现设备的快速联网。	-30°C ~+85°C
E_UHBDD-10W	具备优异的 EMS 性能的电源隔离模块，高输入范围，自带输出过压、过流保护。	-40°C ~+85°C

点击购买

M3568-4GF16GLI-T 核心板可作为智能家居应用方案的主控板，板载低功耗应用处理器，可搭载 Buildroot，Ubuntu，Debian 等操作系统。核心板上搭配外围的 Wi-Fi、以太网可以实时远程收发数据，对家用电器、设备进行监控、远程控制。

M3568-4GF16GLI-T产品特性：

- 内核 RK3568J；
- 主频 2GHz；
- 内存 4GB DDR4；
- 低功耗、高性能；
- 工作环境温度 -40°C ~+85°C。



点击购买

RS200 是致远电子推出的智能雨量检测模块。可以快速、实时检测玻璃表面雨量变化的情况，且不受阳光、粉尘、树叶等干扰。可以直接为用户反馈无雨、小雨、中雨、大雨等数字状态，给予用户轻松便捷的开发体验。

RS200产品特性：

- 基于光学系统，能准确检测玻璃表面的雨量状态；
- 使用 HALIOS®-SD 测量方法，超强抗太阳光干扰能力；
- 模块安装在玻璃罩内部，不与外界自然环境接触，减少环境干扰，增长使用寿命；
- 超小体积，直径仅为 12.5mm，轻松适配各种摄像头；
- 使用带 CRC-8 校验的通信协议，提高通信抗干扰能力；
- 光学系统自校准；
- 自然环境光检测，为摄像头提供更多辅助参数；
- 板载温度测量，优异的温度线性关系保证测量的准确度；
- 支持光学睡眠模式，延长光学器件使用寿命；
- UART 通信接口；
- 3.3V 供电电压；
- 模块错误自检测输出；
- 工作环境温度 -40°C ~+85°C。

点击购买

ZM602 系列无线模组是一款高性能、高可靠的 Wi-Fi + BLE 二合一无线模组，支持 802.11 b/g/n Wi-Fi 协议以及蓝牙 5.0 通信协议，内部集成完整的射频收发电路，采用双 UART 通信接口，内置 PCB 天线以及射频连接器，具有高集成度、应用灵活等特点，广泛应用于智能家居、智慧城市、医疗电子 、智慧楼宇等领域。

ZM602产品特性：

- Wi-Fi 协议：IEEE 802.11 b/g/n；
- 蓝牙协议：BLE 5.0；
- 2.4GHz 频带 1T1R 模式，20Mhz 带宽，数据速率高达 72.2Mbps；
- WiFi 工作模式：Station 模式、SoftAP 模式、SoftAP+Station 模式；
- 通信接口：UART×2 或 SDIO；
- 串口波特率：1200~10Mbps；
- 支持 TCP/UDP /MQTT/HTTP 协议；
- 支持最多 4 路数据传输通道；
- 支持 UDP 组播功能；
- 工作环境温度 -30°C ~+85°C。

点击购买

E_UHBDD-10W 隔离电源模块，用于主控板的隔离电源供电，可以提供优异的 EMS 性能。

E_UHBDD-10W产品特性：

- 隔离电压 1500VDC；
- 输入电压范围 9V~36V；
- 效率高达 87%；
- 可持续短路，自恢复；
- 输出过压保护、输出过流保护；
- 工作环境温度 -40°C ~+85°C。

2023/12 第12期

微文摘

ZLG MICRO DIGEST

ZLG致远电子官方微信

广州致远电子股份有限公司
更多详情请访问 www.zlg.cn
欢迎拨打全国服务热线 400-888-4005