

## 应用手册

### 使用 Advantech ADAM 67xx IoT gateway 通过 MXE API 和 OMNEO Dante OCA 网络接口远程控制 MXE 矩阵

MXE 矩阵配置有 1 个 OMNEO DANTE OCA 网络接口用于连接其他系统，使用 CAT 网线和 Ethernet 网络交换机。

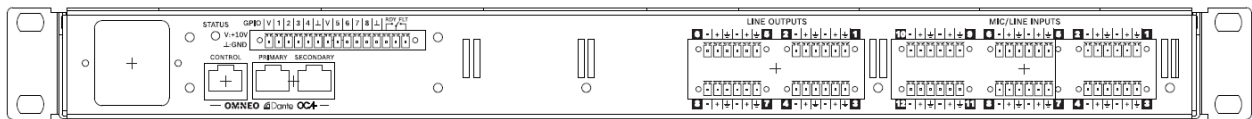


图 1: MXE 背面板

在 MXE 背面板上可以找到网络接口 (OMNEO DANTE OCA)，总共包括 3 个网络端口：CONTROL, PRIMARY 和 SECONDARY。

3 个网络端口可以通过 SONICUE 进行配置运行于 Transparent, RSTP 或 Glitch-Free mode。

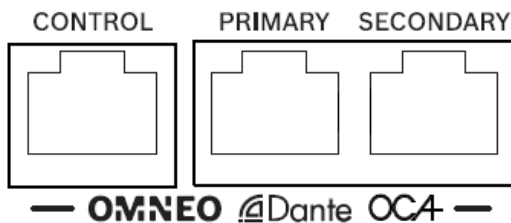


图 2: MXE 网络接口

## 需求

MXE 矩阵，固件版本 1.0.2561 (或更高)

MXE 前面板菜单中激活 MXE 应用编程接口 (API)

电脑安装 SONICUE 1.2 (或更高版本)

## 文档 (除本手册外的其他推荐)

关于 Advantech ADAM 6700 系列 IoT gateway 所有硬件连接配件的详细说明可以参考其设备操作手册。

## 1. Advantech ADAM 6700 Series IoT gateways (第三方产品)

### 应用

Advantech' s ADAM 6700 系列 IoT gateway 主要设计作为网关用于连接“老”设备，从而配备到基于 IoTa 和云服务的现代化系统中。

与 MXE 结合会产生一些有趣的应用：

- GPIO 扩展 -> GPIO (通用控制输入/输出接口) 的数量
- GPIO 扩展 -> 远程 GPIO
- Email 邮件通知 (免费)
- SMS 短信通知 (基于服务商收费)
- 接入第三方系统 (需要更高级的编程技巧)

### 编程

可以通过 IBM 的 Node-RED 进行编程。Node-RED 是一个可视化的基于 flow 的编程工具，它被预装在 ADAM-6700 系列 IoT gateways 中。在 YouTube 上可以找到很多关于如何使用 Node-RED 进行编程的教学视频。

### 安装

所有型号都具有相同的尺寸，可以安装于 DIN 轨道上。

### 区别

下列 ADAM 67xx IoT gateway 可选，唯一不同的是连接，6750 与 MXE 的结合已经成功测试：

- ADAM-6717 IoT gateway
- ADAM-6750 IoT gateway
- ADAM-6760D IoT gateway



图 3: Advantech ADAM-6717 (左) , ADAM-6750 (中) 和 ADAM-6760D (右)

## Ethernet 端口

每个型号除了模拟和数字输入/输出不同外，所有型号都配置 2 个 Ethernet 端口，具有相同的功能。

2 个 Ethernet 端口 (ENET 0 和 ENET 1) 位于 ADAM 6700 系列设备的顶部:

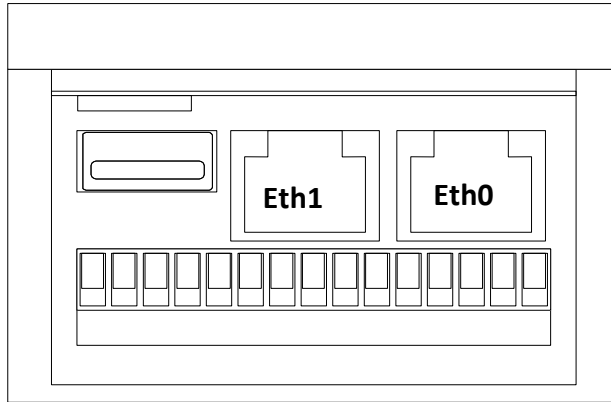


图 4: Advantech ADAM-6700 系列网络接口

## 基本设置 - 网络连接

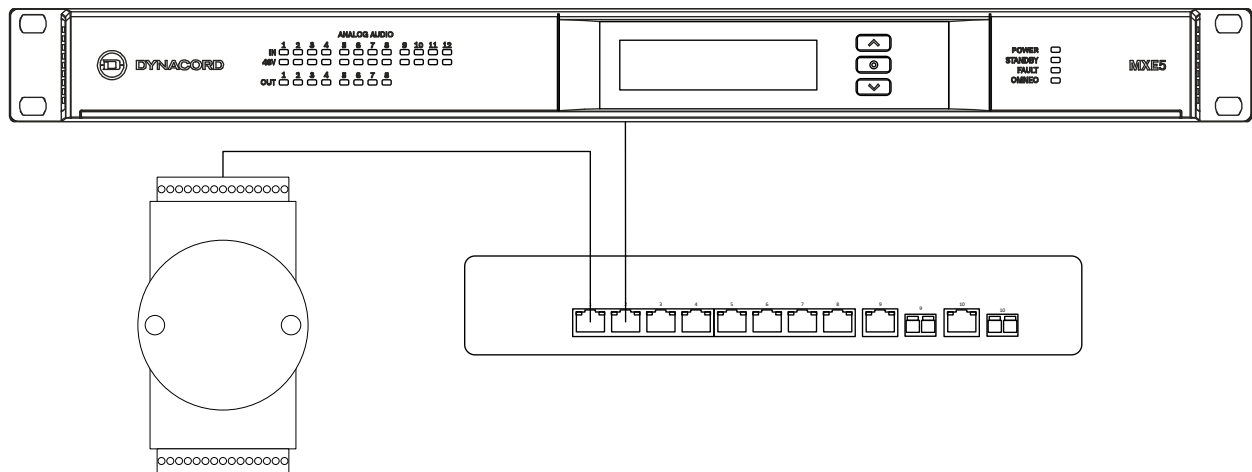


图 5: Advantech ADAM-6700 系列 IoT gateway 通过 Ethernet 网络交换机连接到 Dynacord MXE 矩阵

## 1.1. Advantech ADAM-6717 IoT gateway

产品图片:

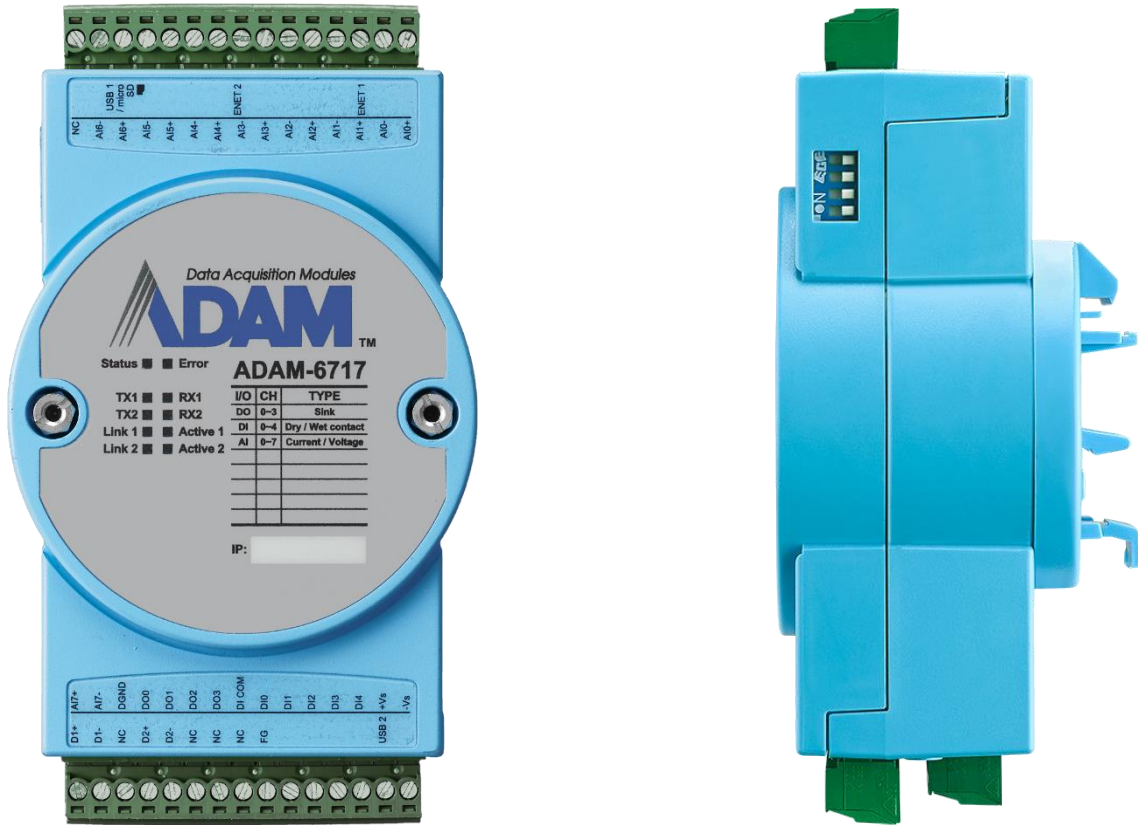


图 6: Advantech ADAM-6717 IoT gateway 前视图 (左) 和侧视图 (右)

### 基本描述

ADAM-6717 IoT gateway 是 Advantech ADAM 6700 系列 IoT gateway 的一部分。

具有如下输入/输出:

- 8 x 模拟输入
- 5 x 数字输入
- 4 x 数字输出

### 应用示例

- 用 1 个滑动变阻器 (一般 10K 欧姆) 连接到 ADAM 模拟输入端控制 MXE DSP 电平值
- 用 1 个开关或继电器连接到 ADAM 数字输入控制 MXE DSP 静音或设置 1 个故障标记
- 用 1 个 LED 连接到 ADAM 数字输出来显示 MXE 的故障信息
- 用 1 个小继电器连接到 ADAM 数字输出, 用于作为干接点开关控制更大电流或电压。

## 1.2. Advantech ADAM-6750 IoT gateway

产品图片:

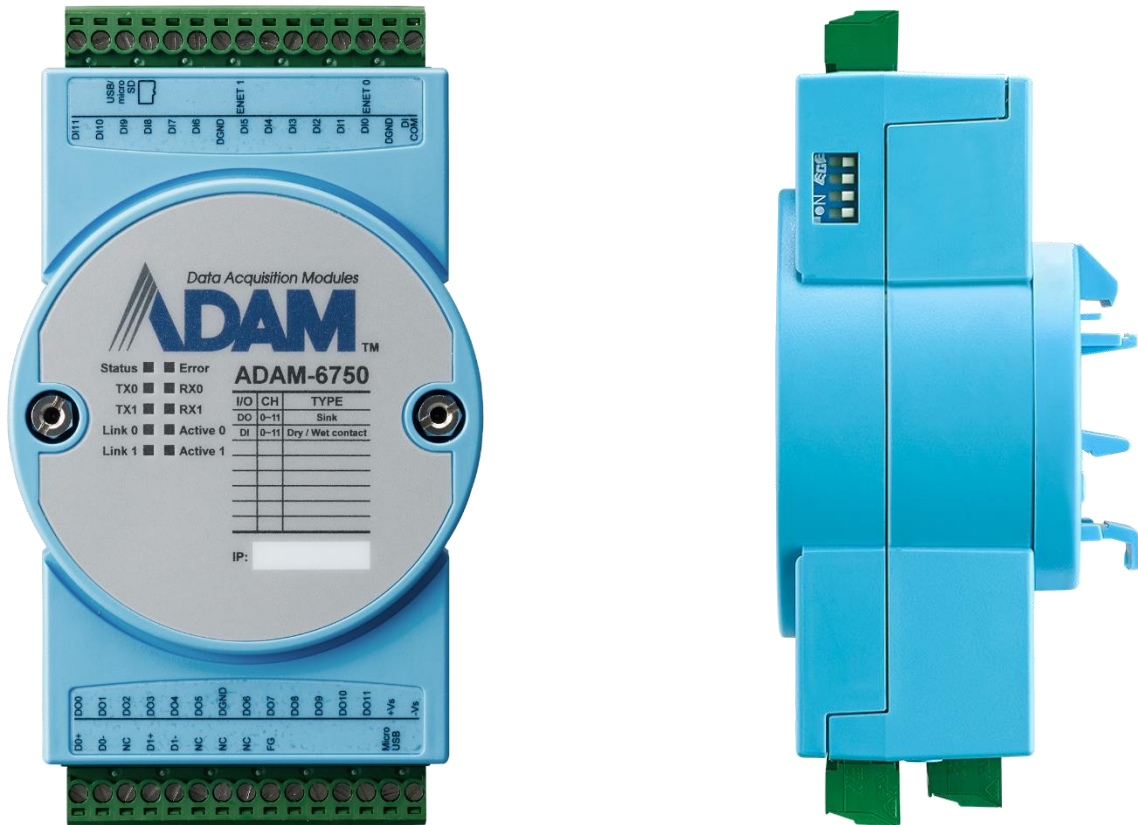


图 7: Advantech ADAM-6750 IoT gateway 前视图 (左) 及侧视图 (右)

### 基本描述

The ADAM-6750 IoT gateway 是 Advantech ADAM 6700 系列 IoT gateway 的一部分。

具有如下输入/输出:

- 12 x 数字输入
- 12 x 数字输出

### 应用示例

- 用 1 个开关或继电器连接到 ADAM 数字输入控制 MXE DSP 静音或设置 1 个故障标记
- 用 1 个 LED 连接到 ADAM 数字输出来显示 MXE 的故障信息
- 用 1 个小继电器连接到 ADAM 数字输出, 用于作为 MXE 逻辑触发的干接点开关控制更大电流或电压。

### 1.3. Advantech ADAM-6760D IoT gateway

产品图片:

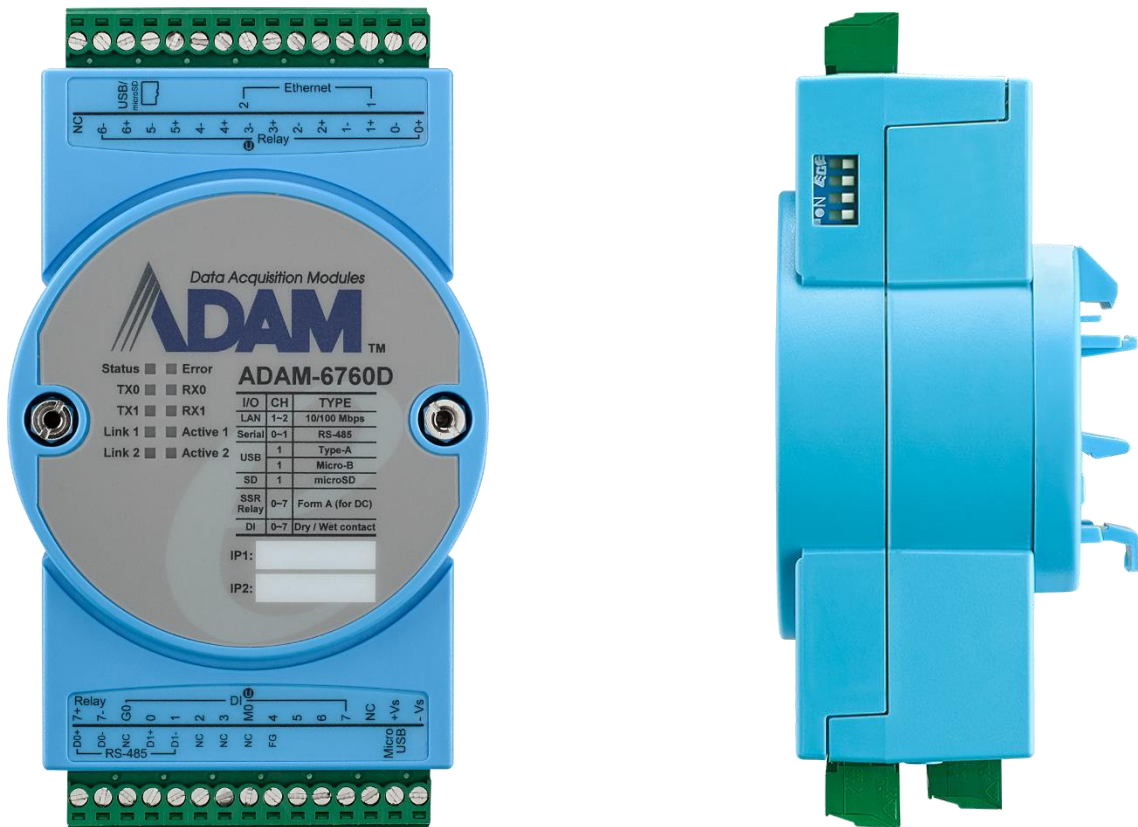


图 8: Advantech ADAM-6760 IoT gateway 前视图 (左) 和侧视图 (右)

#### 基本描述

The ADAM-6760 IoT gateway 是 Advantech ADAM 6700 系列 IoT gateway 的一部分。

具有如下输入/输出:

- 8 x 数字输入
- 8 x 继电器输出 (PhotoMOS SPST)

#### 应用示例

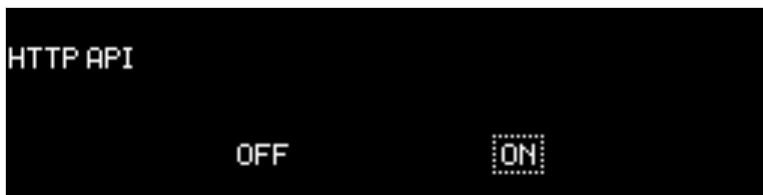
- 用 1 个开关或继电器连接到 ADAM 数字输入控制 MXE DSP 静音或设置 1 个故障标记
- 用 1 对阻抗连接到 ADAM 继电器输出作为由 MXE 逻辑触发的用于其他系统的监测、干接点或者故障信号指示。

## 2. MXE HTTP API

MXE HTTP API 是连接 OCA 控制系统既简便又易用的集成接口。

### 2.1. 配置

HTTP API 默认为禁用，如需激活，在 MXE 前面板主菜单中的 API 界面开启。



### 2.2. HTTP 服务器

MXE 运行一个本地的 HTTP 服务器，用于 API，http 端口：8008、https 端口：443

HTTP URI 资源根目录：

<http://<hostname-or-ip>:8008/api/v1/>

HTTPS URI 资源根目录：

<https://<hostname-or-ip>:443/api/v1/>

主机名称可以通过设备型号和 MAC 地址获取：MXE -< MAC 地址后 3 个字节>

示例：MXE5 的 MAC 地址 00:1C:44:F5:60:59

则主机名称为：mxe-f56059



## 2.3. HTTP 会话

所有格式正确的请求都返回 JSON-encoded 类型为 application/json MIME 的负载内容。

每个资源都接受 JSON-encoded 参数，该参数应以 POST 负载提供。

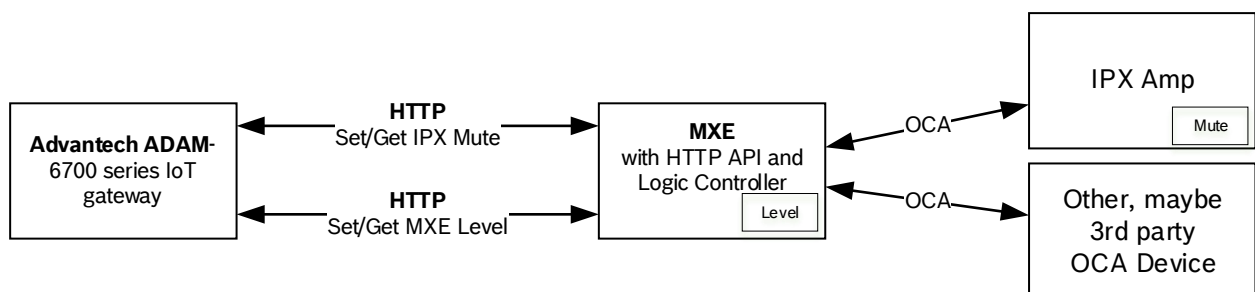
错误响应包含 1 个 4xx HTTP 响应代码和 1 个 text/plain 响应正文。

## 2.4. 资源

资源	参数	示例
/virtual_logic	由 100 个布尔值组成的 JSON 数组，指示相应索引上虚拟逻辑值的状态	[true,false,false,...]
/virtual_logic/<0..99>	指示虚拟逻辑值状态的布尔值	true
/virtual_analog	由 100 个浮点值组成的 JSON 数组，指示相应索引上虚拟模拟值的数值	[0, 1.2, -42.27, ...]
/virtual_analog/<0..99>	指示虚拟模拟值的浮点值	-80.0

## 2.5. 应用示例

如下图所示的系统中，可以通过 HTTP API 设置和获取功放通道的静音状态或者矩阵分区的总控电平。因此允许来自第三方设备或软件对于该功能的外部控制，本例中为 Advantech ADAM-6700 系列 IoT gateway。





### 2.5.1. 设置/ 获取功放通道静音

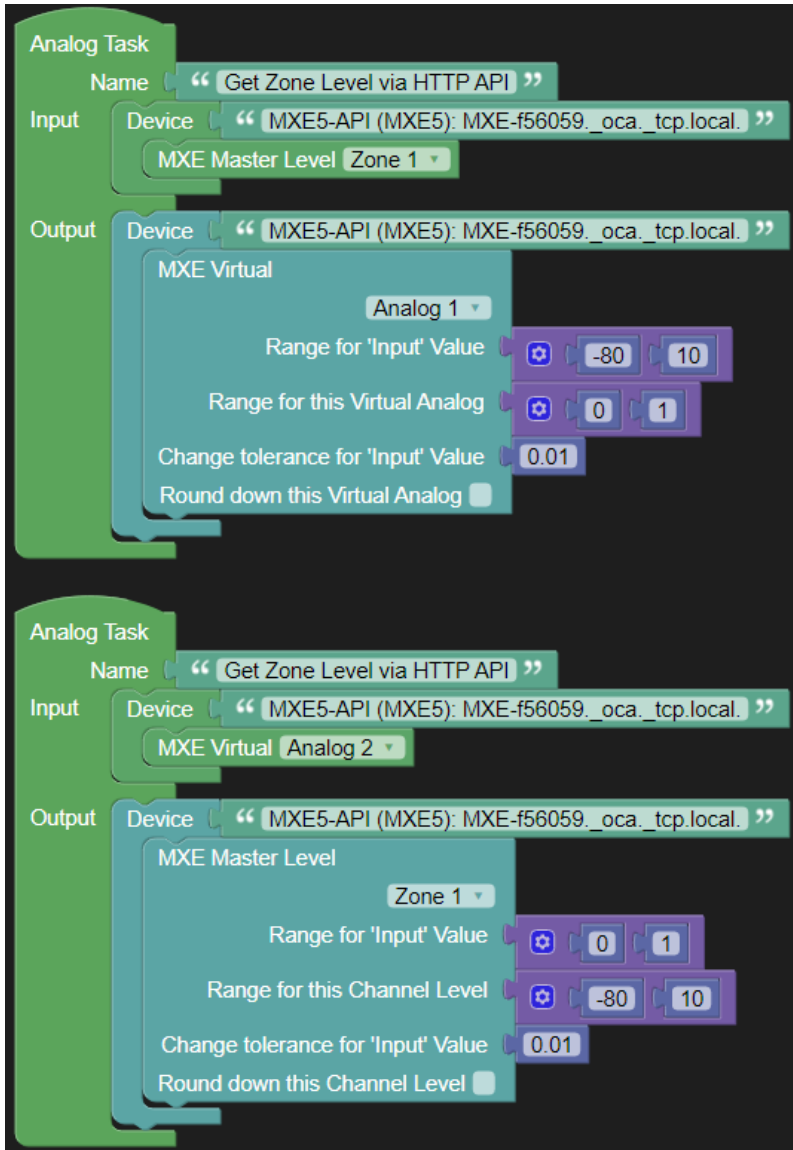
1. 在 SONICUE 中配置如下逻辑，用于开启 API 功能的 MXE。



2. 注意，虚拟逻辑模块在 API 中基数从 0 开始，而在逻辑 GUI 中基数从 1 开始。
3. 注意，如 2 个虚拟逻辑模块用于设置和获取静音，使用相同的虚拟逻辑模块将造成不希望的数据竞赛。

## 2.5.2. 设置 /获取分区总电平

1. 在 SONICUE 中配置如下逻辑，用于开启 API 功能的 MXE



2. 本例使用模拟输出的“范围...”选项实现与 API 数值 0...1 范围的对应，这可以帮助在实际参数允许范围内实现稳定的 API 均匀调整需求。比如：某个固定安装项目中不允许用户将电平调低于-40dB，则不需要更新网页编码，只需要将分区输出电平的“范围...”从-80....10 调整为-40....10。
3. 本例使用 0.01 的“Change tolerance”，防止数据反馈循环。

## 2.6. 安全性

当启用 OCA Control Security (OCA 控制安全) 后, 所有资源都需要基本的 HTTP 身份验证, 在该模式下, 将不再允许使用不安全的 HTTP 端口。如果 OCA Control Security 已经启用, API 网页服务器将接受与给定 PSK 身份和密钥相匹配的资质。如果你已经分享了一个包含 “HTTP-API” 身份的 PSK, 就可以用 “HTTP-API” 作为用户名和共享密码短语, 用作基本 HTTP 身份验证的凭据。

MXE 使用自签名证书用于 HTTPS 连接。

启用 OCA Control Security 不属于本 API 部分, 请参考 OCA 标准 [1]学习如果通过 OCA 和分享 PSK, 或者已有的 OCA 控制软件和设备启用 Control Security 。

[1] <https://www.ocaalliance.com/standards-specifications/>

### 3. Advantech ADAM/APAX 实用程序

Advantech ADAM/APAX 实用程序工具可以从 [Advantech website > Support](#) 进行下载。

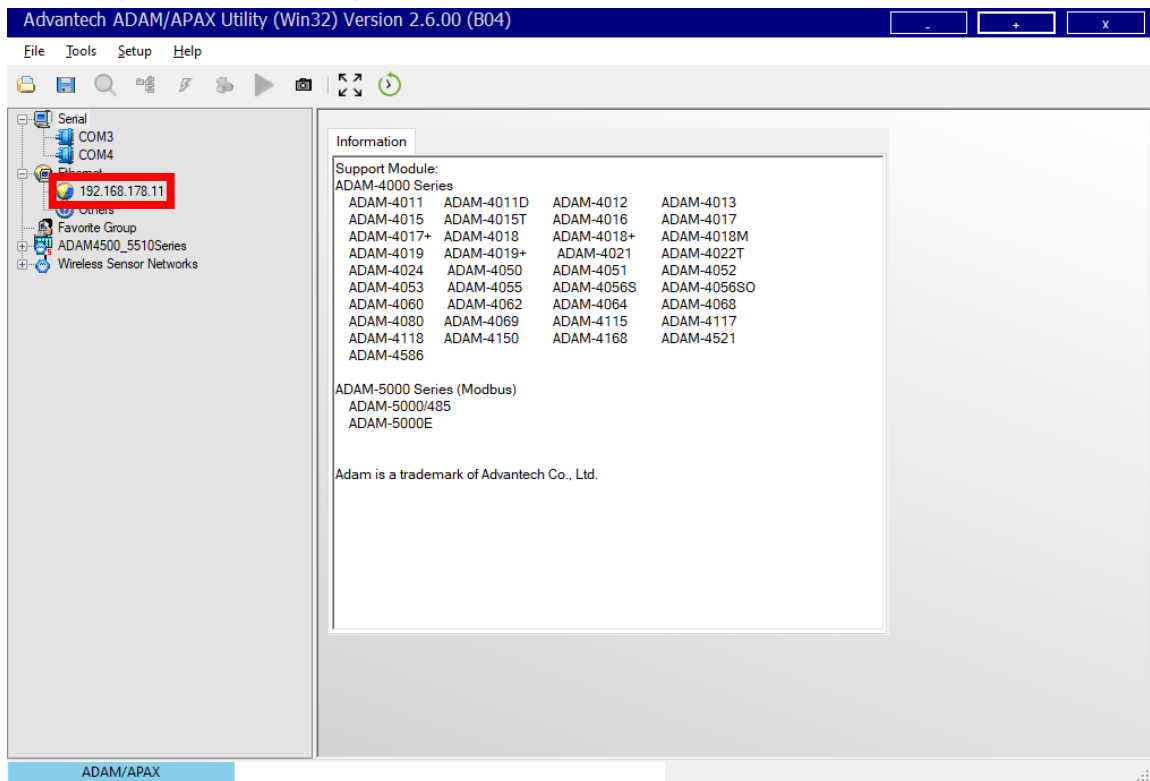
安装完成后将生成一个桌面快捷程序，你需要管理员权限来启动该应用。

如果没有 DHCP 服务器，ADAM 就保留默认地址：

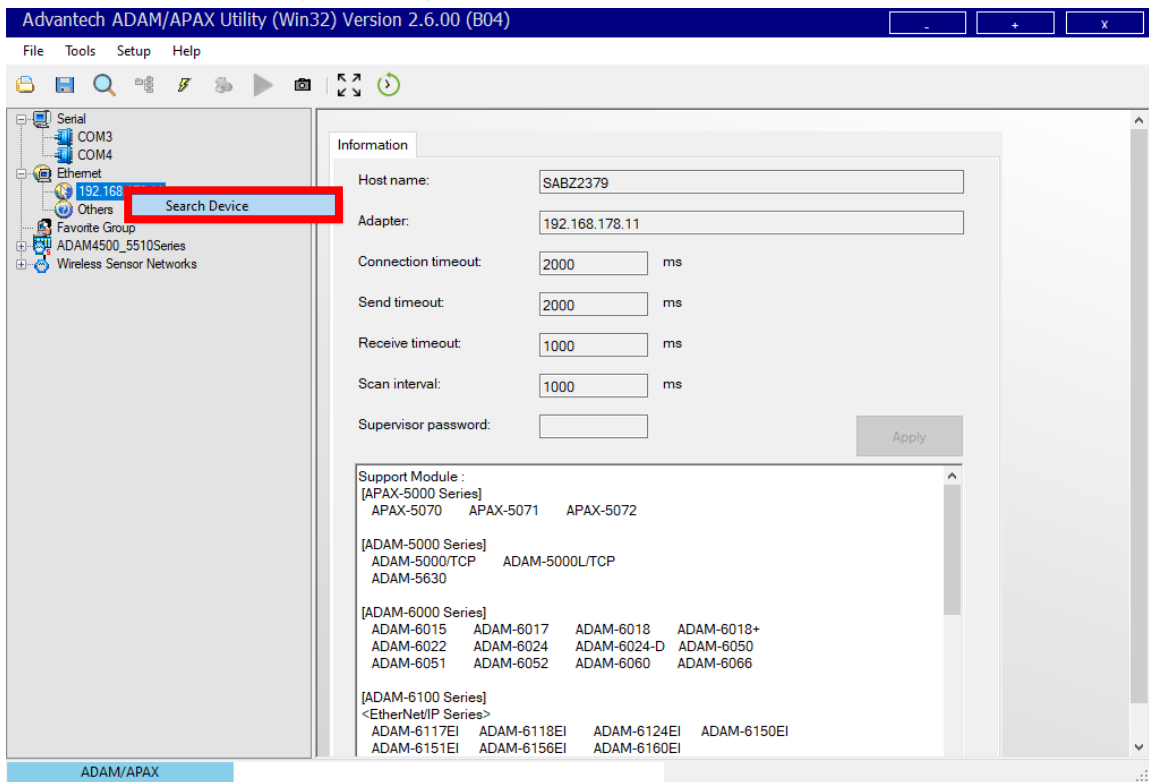
- 接口 Eth0: 10.0.0.1
- 接口 Eth1: 11.0.0.1

在后续的截屏中，ADAM 的 IP 地址配置为 192.168.178.101，而电脑的网络接口地址配置为 192.168.178.11。

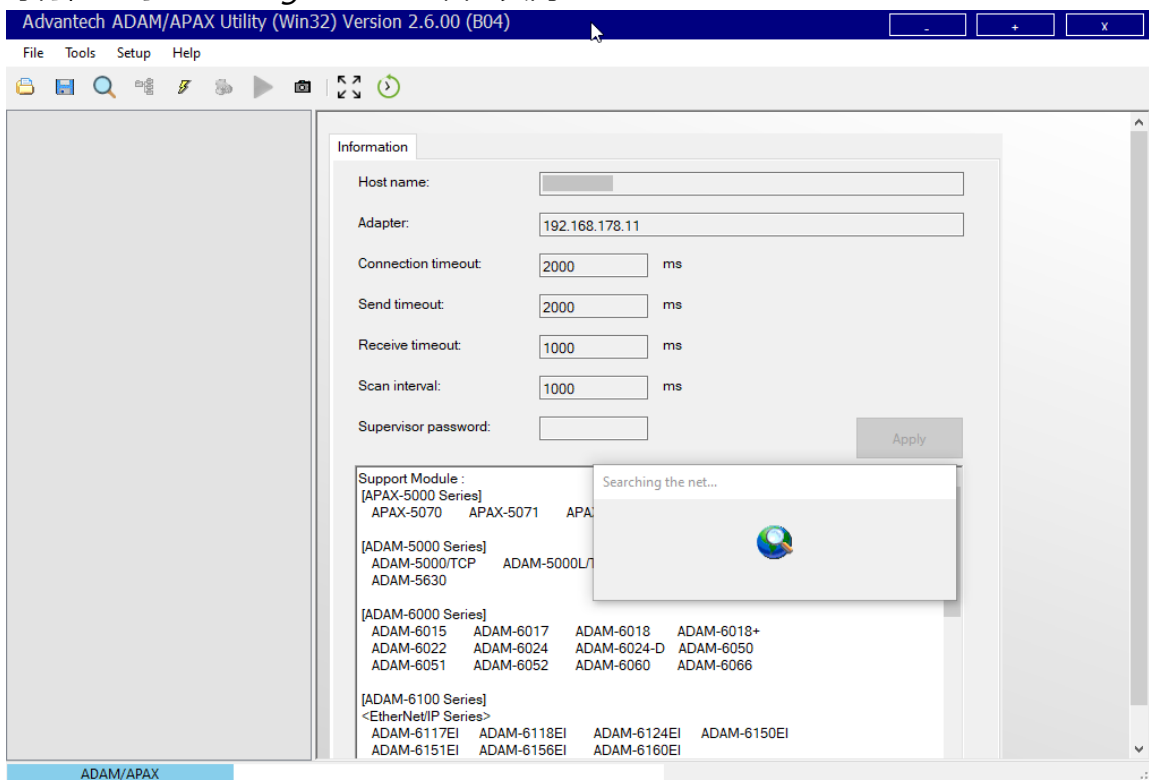
#### 3.1. 打开 Advantech ADAM/APAX 实用程序工具，在左侧 Ethernet 下面的子窗口中，右击 IP 地址 (=电脑的网络接口)



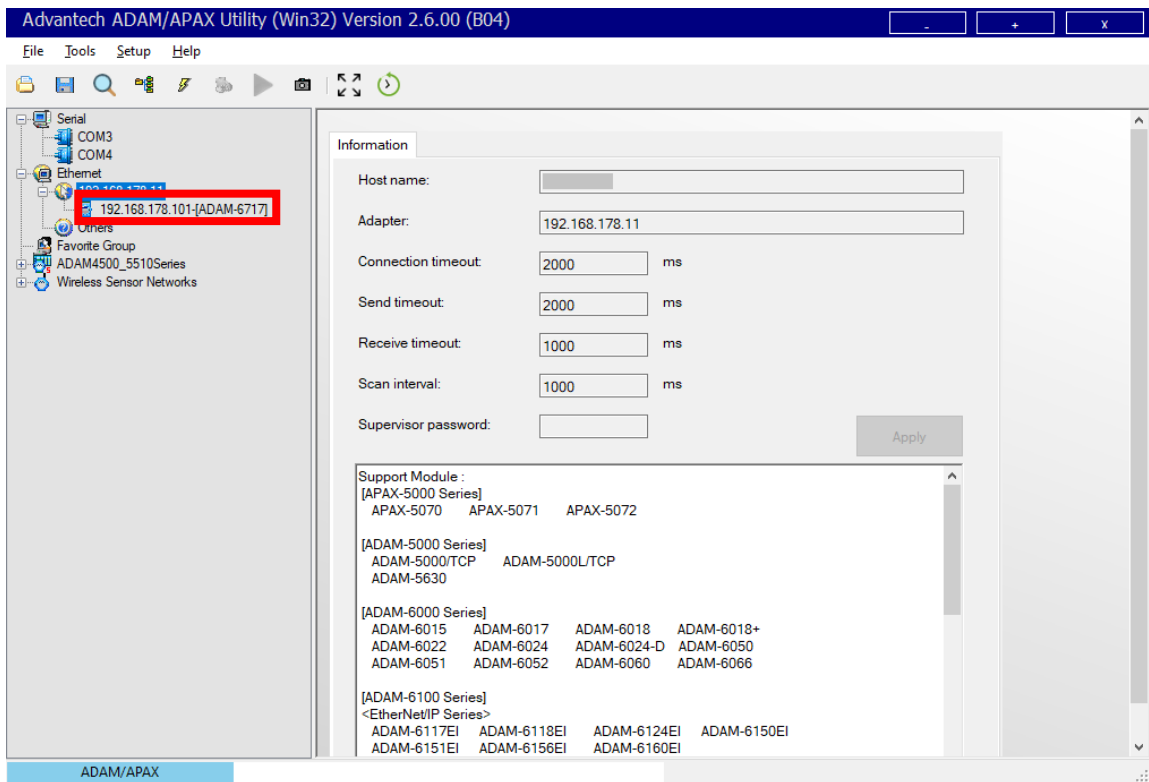
### 3.2. 选择 *Search Device* (寻找设备) 寻找 ADAM



### 3.3. 等待, 直到 *Searching the net...* 窗口关闭

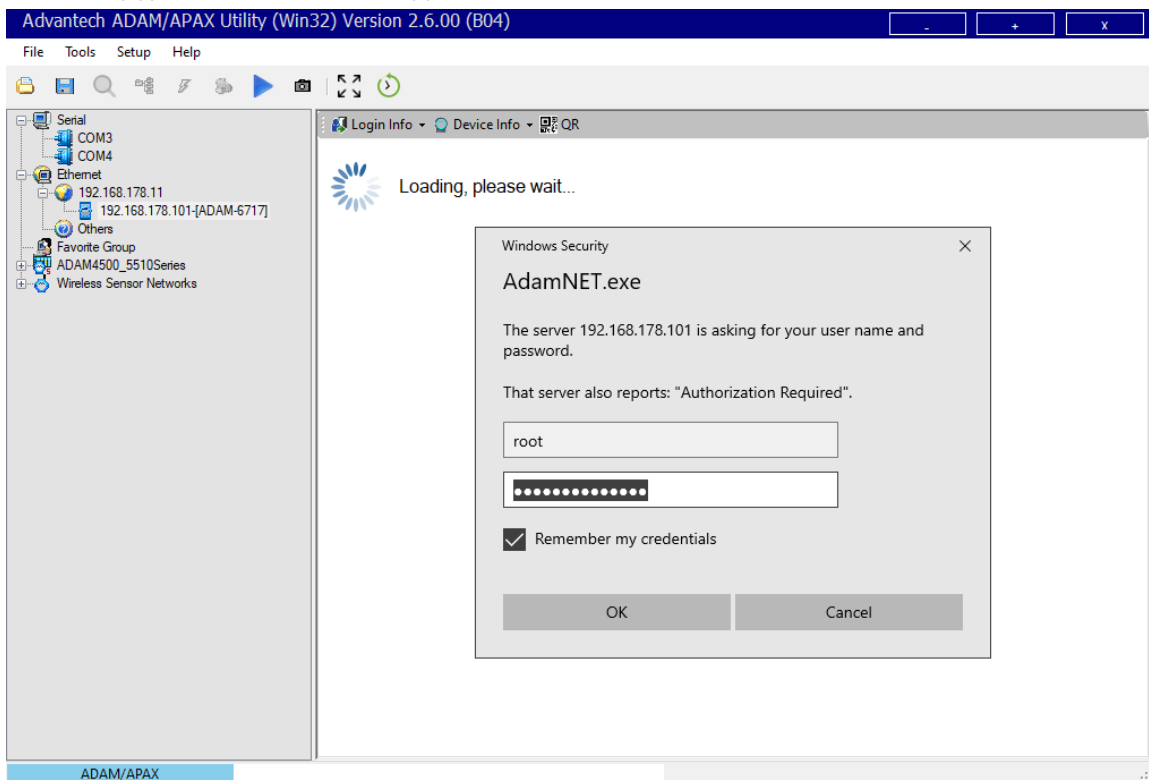


### 3.4. 双击找到的 ADAM

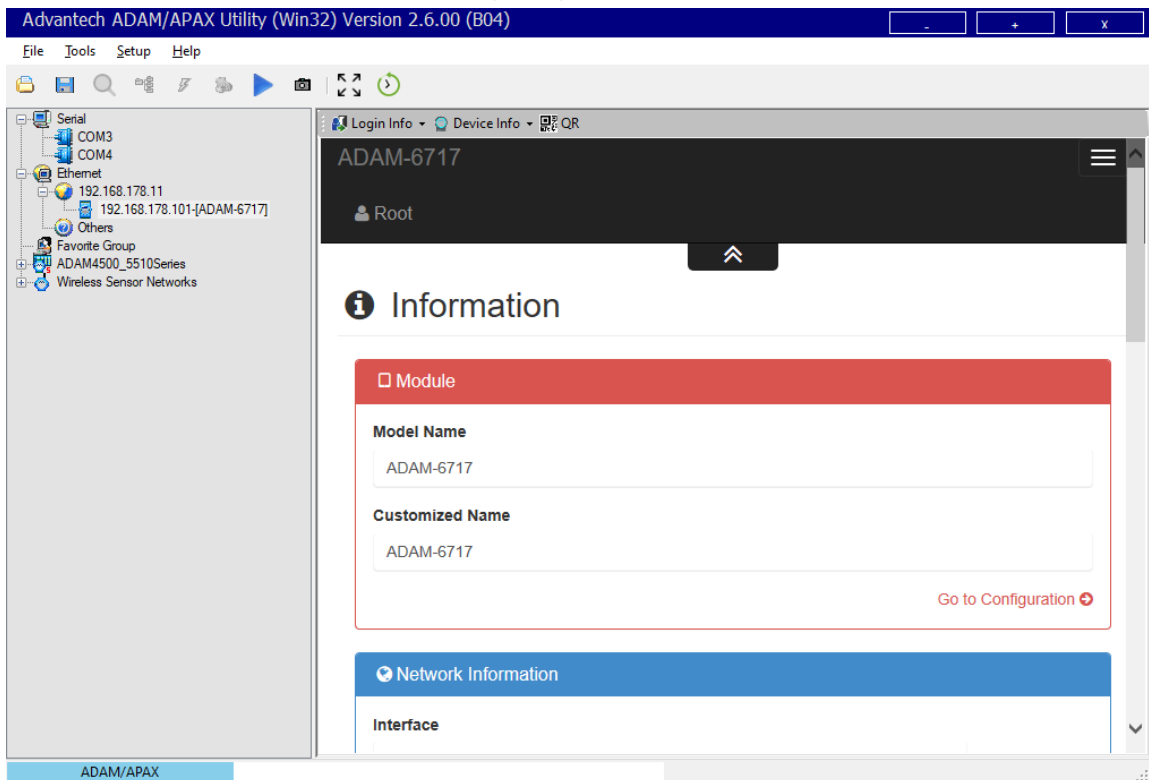


### 3.5. 使用用户名“root”和密码“00000000”（8个“0”）登录 ADAM

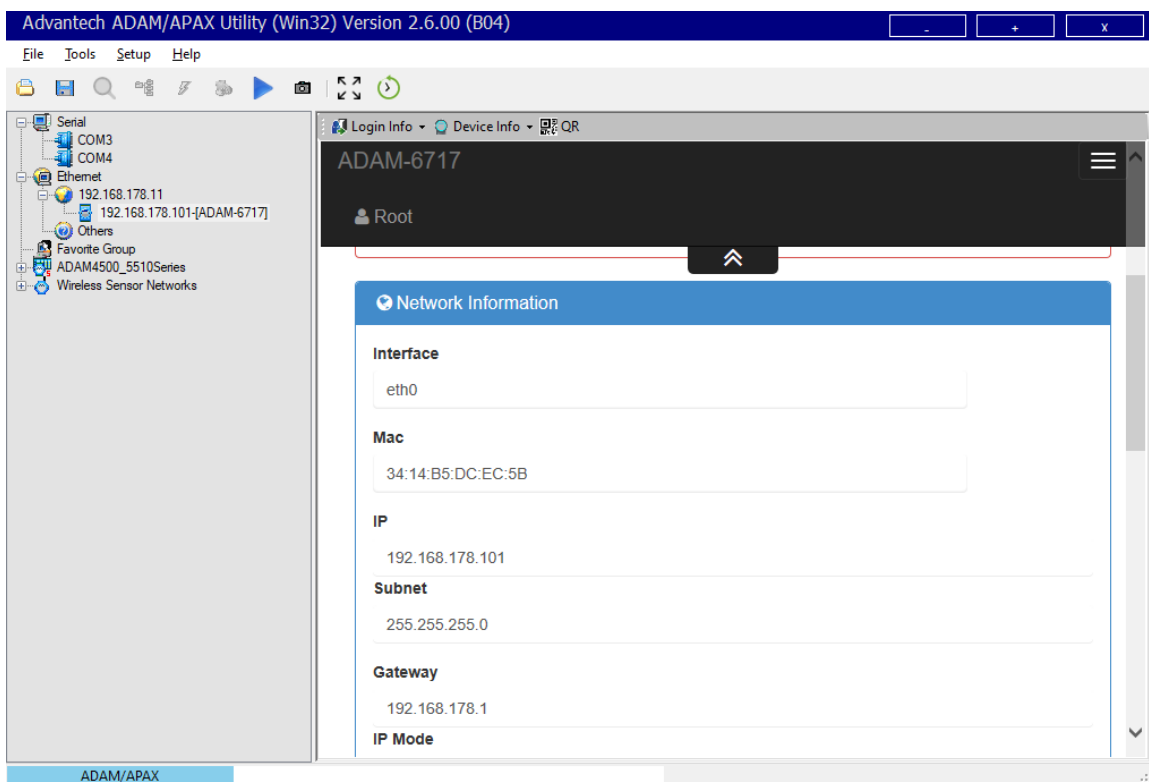
- 该用户名和密码需要变更，以保护系统，防止被侵入




### 3.6. 现在，你可以登入并检查 Module... (模块) 的信息

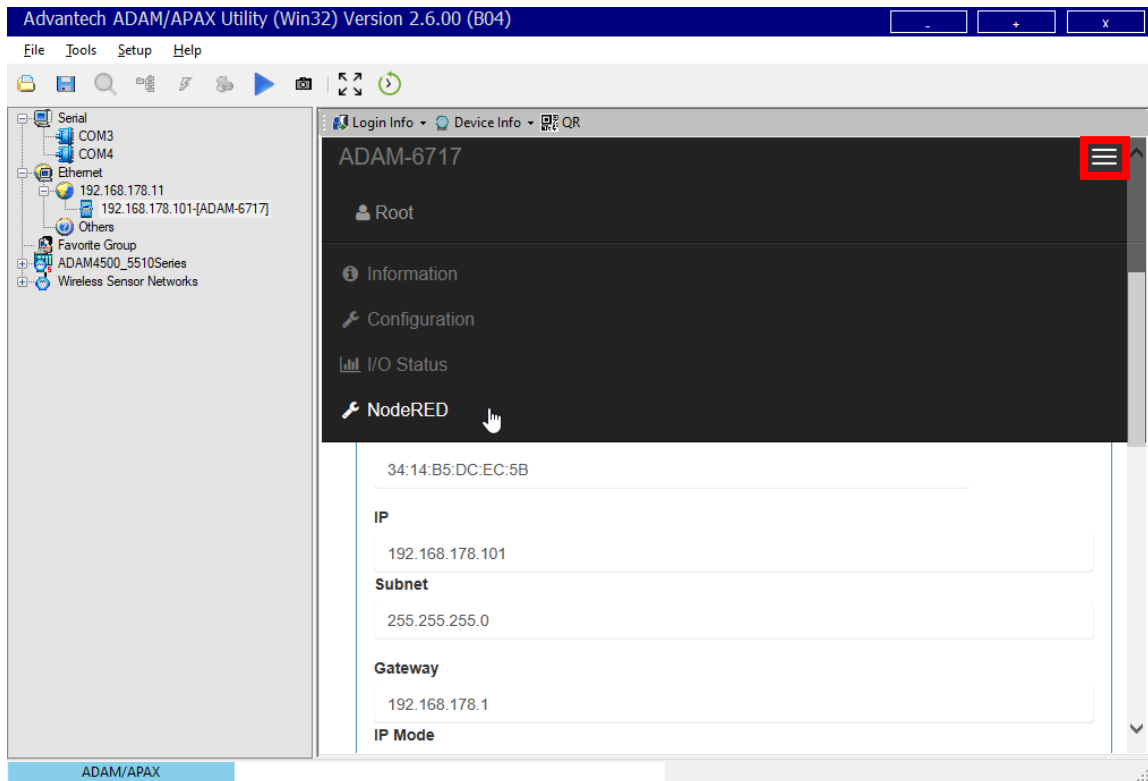


### 3.7. ... 然后，检查接口 eth0 和 eth1 的网络信息。

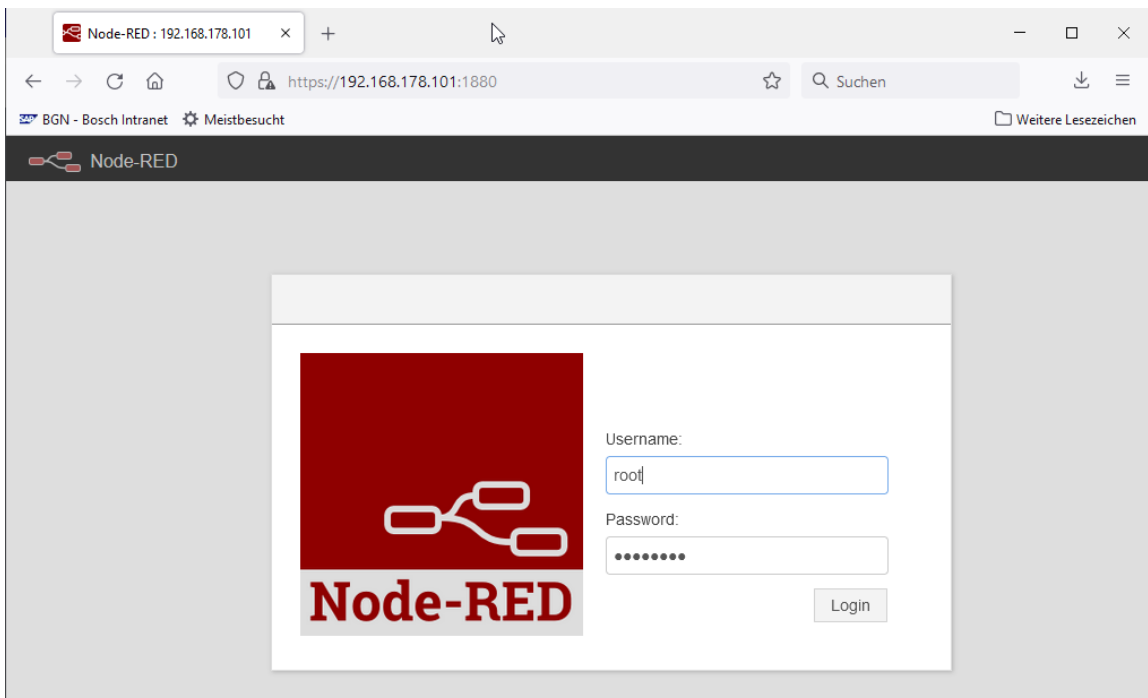




3.8. 点击右上角的 -按钮打开菜单，然后单击 *NodeRED*



3.9. 使用用户名“root”和密码“00000000”（8个“0”）登录 Node-RED  
- 该用户名和密码需要变更，以保护系统，防止被侵入



此时，ADAM 的 Node-RED 网页将被打开；下一步操作详见第 4 章：Node-RED

## 4. Node-RED

IBM 的 Node-RED 基于流 (flow) 的可视化编程工具已经预装在 Advantech ADAM-6700 系列 IoT gateways。

在 YouTube 上, 你可以找到很多关于如何使用 Node-RED 进行编程的教学视频。

### 4.1. 基本概念

Node-RED 中所谓的流 (flow) 是图形化的连接点。

通常在 Node-RED 中有一些是核心节点, 还有一些是作为运行 Node-RED 设备比如:

Advantech ADAM-6700 系列的 GPIO 接口的特别节点。

如果有些功能, 由核心节点和特别节点 (如 Advantech ADAM 节点) 都不足够提供, 可以安装额外的节点库或者基于 JavaScript 编写定制功能 (不包含在本应用手册)。

所有核心节点 (Inject, Debug, Function, Change, Switch and Template) 的描述可以详查:

[Node-RED website > documentation > The Core Nodes.](#)

其他节点, 比如 Email, 可以很容易从各种各样有关 Node-RED 的库或者论坛进行安装。

Advantech ADAM-6700 系列已经预装了 Email 节点以及其他本示例中应用到的节点。

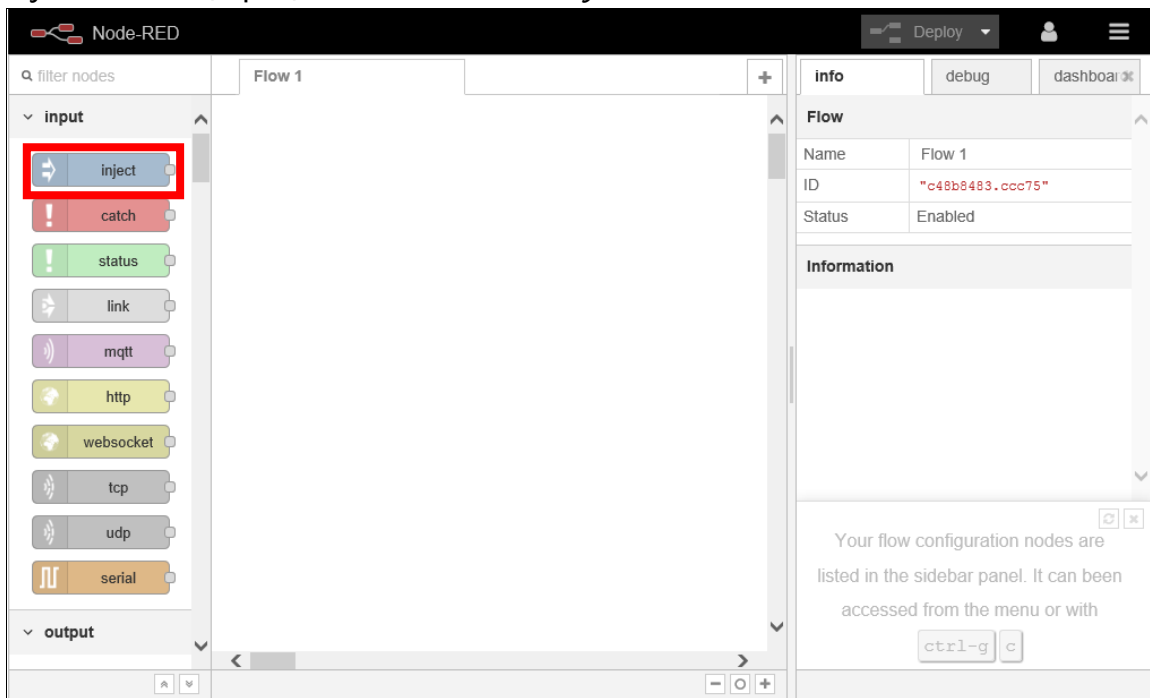
本例中应用到的节点:

- inject
- http request

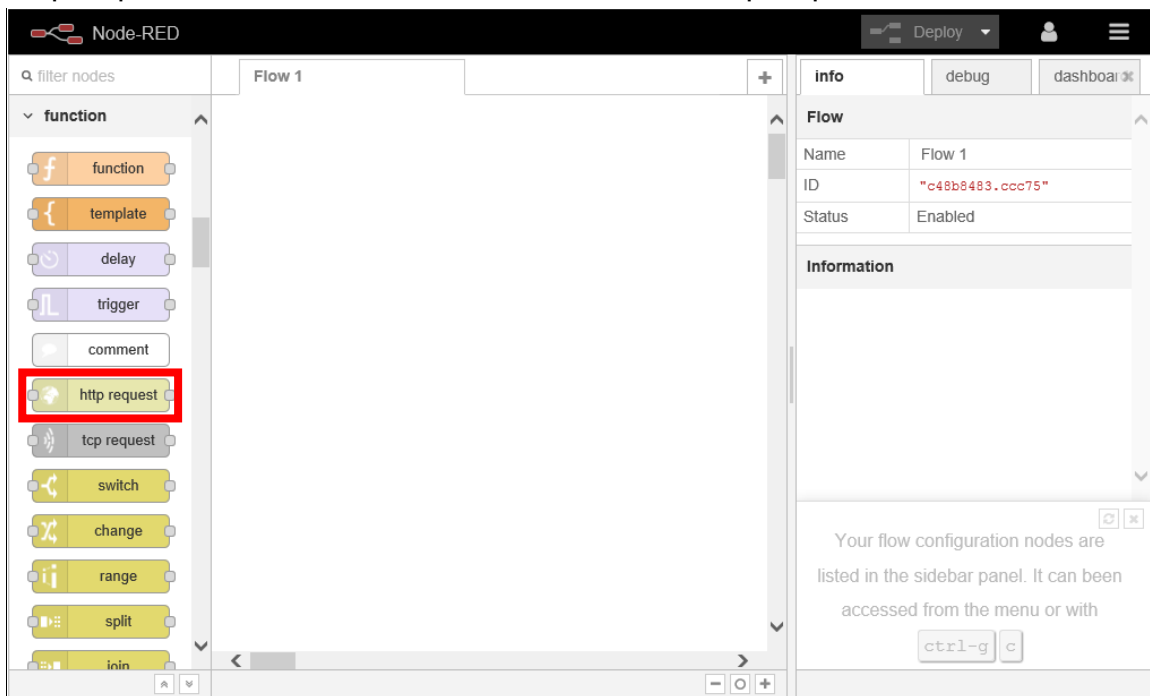
本例中应用到的 Advantech ADAM 节点

- get ai value
- get dio value
- set do value

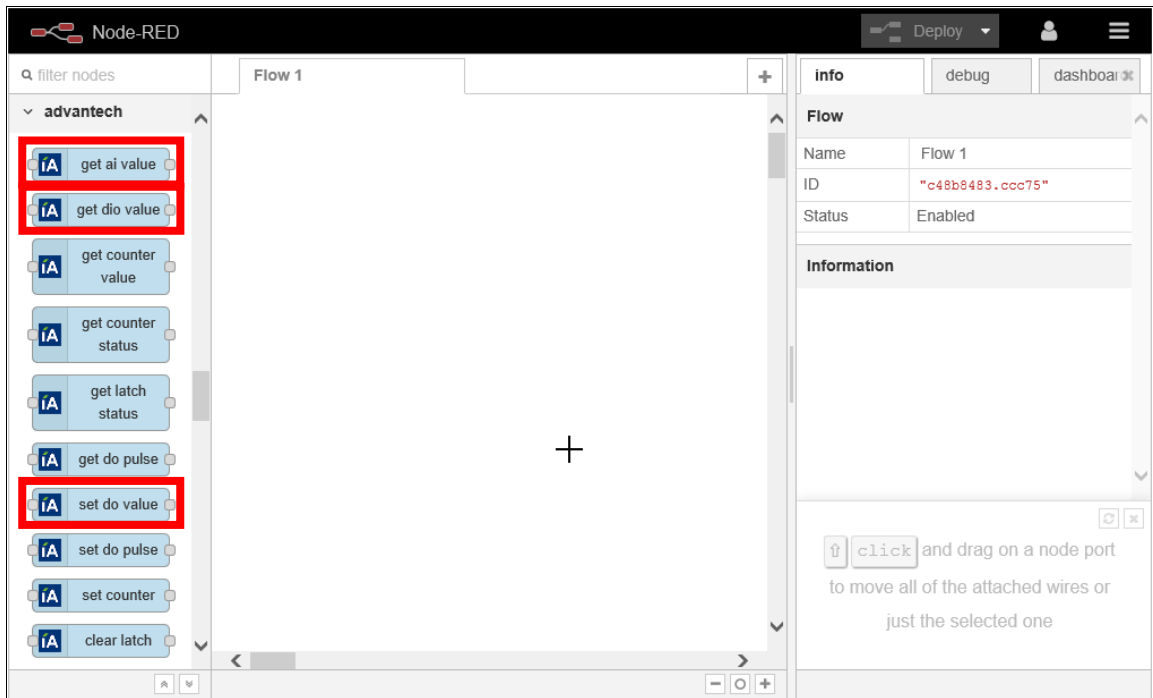
#### 4.1.1. Inject: 在输入 (input) 文件夹下可以找到 *inject* 节点



#### 4.1.2. Http request: 在功能 (function) 文件夹下可以找到 http request 节点



4.1.3. Set/get nodes: 在 Advantech 文件夹下可以找到 *get ai value*、*get dio value*、*set do value* 节点



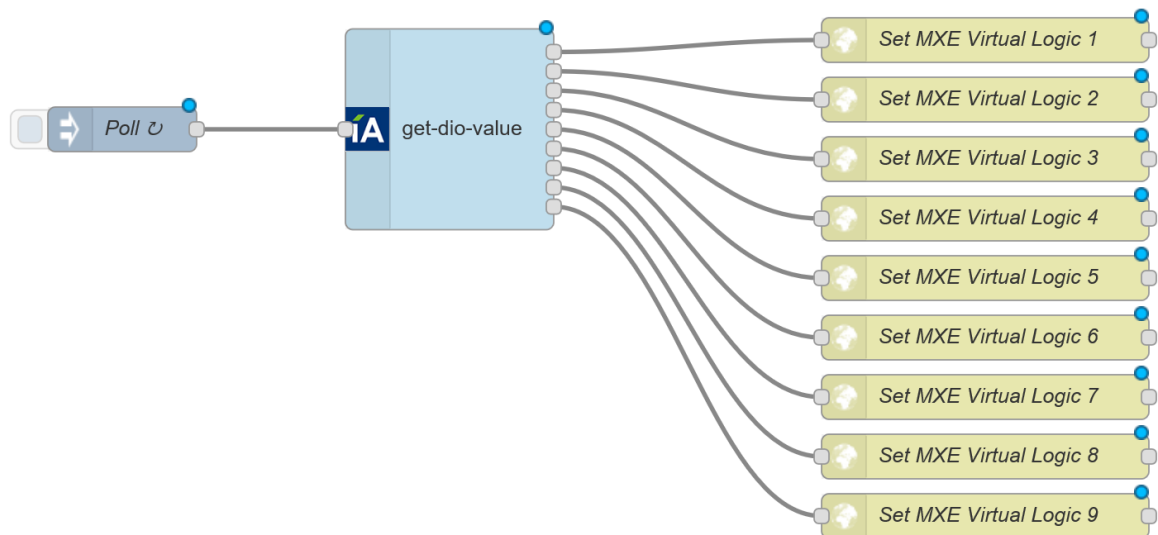
## 4.2. 运行于 ADAM-6750 的通用 Node-RED 示例

下面 3 个示例中 Inject 节点被配置为定期（每秒）触发设置/获取。这就是通常所说的调查 “Polling”，因此该节点标记为 “Poll”，详见示例。

### 4.2.1. 例 1: 获取 ADAM-6750 数字输入/输出状态（“On”、“Off”），并设置 MXE Virtual Logic values（“0”、“1”）

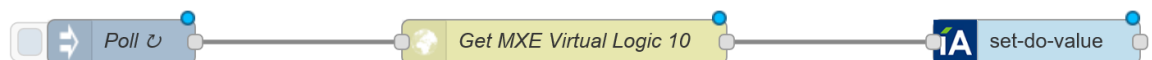
前面 5 个 *get-dio-value* 输出是 ADAM-6750 五个数字输入的状态。

后面 4 个 *get-dio-value* 输出是 ADAM-6750 四个数字输出的状态。



### 4.2.2. 例 2: 获取 MXE Virtual Logic status（“0”、“1”），并设置 ADAM-6750 数字输出状态（“On”、“Off”），

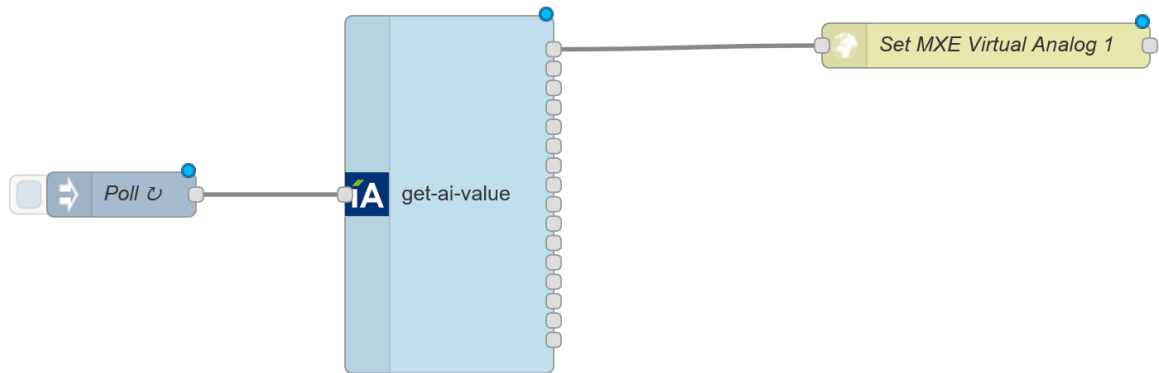
*Set-do-value* 模块可以配置用来设置 4 个 ADAM-6750 数字输出中的其中 1 个。



4.2.3. **例 3:** 获取 ADAM-6750 模拟输入数值 ( "0..1" ) , 并设置 MXE Virtual Analog 数值 ( "0..1" )

前面 8 个 *get-ai-value* 输出是 ADAM-6750 八个模拟输入的数值。

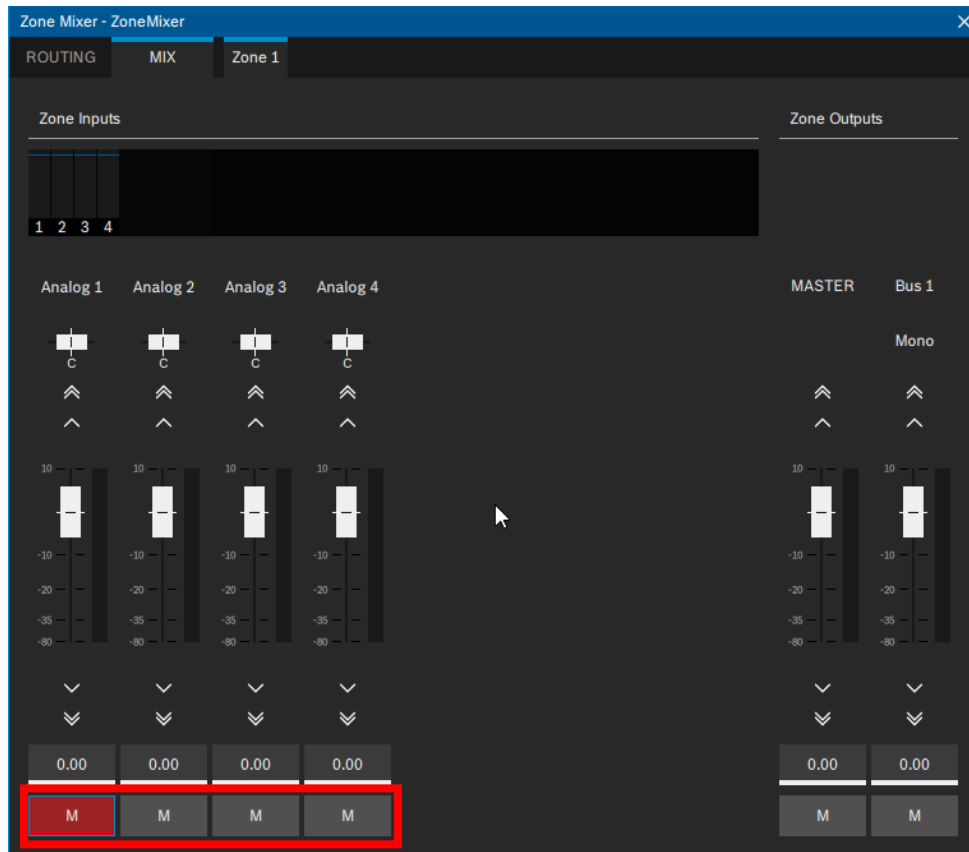
后面 8 个 *get-ai-value* 输出是这 8 个模拟输入的状态



### 4.3. 详细示例: 通过 ADAM-6750 数字输入控制 MXE 4 通道静音 (MUTE)

下面示例展示如何配置 1 台 ADAM-6750 以及 1 台 MXE5, 通过 ADAM 数字输入设置 MXE5 分区混音的静音 (MUTE)

#### 4.3.1. MXE 分区混音配置 (详见另一个 SONICUE 培训视频) , 通过 ADAM-6750 数字输入控制 Zone 1 模拟输入 1-4 的多个静音 (MUTE) 按键



#### 4.3.2. MXE Logic 逻辑配置 (详见另一个 SONICUE 培训视频)

每个逻辑任务 (Logic Tasks) 看起来对应不同的虚拟逻辑数值 (Virtual Logic value) (1-4) , 并对应控制分区混音 Zone1 的输入通道 1-4 的静音 (Mute) 。



Matrix1 - DeviceSetting - SONICUE

File Matrix1 DSP Logic DEPLOY

- Logic Task
- Logic Input
- Logic Output
- Analog Task
- Analog Input
- Analog Output
- Devices

**Logic Task: Zone1 Ch1 Mute**

Name: "Zone1 Ch1 Mute"

Input: Device "Matrix1 (MXE5): MXE-f56027\_oca\_tcp.local"

Logic Debug #5: MXE Virtual Logic 1

Value:

Output: Device "Matrix1 (MXE5): MXE-f56027\_oca\_tcp.local"

MXE Mute Zone 1 Input 1

**Logic Task: Zone1 Ch2 Mute**

Name: "Zone1 Ch2 Mute"

Input: Device "Matrix1 (MXE5): MXE-f56027\_oca\_tcp.local"

Logic Debug #6: MXE Virtual Logic 2

Value:

Output: Device "Matrix1 (MXE5): MXE-f56027\_oca\_tcp.local"

MXE Mute Zone 1 Input 2

**Logic Task: Zone1 Ch3 Mute**

Name: "Zone1 Ch3 Mute"

Input: Device "Matrix1 (MXE5): MXE-f56027\_oca\_tcp.local"

Logic Debug #7: MXE Virtual Logic 3

Value:

Output: Device "Matrix1 (MXE5): MXE-f56027\_oca\_tcp.local"

MXE Mute Zone 1 Input 3

**Logic Task: Zone1 Ch4 Mute**

Name: "Zone1 Ch4 Mute"

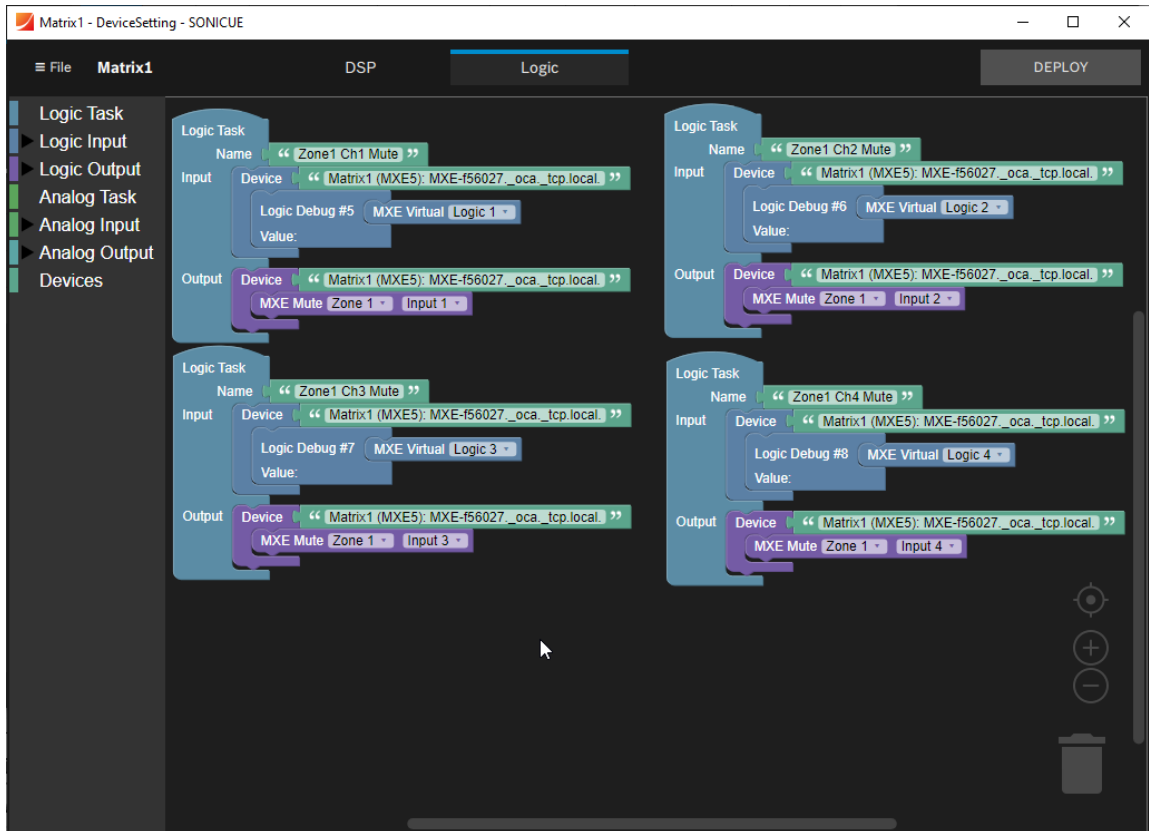
Input: Device "Matrix1 (MXE5): MXE-f56027\_oca\_tcp.local"

Logic Debug #8: MXE Virtual Logic 4

Value:

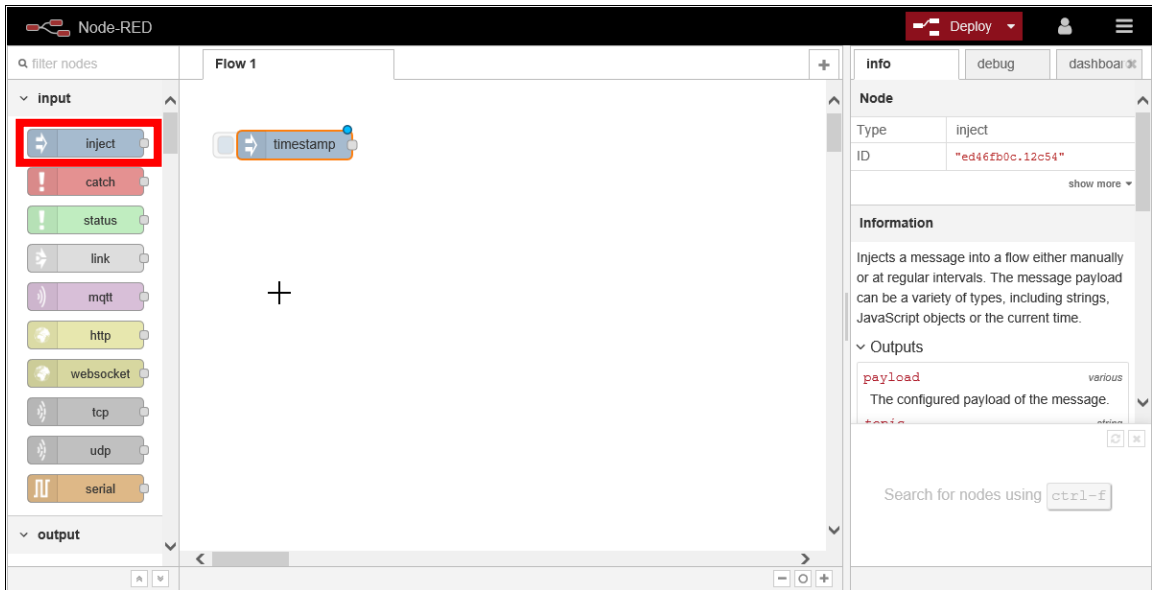
Output: Device "Matrix1 (MXE5): MXE-f56027\_oca\_tcp.local"

MXE Mute Zone 1 Input 4



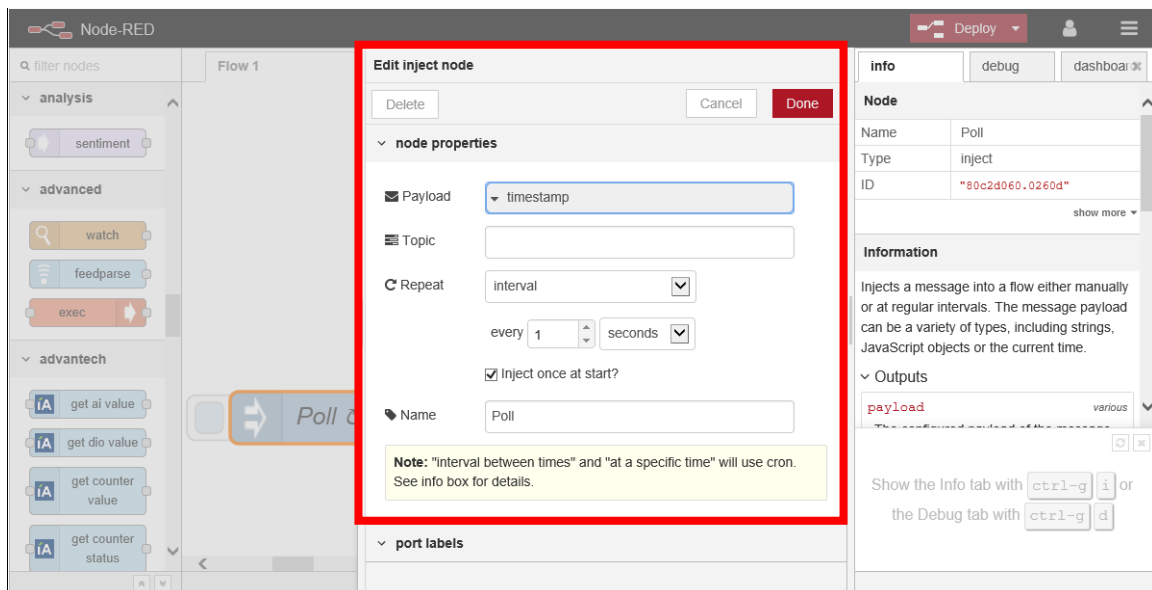
### 4.3.3. Node-RED 配置 (详见如下步骤)

#### Step 1: 从输入文件夹添加 *inject* 节点

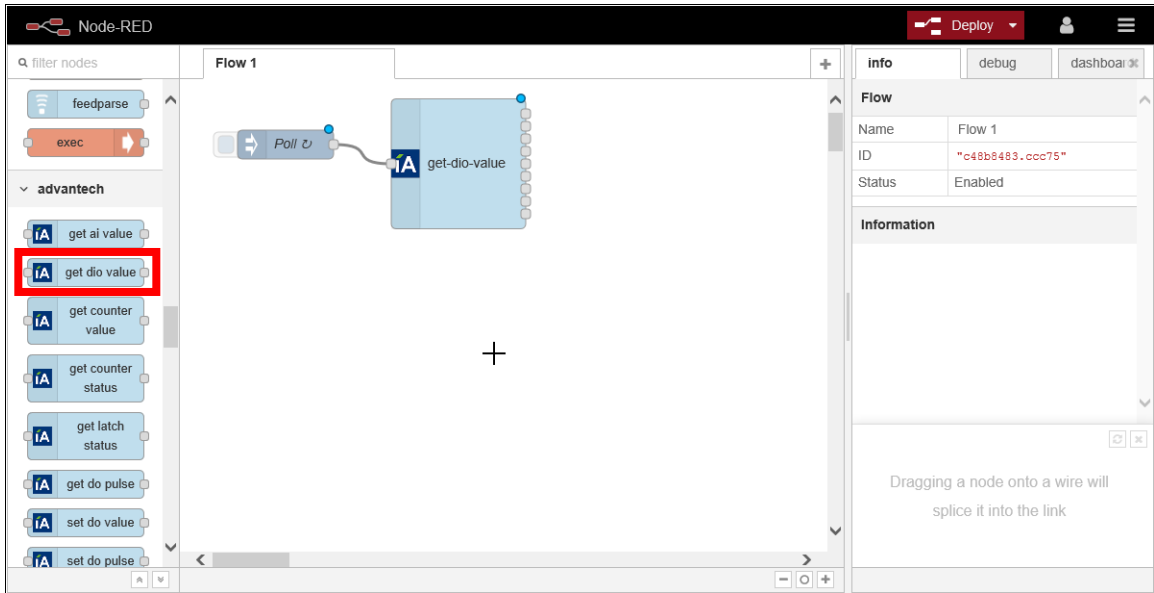


**Step 2:** 双击标记为 *timestamp* 的 inject 节点, 打开 *Edit inject node* 节点编辑窗口并完成如下调整:

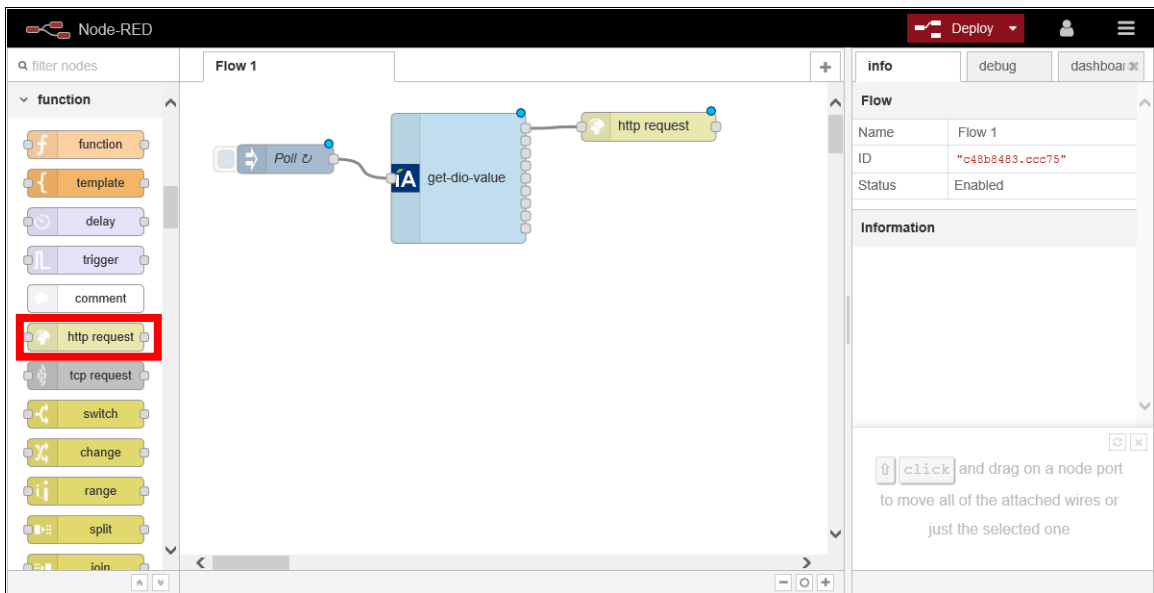
- 设置 *Repeat* 为 *interval*, 每 1 秒
- 勾选 *Inject once at start*
- 变更命名为 "Poll"



**Step 3:** 从 *advantech* 文件夹添加 1 个 *get-dio-value* 节点，并连接到如下截图所示的标记为 Poll 的 inject 节点

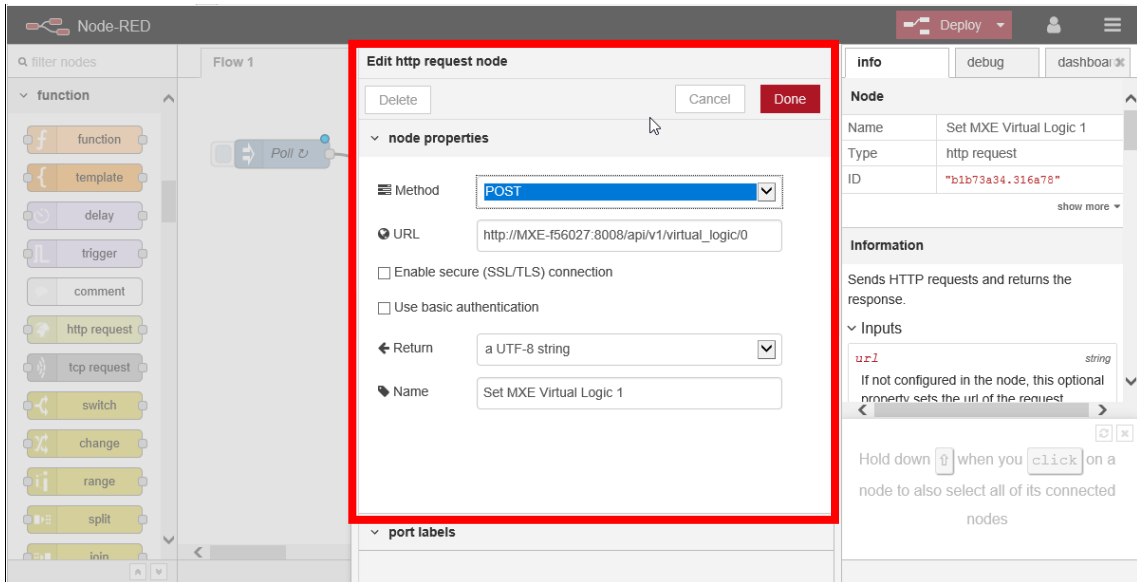


**Step 4:** 从 *function* 文件夹添加 1 个 *http request* 节点，并连接到如下截图所示的 *get-dio-value* 节点



**Step 5:** 双击 *http request* 节点, 打开 *Edit http request node* 节点编辑窗口并完成如下调整 (示例 MXE 名称 “MXE-f56027” ) :

- 设置 *Method* 为 *POST*
- *URL* 输入 “*http://MXE-f56027:8008/api/v1/virtual\_logic/0*”
- *Name* 设置为 “*Set MXE Virtual Logic 1*”



**Step 6:** 复制标签为 *Set MXE Virtual Logic 1* 的 *http request* 节点, 并粘贴 3 次

对粘贴的第 1 个节点做如下调整 (切记: 本例 MXE 名称 “MXE-f56027” -> 你需要使用实际的 MXE 名称) :

- *URL* 调整为 “*http://MXE-f56027:8008/api/v1/virtual\_logic/1*”
- *Name* 调整为 “*Set MXE Virtual Logic 2*”

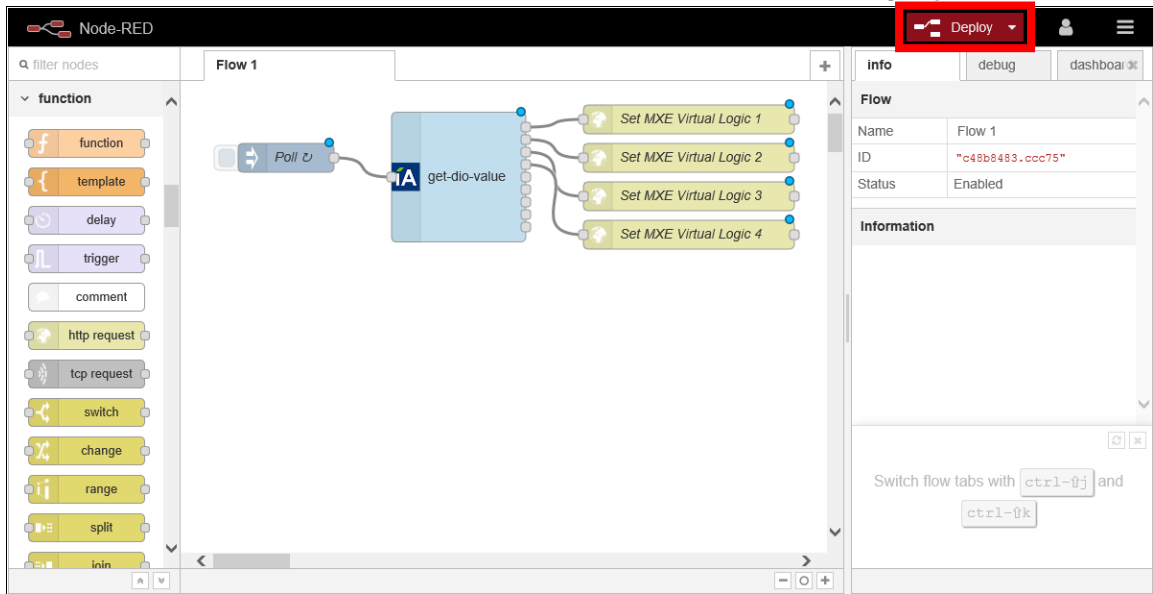
对粘贴的第 2 个节点做如下调整:

- *URL* 调整为 “*http://MXE-f56027:8008/api/v1/virtual\_logic/2*”
- *Name* 调整为 “*Set MXE Virtual Logic 3*”

对粘贴的第 3 个节点做如下调整:

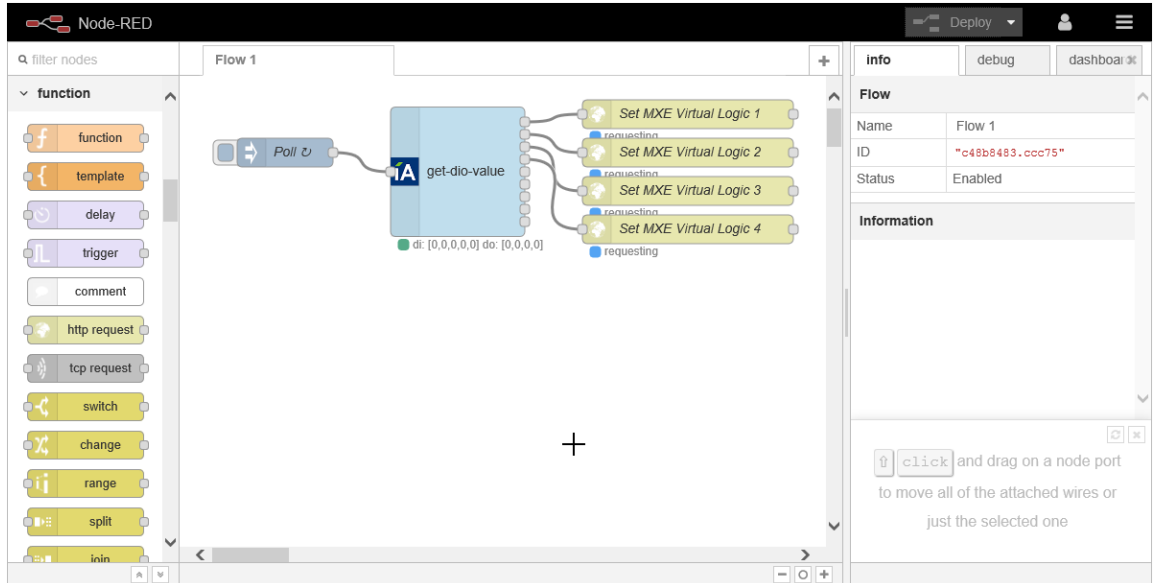
- *URL* 调整为 “*http://MXE-f56027:8008/api/v1/virtual\_logic/3*”
- *Name* 调整为 “*Set MXE Virtual Logic 4*”

当你完成了流 (flow) *Flow 1*，应该如下所示，最后你可以点击 **Deploy** 按钮



该 flow 开始工作:

- ADAM-6750的数字输入和输出状态显示在 *get-dio-value* 节点的下方，同时 http request 节点开始请求 *requesting*



**DONE!**