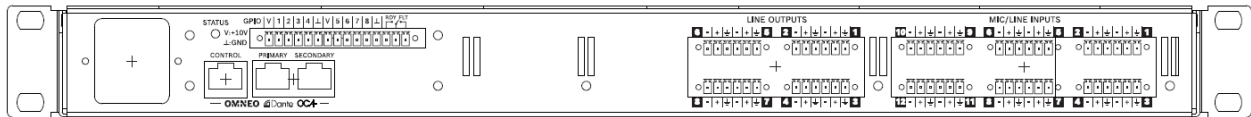


## Application Note

### Remote control of MXE Matrix Mix Engines via the MXE API and the OMNEO Dante OCA network interface using Advantech ADAM 67xx IoT gateway(s)

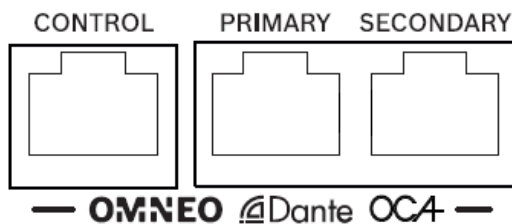
MXE Matrix Mix Engines are equipped with an OMNEO Dante OCA network interface for connecting to other systems, using CAT cables and Ethernet network switches.



**Image 1:** MXE rear view

The network interface (*OMNEO Dante OCA*) can be found on the MXE's rear panel. It offers in total three network ports: *CONTROL*, *PRIMARY* and *SECONDARY*.

The three network ports can be configured via SONICUE to run either in Transparent, RSTP or Glitch-Free mode.



**Image 2:** MXE network interface detail view

## Requirements

MXE Matrix Mix Engine with firmware 1.0.2561 (or higher)

MXE Application Programming Interface (API) activated via MXE front panel menu

SONICUE Sound System Software 1.2 (or higher) installed on computer

## Documentation (recommended in addition to this application note)

A detailed description of all hardware connection options of the Advantech ADAM 6700 Series IoT gateways can be found in the device manual(s).

# 1. Advantech ADAM 6700 Series IoT gateways (third party)

## Applications

Advantech's ADAM 6700 Series IoT gateways offer many interesting applications with MXE.

- GPIO extension -> number of GPIOs (General Purpose Inputs and Outputs)
- GPIO extension -> remote location of GPIOs
- Email notification (free of charge)
- SMS notification (cost depends on service provider)
- Interface to 3<sup>rd</sup> party systems (requires advanced programming skills)

## Programming

Programming can be done via IBM's Node-RED, a visual, flow-based programming tool. It is pre-installed on the ADAM-6700 series IoT gateways. On YouTube many tutorial videos can be found, as a nice introduction on how to program with Node-RED.

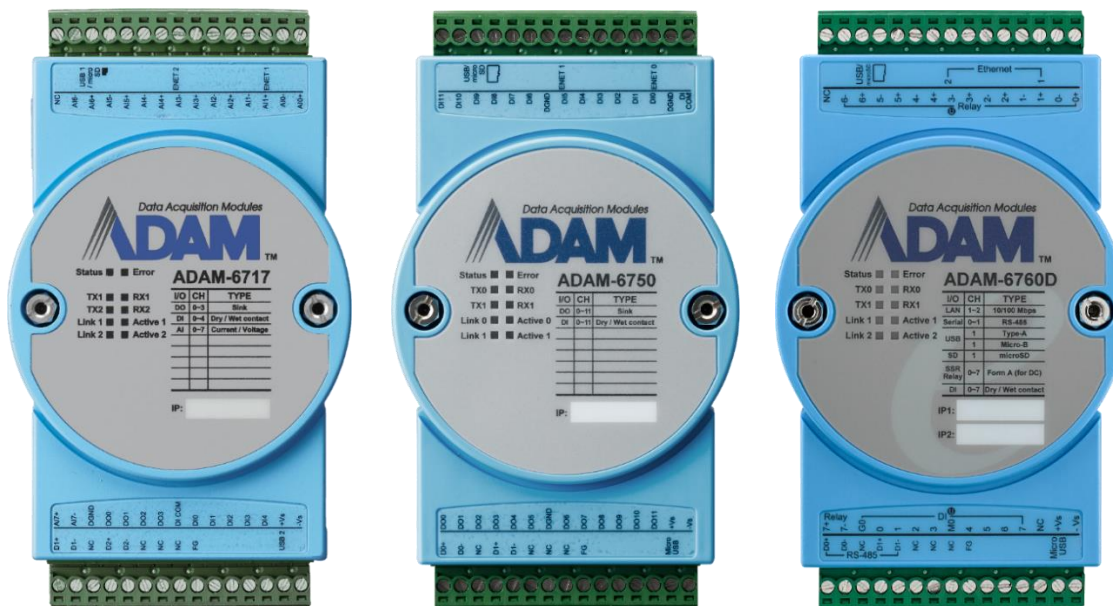
## Installation

All models have the same housing and are intended to be installed on a DIN-rail.

## Variants

The following ADAM 67xx IoT gateways are available, only difference being the connectivity, with the 6750 model having been successfully tested with MXE:

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

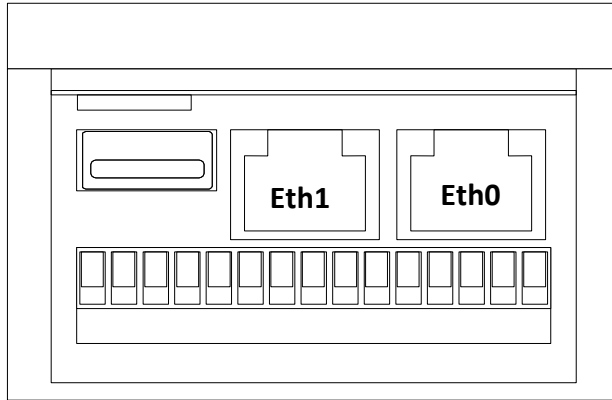


**Image 3:** Advantech ADAM-6717 (left), ADAM-6750 (center) and ADAM-6760D (right)

## Ethernet ports

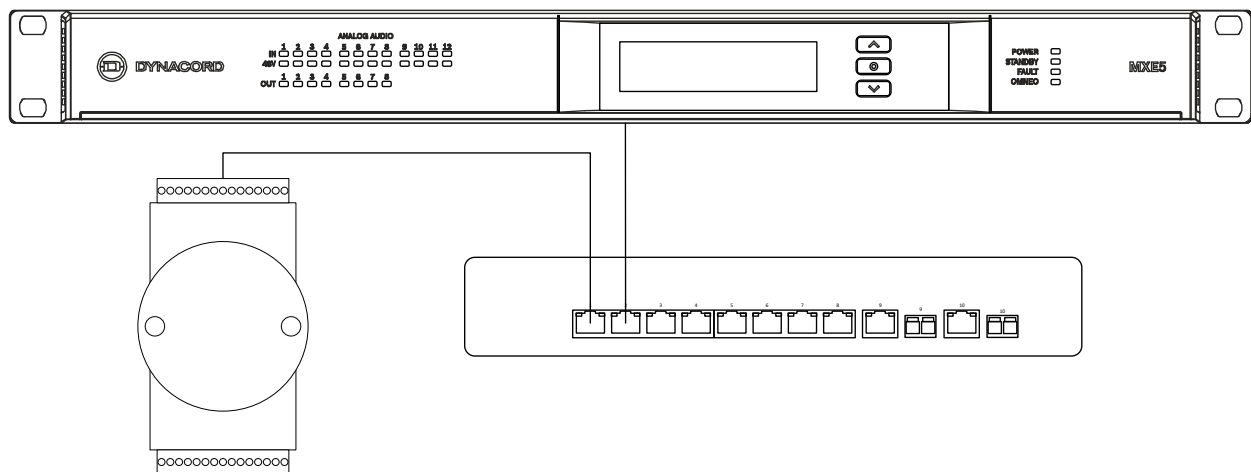
Besides the analog or digital inputs and outputs, that are different on each of the models, all models feature dual Ethernet ports, being equivalent in functionality.

The two Ethernet ports (*Eth0* and *Eth1*) can be found on the top side of the ADAM 6700 devices:



**Image 4:** Advantech ADAM-6700 series network interface detail view

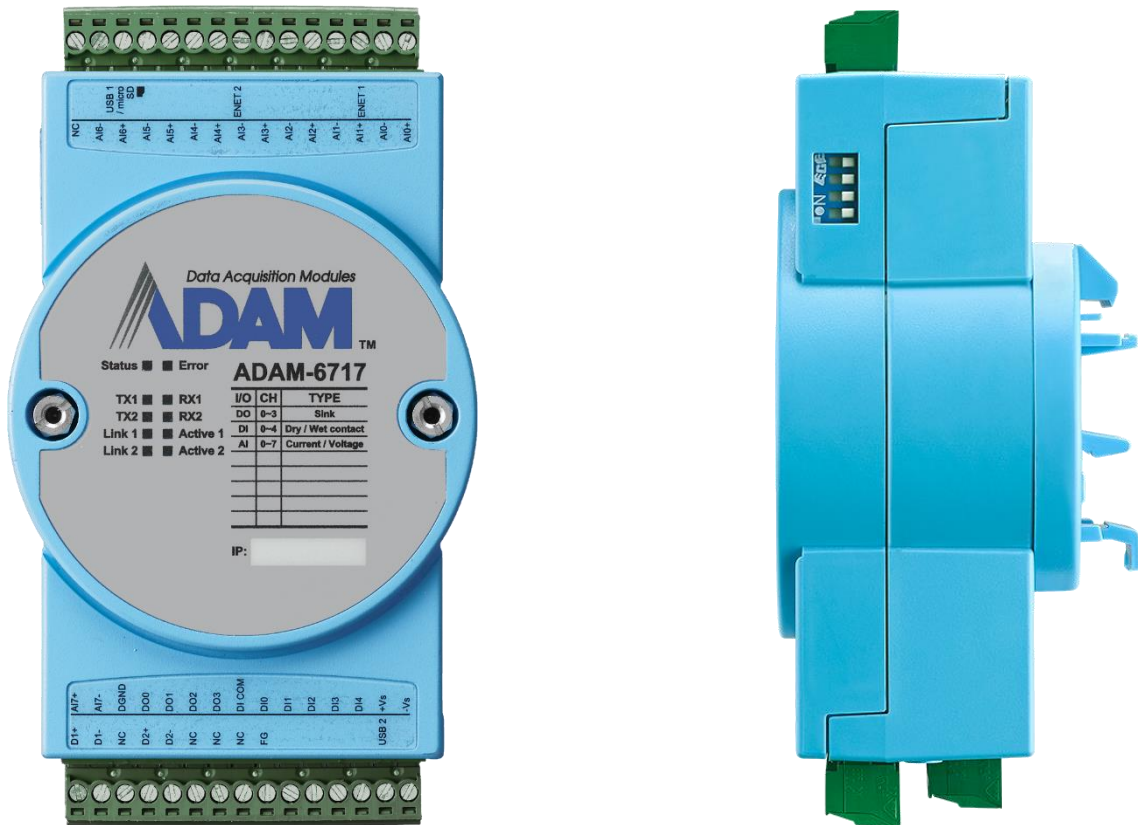
### Basic set up – network connection



**Image 5:** Advantech ADAM-6700 series IoT gateway connection to Dynacord MXE via Ethernet network switch

## 1.1. Advantech ADAM-6717 IoT gateway

### Product images



**Image 6:** Advantech ADAM-6717 IoT gateway front view (left) and side view (right)

### Brief description

The ADAM-6717 IoT gateway is part of Advantech's ADAM 6700 Series of IoT gateways.

It features the following inputs and outputs:

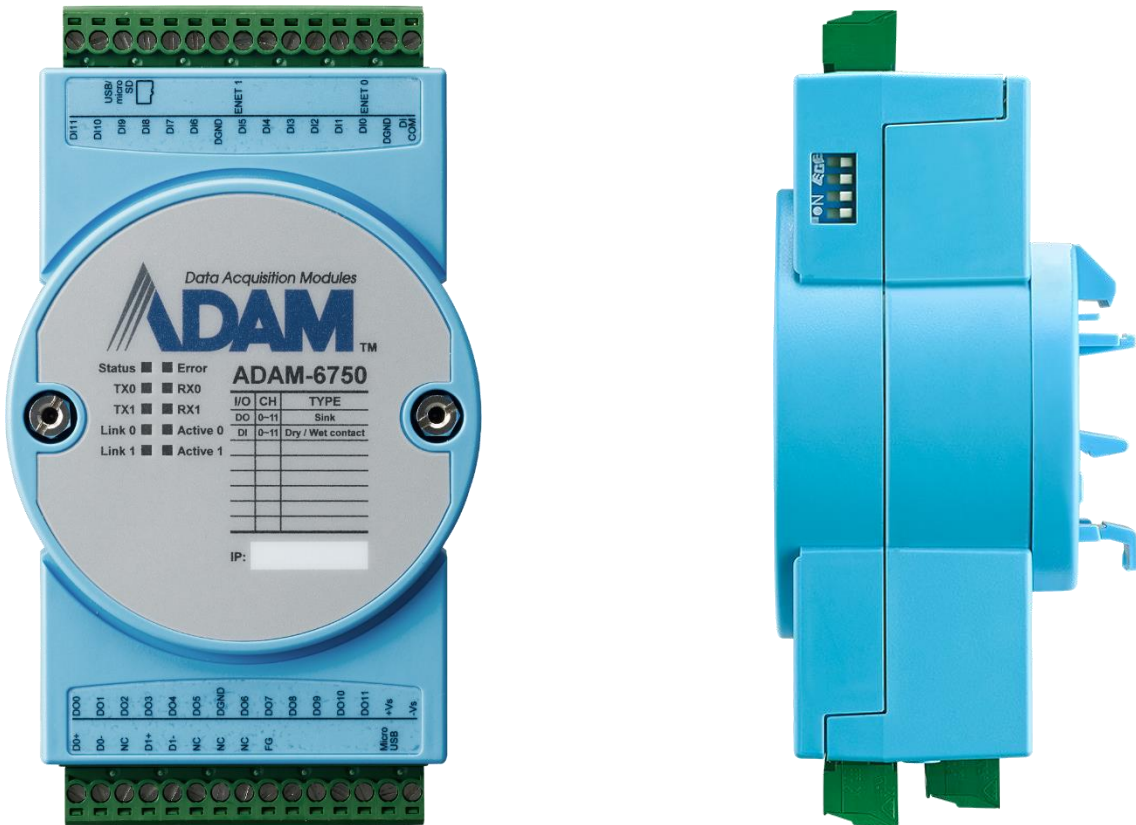
- 8 x analog input
- 5 x digital input
- 4 x digital output

### Application examples

- Use a potentiometer (typically 10 kOhm) connected to an ADAM analog input to control an MXE DSP level value.
- Use a hardware switch or relay contact connected to an ADAM digital input to control an MXE DSP mute or set a fault flag.
- Use an LED connected to an ADAM digital output to signalize a fault in the MXE.
- Use a small relay connected to an ADAM digital output to switch larger currents or voltages potential-free.

## 1.2. Advantech ADAM-6750 IoT gateway

### Product images



**Image 7:** Advantech ADAM-6750 IoT gateway front view (left) and side view (right)

### Brief description

The ADAM-6717 IoT gateway is part of Advantech's ADAM 6700 Series of IoT gateways.

It features the following inputs and outputs:

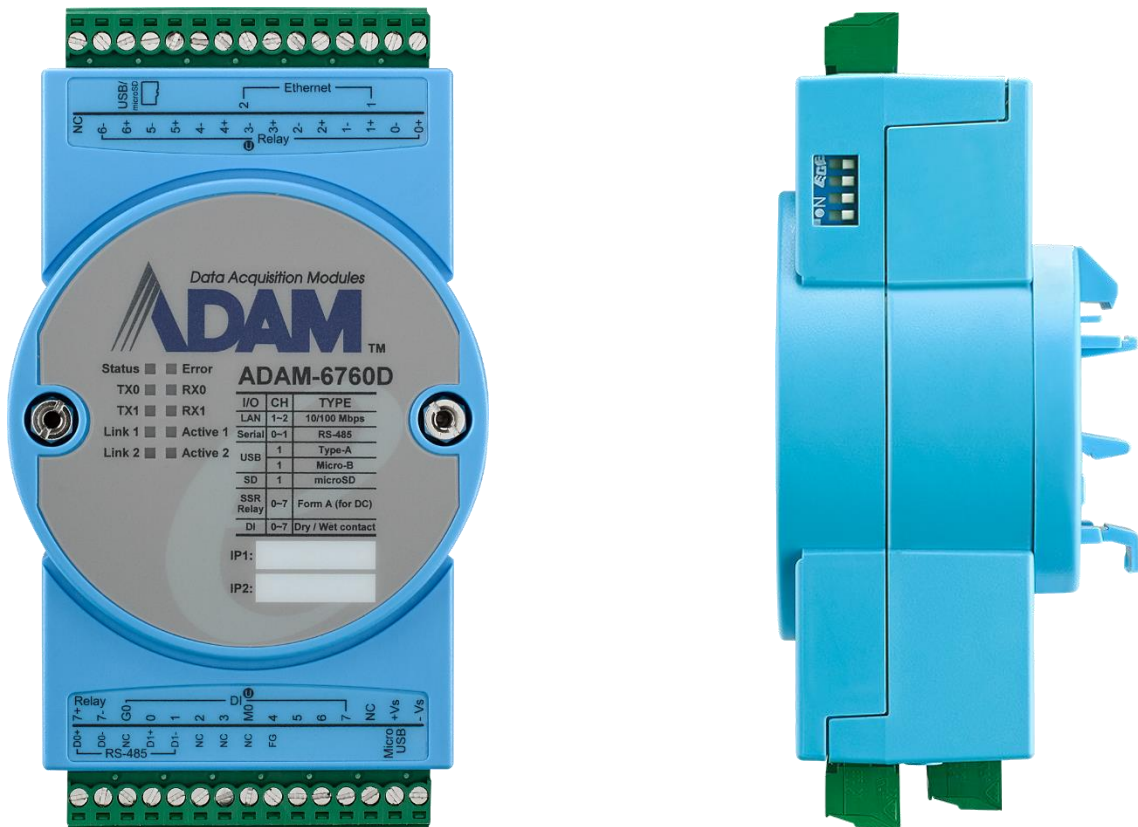
- 12 x digital input
- 12 x digital output

### Application examples

- Use a hardware switch or relay contact connected to an ADAM digital input to control an MXE DSP mute or set a fault flag.
- Use an LED connected to an ADAM digital output to signalize a fault in the MXE.
- Use a small relay connected to an ADAM digital output to switch larger currents or voltages potential-free triggered by MXE logic.

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

#### Product images



**Image 8:** Advantech ADAM-6760 IoT gateway front view (left) and side view (right)

#### Brief description

The ADAM-6717 IoT gateway is part of Advantech's ADAM 6700 Series of IoT gateways.

It features the following inputs and outputs:

- 8 x digital input
- 8 x relay output (PhotoMOS SPST)

#### Application examples

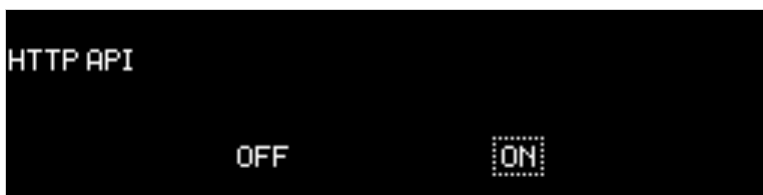
- Use a hardware switch or relay contact connected to an ADAM digital input to control an MXE DSP mute or set a fault flag.
- Use a pair of resistors connected to an ADAM relay output to realize a supervised, potential-free, fault signalization to other systems triggered by MXE logic.

## 2. MXE HTTP API

The MXE's HTTP API serves as a simple and easy to integrate interface to an OCA controlled system.

### 2.1. Configuration

The HTTP API is disabled by default. To activate, turn on the API in Main Menu on the MXE's front display



### 2.2. HTTP Server

MXE devices run a local HTTP server for API purposes on port 8008 (http) or 443 (https).

The root HTTP URI for resources described in this document is

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

The root HTTPS URI for resources described in this document is

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

The hostname can be derived from the device type and MAC address:

mxe-<last-3-bytes-of-MAC-address>

Example: An MXE5 with MAC address 00:1C:44:F5:60:59 will set it's hostname to

mxe-f56059

### 2.3. HTTP Dialect

All properly formed requests return a JSON-encoded payload with an application/json MIME type.

Each resource accepts JSON-encoded Parameters which should be supplied as POST payload.

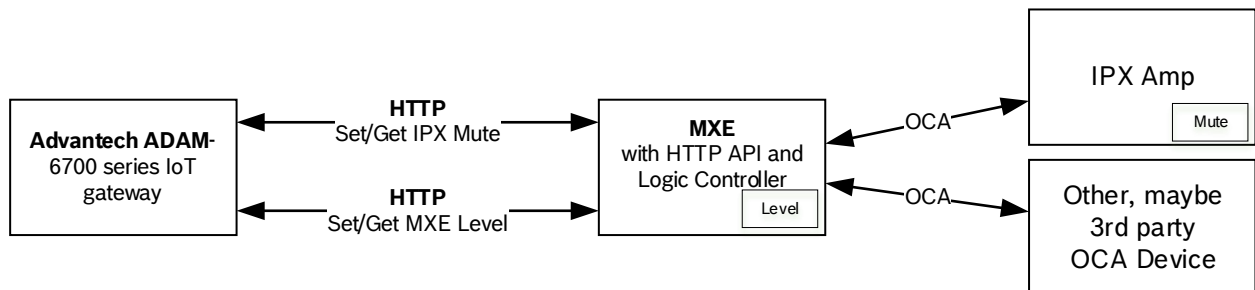
Error responses carry a 4xx HTTP response code and a text/plain response body.

### 2.4. Resources

Resource	Parameter	Example
/virtual_logic	JSON Array of 100 Booleans indicating the state of the virtual logic values on the corresponding index	[true,false,false,...]
/virtual_logic/<0..99>	Boolean indicating the state of the virtual logic value	true
/virtual_analog	JSON Array of 100 Floats indicating the value of the virtual analog value on the corresponding index	[0, 1.2, -42.27, ...]
/virtual_analog/<0..99>	Float indicating the value of the virtual analog value	-80.0

### 2.5. Application Examples

In a system as described in the following picture, it is possible to Set and Get the Mute state of an Amp Channel or the Zone Master Level of another Matrix via the HTTP API, thus allowing external control of that functions from a 3<sup>rd</sup> party device or software, in this example an Advantech ADAM-6700 series IoT gateway.





### 2.5.1. Set / Get Amp Channel Mute

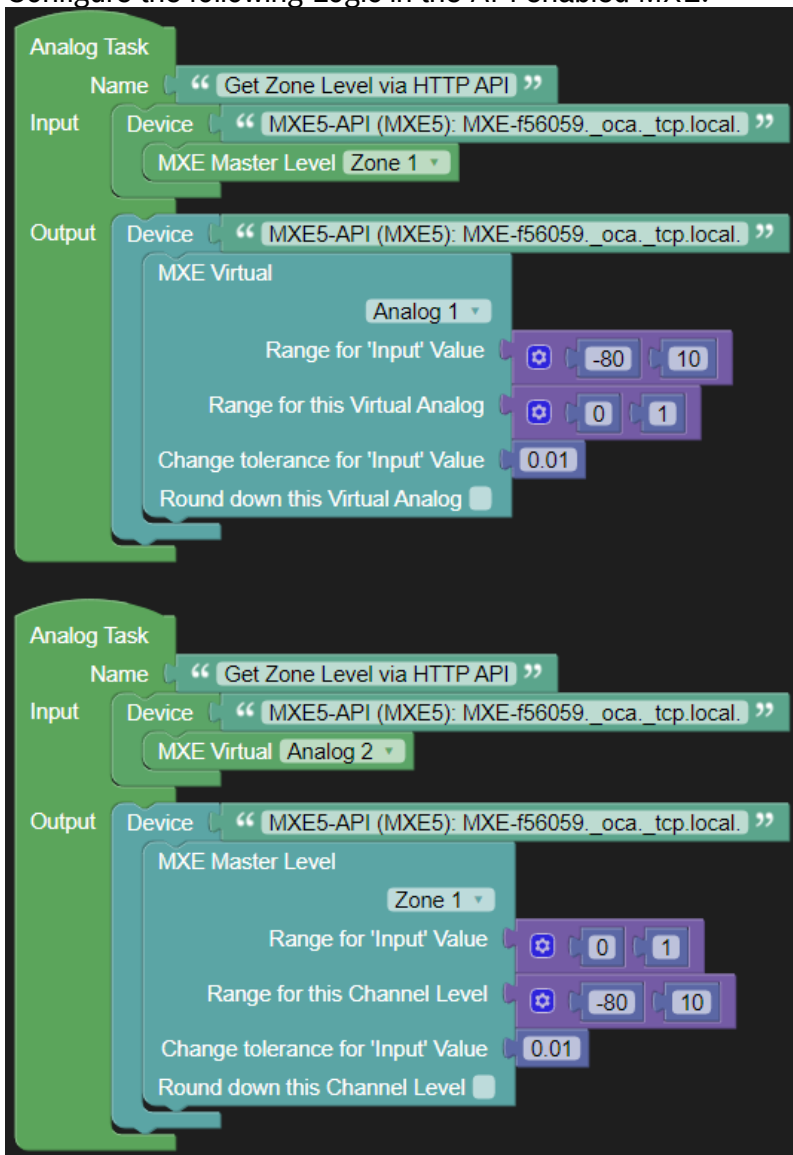
1. In SONICUE, configure the following Logic for the API enabled MXE.



2. Note, that virtual logic block indices are 0-based in the API and 1-based in the Logic GUI.
3. Note, that two virtual logic blocks are used for getting and setting mute. Using the same virtual logic block could result in unwanted data races.

## 2.5.2. Set / Get Zone Master Level

1. Configure the following Logic in the API enabled MXE.



2. This example makes use of the "Range..." options of the Analog Outputs to achieve a normalization of the API value in a 0...1 range. This can help to achieve a stable API even on changing requirements on the actual allowed parameter range. E.g. the installation could require that the user shall no longer be allowed to level down below -40 dB. The website code requires no update, the only change would be the setting in the "Range..." for the Zone level Output, i.e. -40 ... 10.
3. This example uses a "Change tolerance" of 0.01 to avoid data feedback loops.

## 2.6. Security

All resources will require Basic HTTP authentication when OCA Control Security is enabled. Additionally, the insecure HTTP port will no longer be available in this mode. If OCA Control Security is enabled, the API webserver will accept credentials that match a given PSK Identity and Key. If you have shared a PSK with identity “HTTP-API”, you can use “HTTP-API” as user name and the shared passphrase as credentials for the Basic HTTP authentication.

The MXE uses a self-signed certificate for HTTPS connections.

Enabling OCA Control Security is not part of this API, refer to the OCA Standard [1] to learn how to enable Control Security via OCA and share a PSK or use an existing OCA Controller Software or Device.

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

### 3. Advantech ADAM/APAX Utility

The *Advantech ADAM/APAX Utility* tool can be downloaded from the [Advantech website > Support](#).

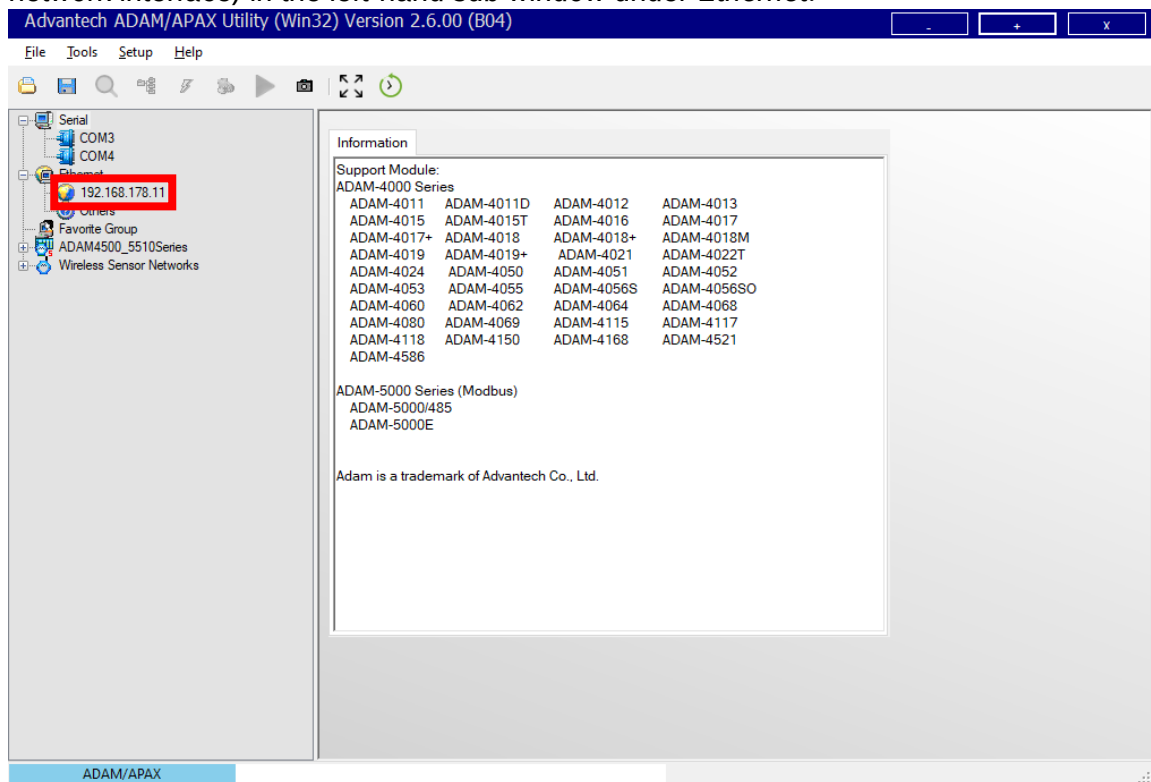
After the installation there'll be a desktop link on your desktop. You'll need administrator rights to launch the application.

Without DHCP server active the IP addresses of the ADAM will be by default:

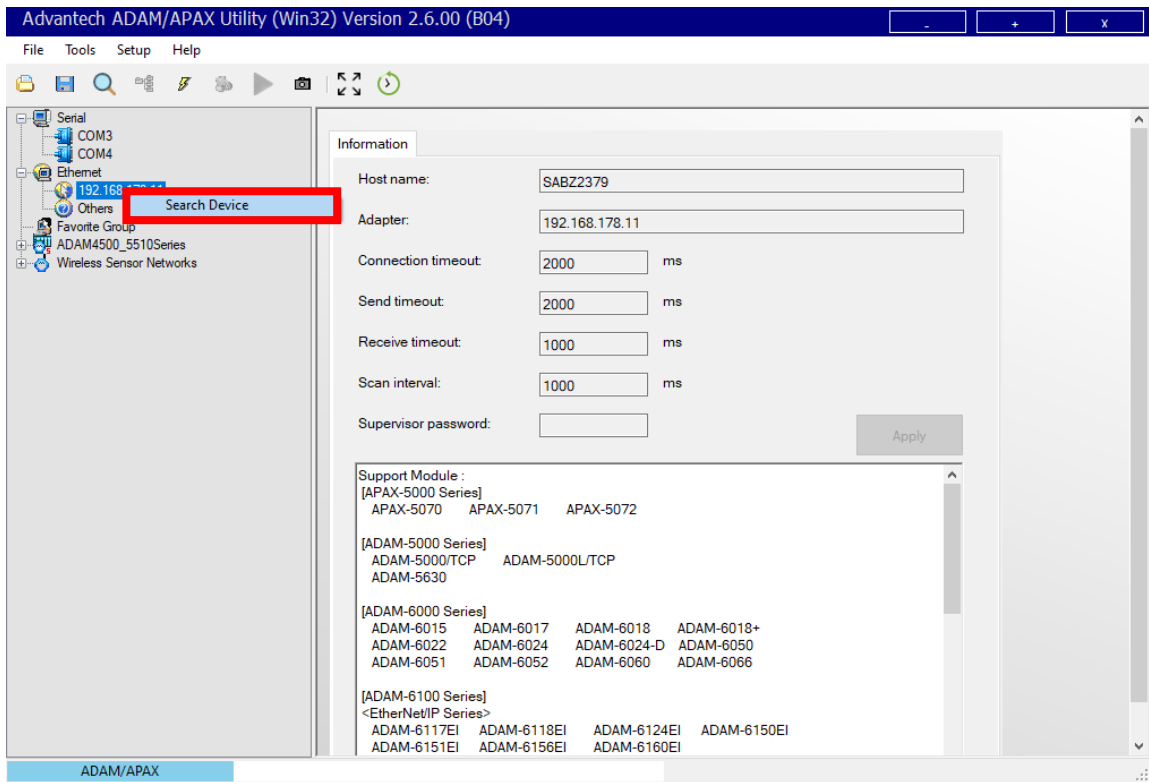
- Interface Eth0: 10.0.0.1
- Interface Eth1: 11.0.0.1

In the following screenshots, configuration of an ADAM with IP address 192.168.178.101 is shown. The PC's network interface was configured as 192.168.178.11.

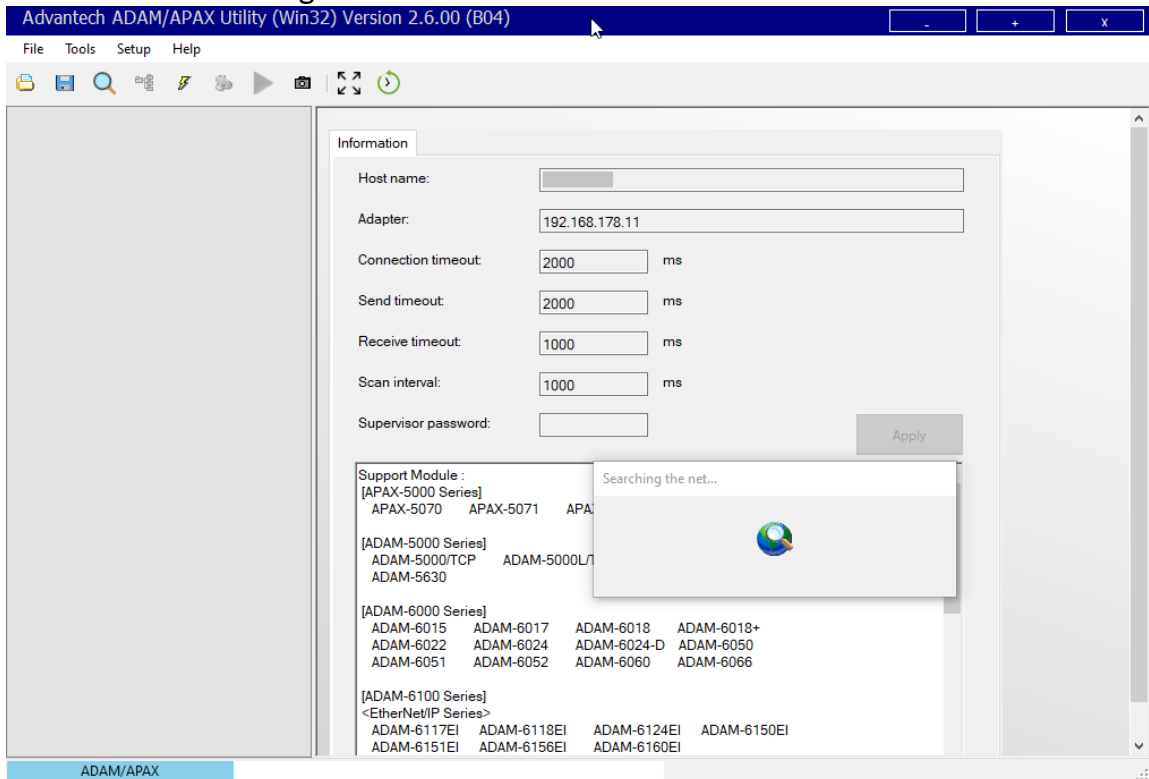
- 3.1. Open the *Advantech ADAM/APAX Utility* tool and right click on the IP address (= PC's network interface) in the left-hand sub window under *Ethernet*.



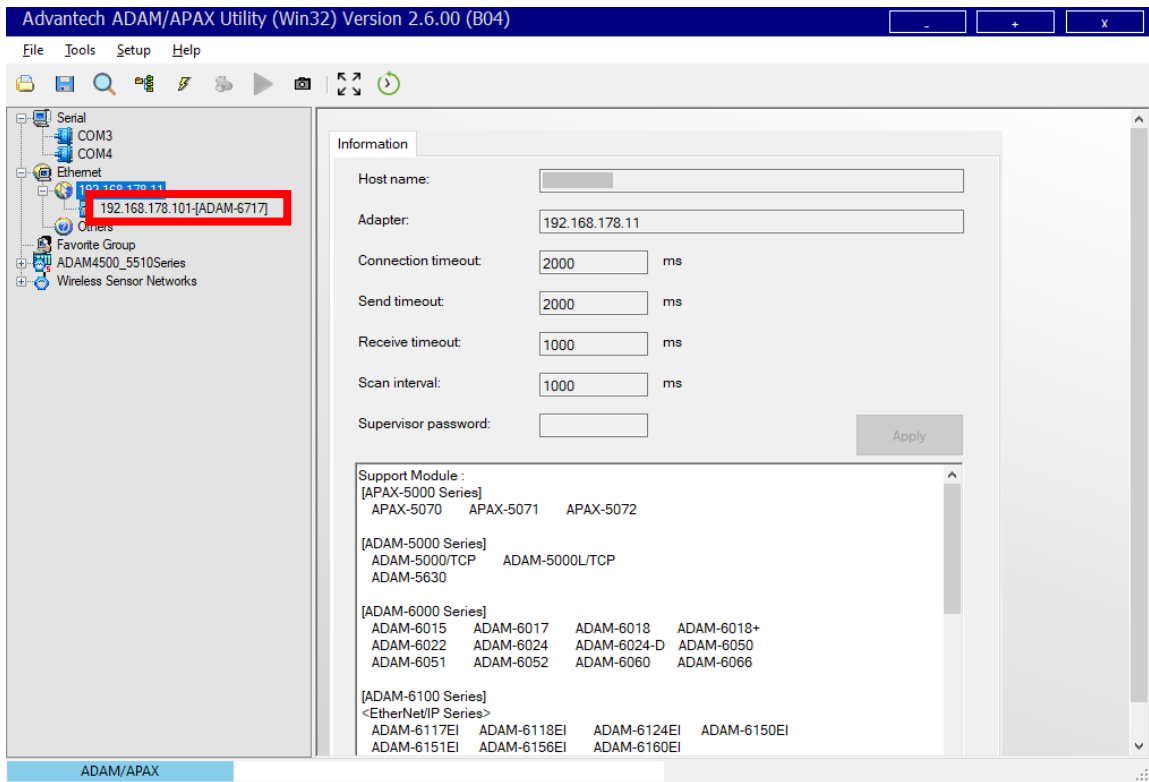
### 3.2. Choose *Search Device* to search for ADAMs



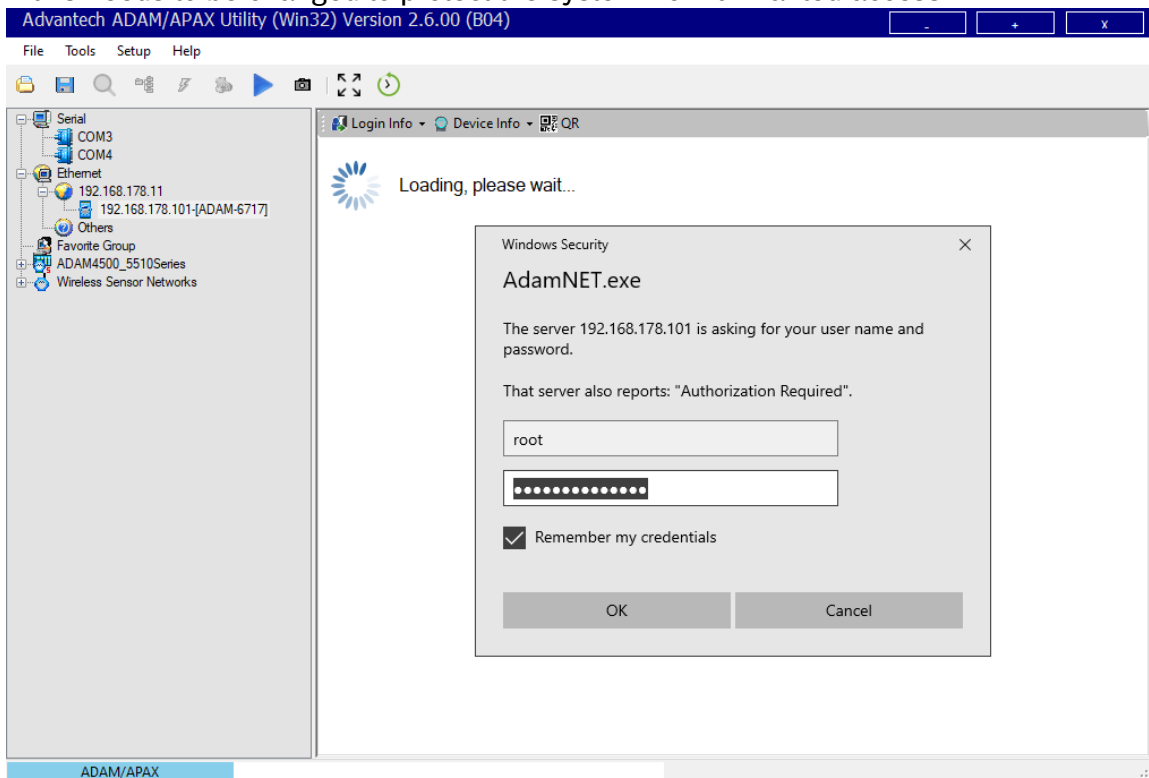
### 3.3. Wait until the *Searching the net...* window closes.



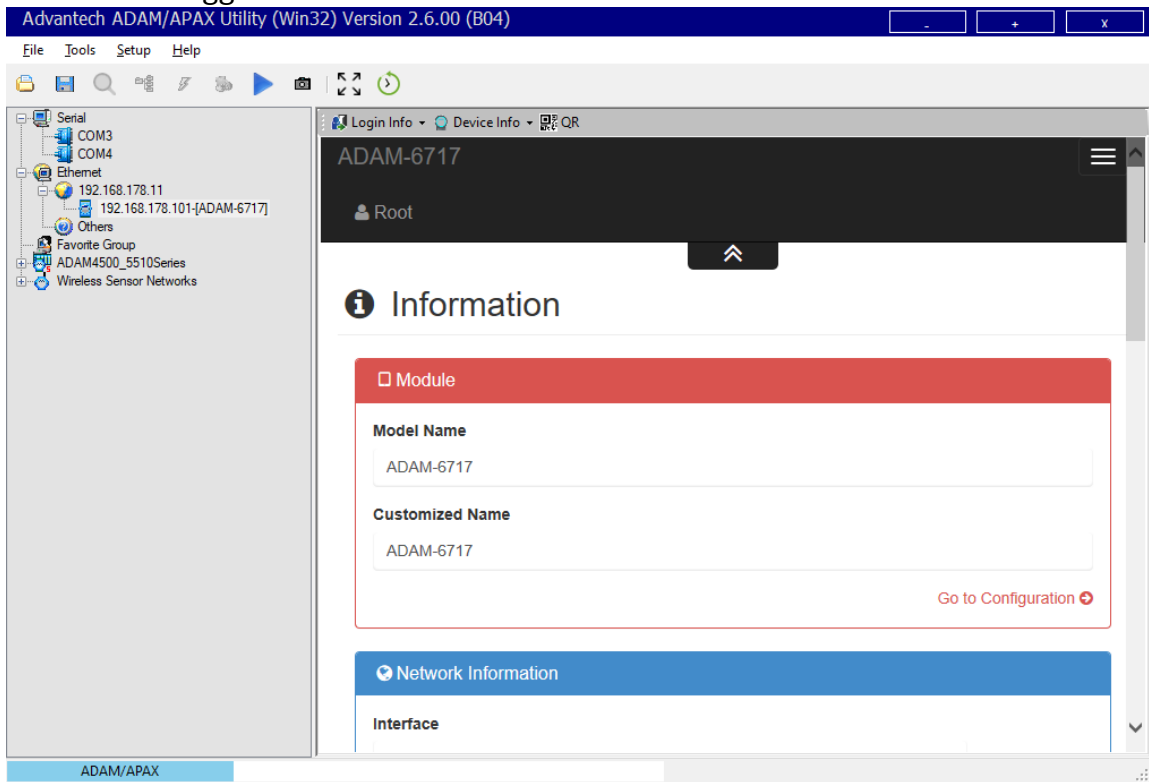
### 3.4. Double-click the ADAM that has been found



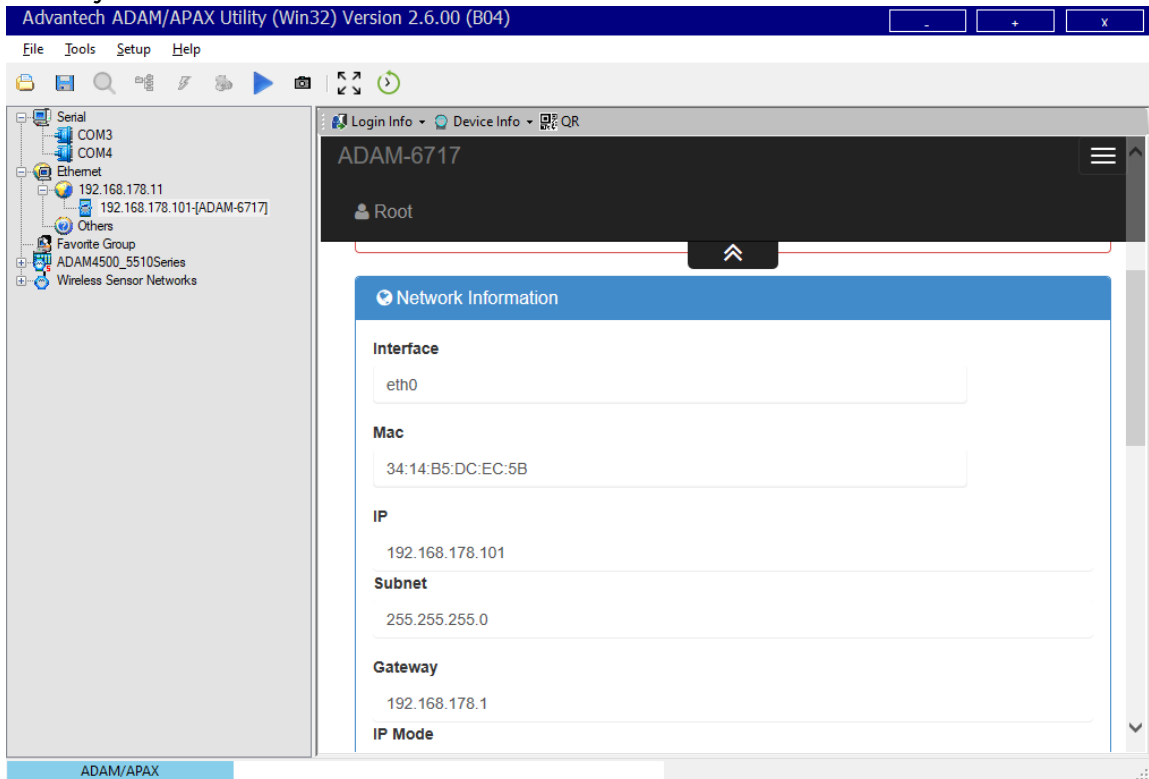
### 3.5. Login to the ADAM with default user “root” and default password “00000000” (eight 8’s) - > this needs to be changed to protect the system from unwanted access!




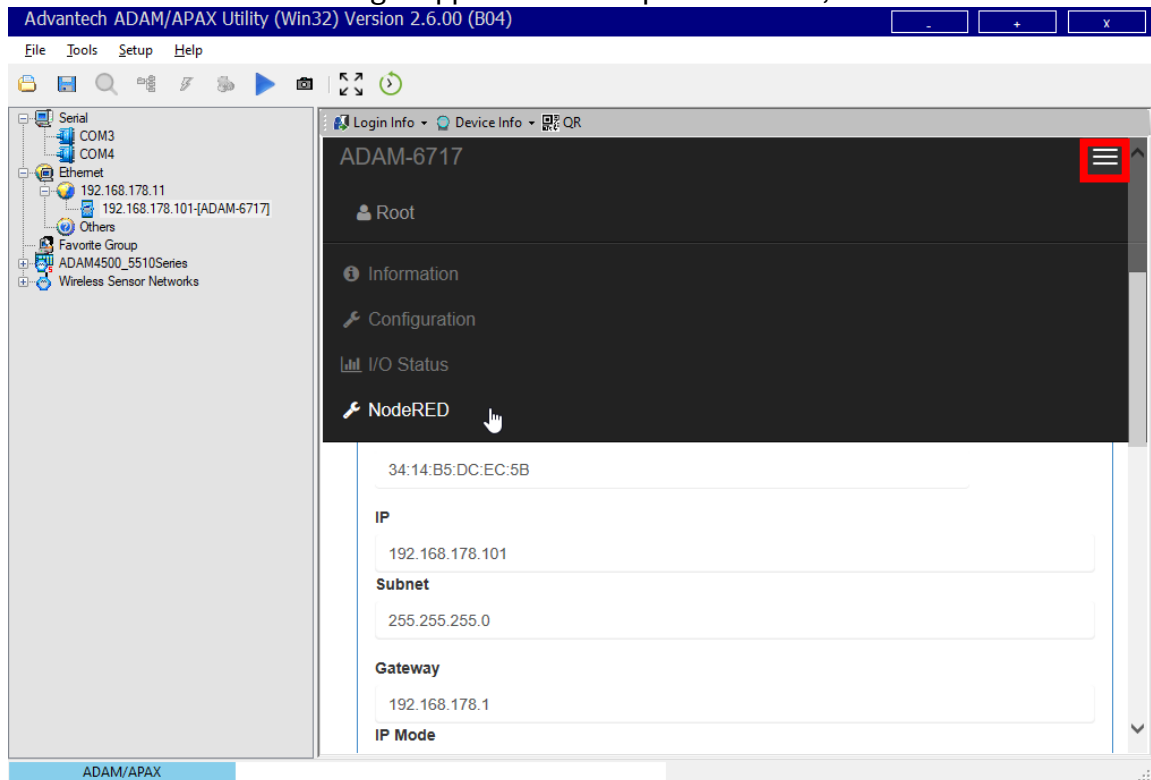
3.6. You are now logged-in and can check information about the *Module...*



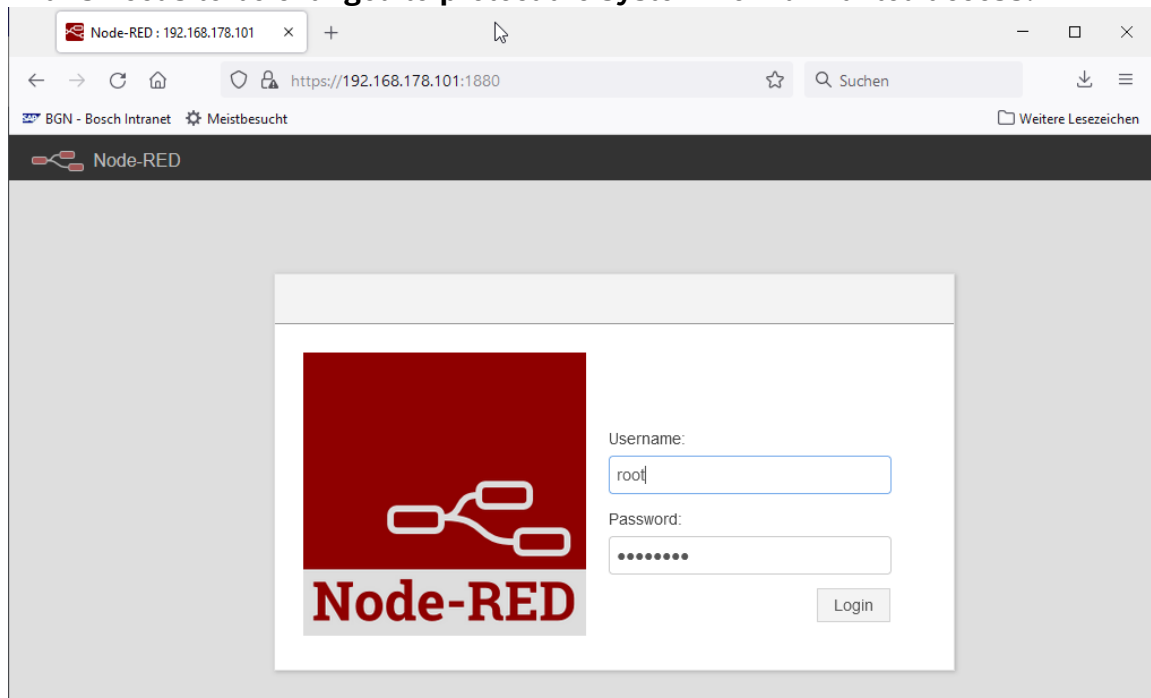
3.7. ...and you can check *Network Information* for *Interface eth0* and also *eth1*.



3.8. Click the -button in the right upper corner to open the menu, then click *NodeRED*



3.9. Login to Node-RED with default user “root” and default password “00000000” (eight 8’s)  
-> **this needs to be changed to protect the system from unwanted access!**



The ADAM’s Node-RED web site should now open; see chapter 4. *Node-RED* for next steps



## 4. Node-RED

IBM's Node-RED visual, flow-based programming tool is pre-installed on the Advantech ADAM-6700 series IoT gateways. On YouTube many tutorial videos can be found, as a nice introduction on how to program with Node-RED.

### 4.1. Basic concept

In Node-RED so-called flows are built by graphically linking nodes.

There are some core nodes, that are always available in Node-RED, and there often are special nodes for interfaces of devices running Node-RED, like the Advantech ADAM-6700 series' GPIOs.

If the functionalities provided by the (core) nodes and special nodes – such as the Advantech ADAM nodes – is not sufficient, it's possible to install additional node libraries or to write custom functions with JavaScript (not covered by this application note).

A description of all core nodes (Inject, Debug, Function, Change, Switch and Template) can be found on the [Node-RED website > documentation > The Core Nodes](#).

Other nodes, like Email, can be easily installed from various libraries and forums for Node-RED. On the Advantech ADAM-6700 series the Email node is pre-installed, and so are all other nodes used in the examples.

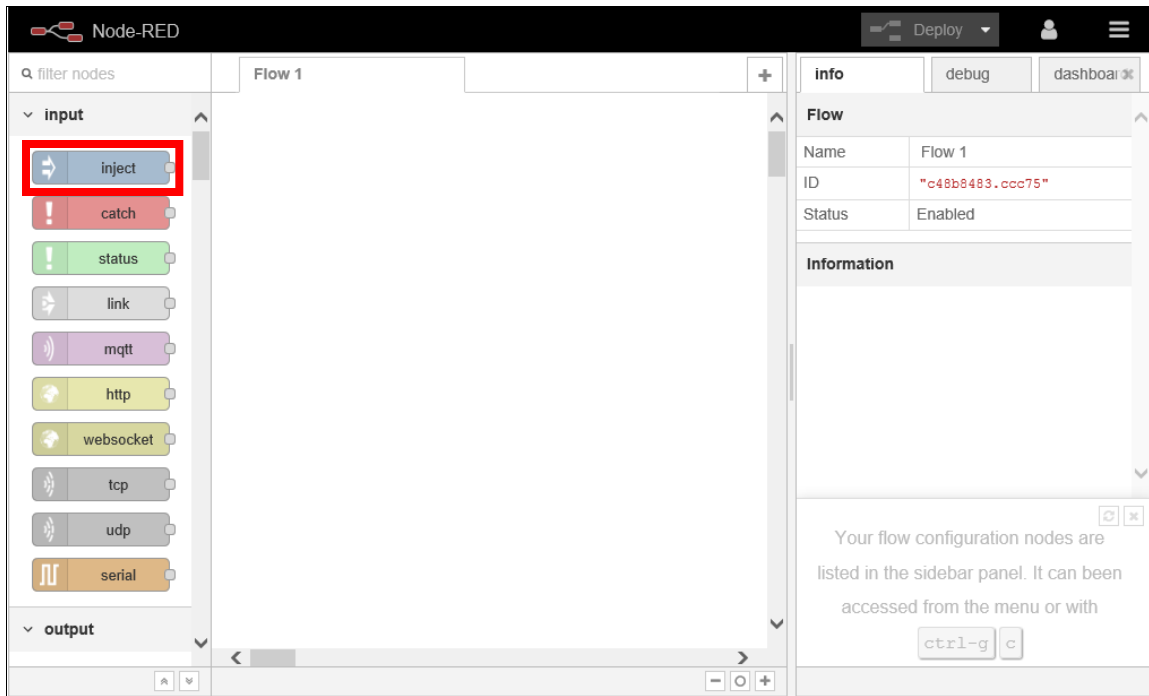
Nodes used in the examples:

- inject
- http request

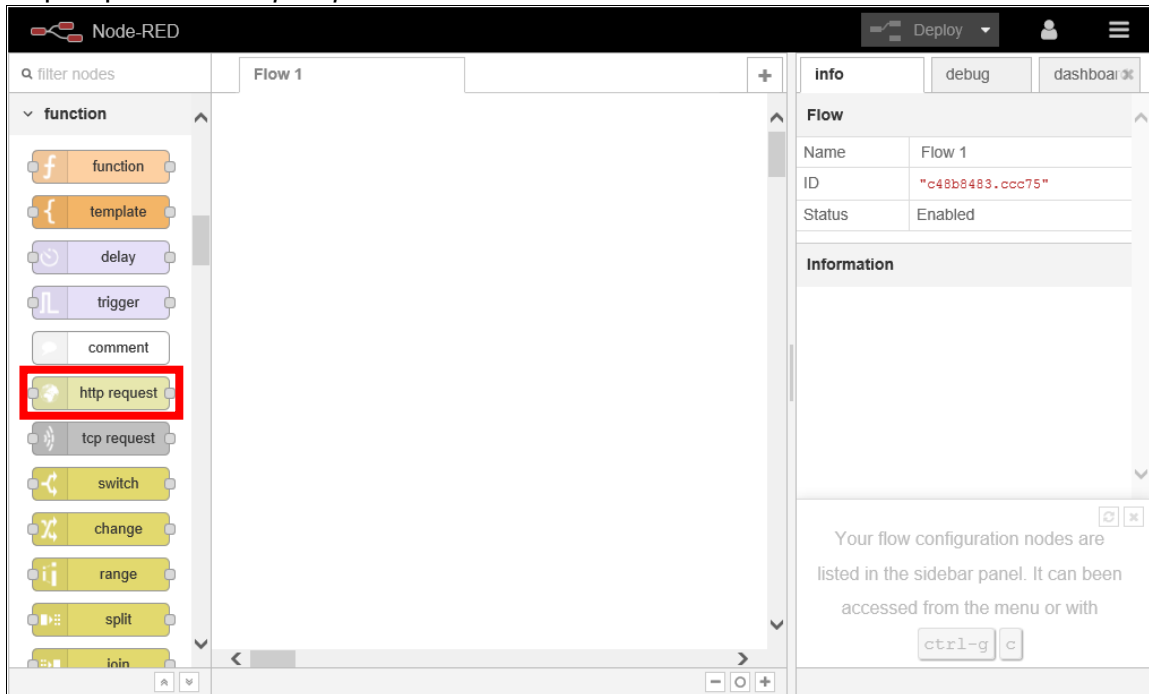
Advantech ADAM nodes used in the examples

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

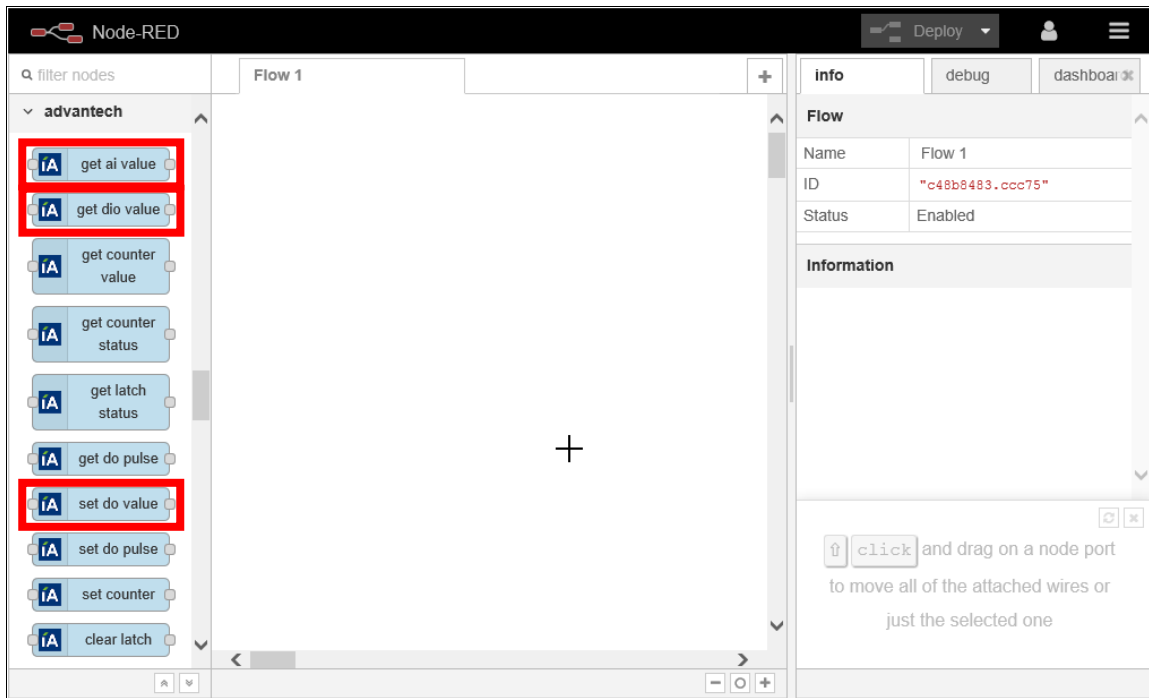
#### 4.1.1. Inject: The *inject* node can be found in the *input* folder



#### 4.1.2. Http request: The *http request* node can be found in the *function* folder



4.1.3. Set/get nodes: The *get ai value*, *get dio value*, *set do value* nodes can be found in the *Advantech* folder

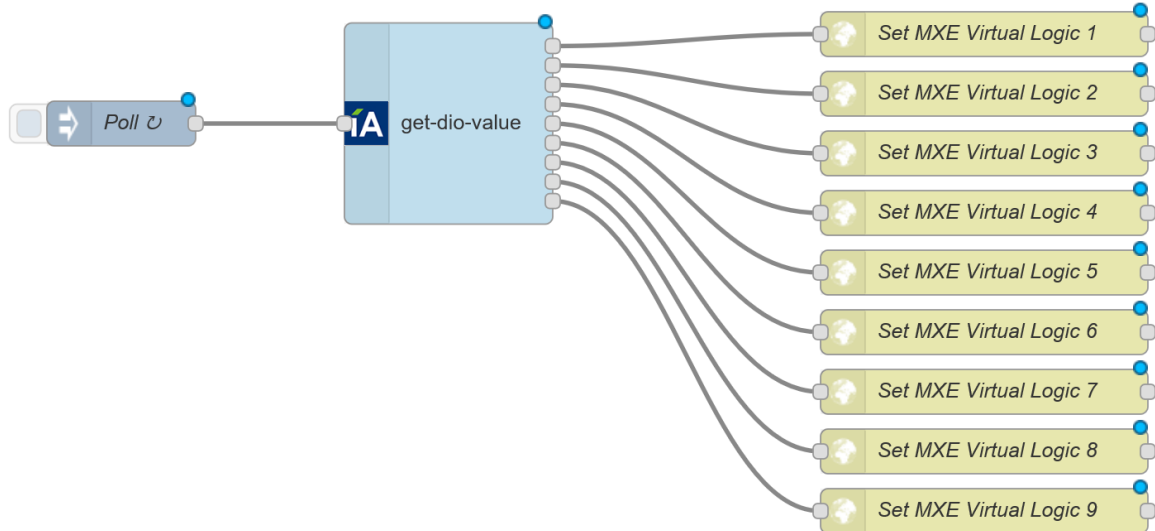


## 4.2. Generic examples for Node-RED running on the ADAM-6750

In the following three examples an Inject node is configured to periodically (every second) trigger set/get actions. This is generally known as “polling”, so the node is labeled as *Poll*. See detailed example for further details.

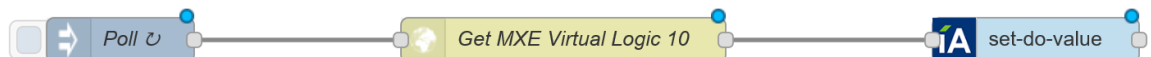
### 4.2.1. **Example 1:** Get ADAM-6750’s digital input/output status (“On”, “Off”), and set MXE Virtual Logic values (“0”, “1”)

The first five *get-dio-value* outputs are the status of the ADAM-6750’s five digital inputs. The next four *get-dio-value* outputs are the status of the ADAM-6750’s four digital outputs.



### 4.2.2. **Example 2:** Get MXE Virtual Logic status (“0”, “1”) and set ADAM-6750’s digital output (“On”, “Off”)

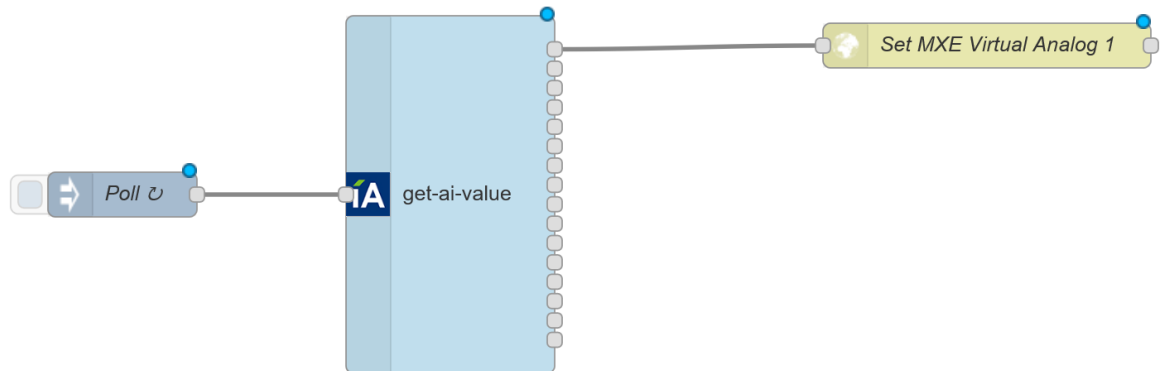
The *set-do-value* block can be configured to set one of the ADAM-6750’s four digital outputs.



### 4.2.3. **Example 3:** Get ADAM-6750’s analog input value (“0...1”), and set MXE Virtual Analog value (“0...1”)

The first eight *get-dio-value* outputs are the value of the ADAM-6750's eight analog inputs.

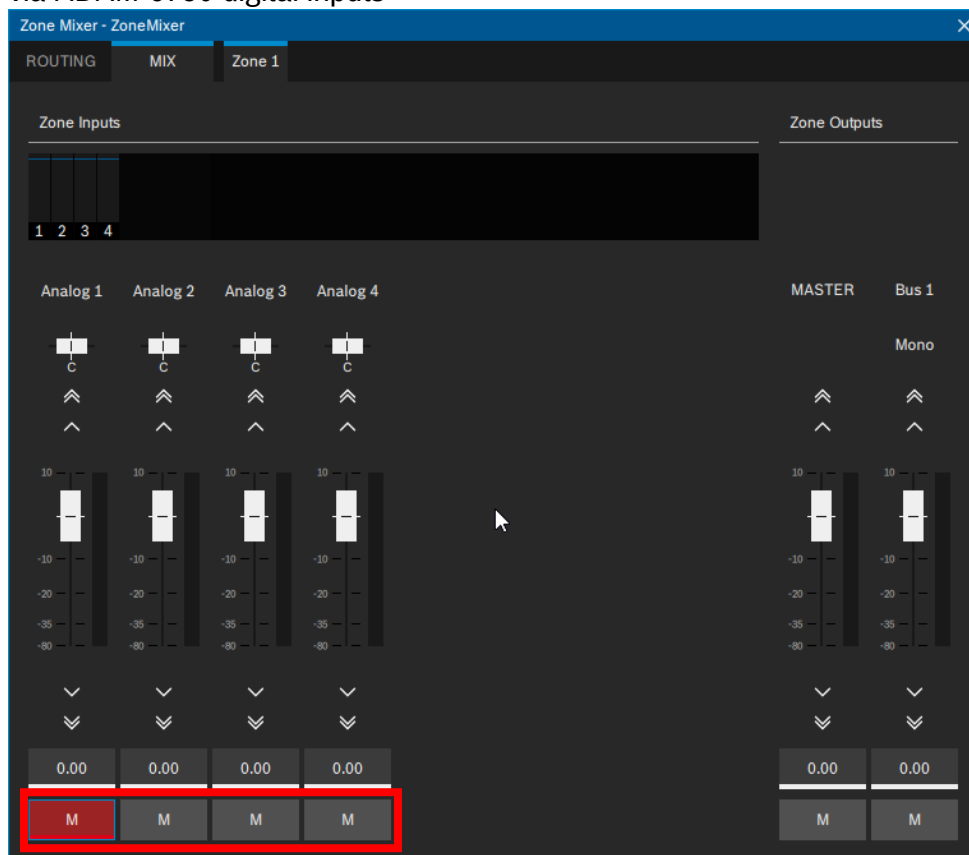
The next eight *get-dio-value* outputs are the status of these eight analog inputs.



### 4.3. Detailed example: MXE 4 channel mute via ADAM-6750 digital inputs

The following example shows how to configure an ADAM-6750 and an MXE5 to set MXE5 Zone Mixer Mutes via the ADAM's digital inputs

- 4.3.1. MXE Zone Mixer configuration (covered in detail in other SONICUE training such as YouTube videos) with *M* mute buttons for channel *Analog 1-4* of *Zone 1* to be controlled via ADAM-6750 digital inputs



- 4.3.2. MXE Logic configuration (covered in detail in other SONICUE training such as YouTube videos)

Each of the *Logic Tasks* “looks” at a different *Virtual Logic* value (1-4) and controls then one of the Zone Mixer’s *Zone 1* channel *Input 1-4* mutes

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"

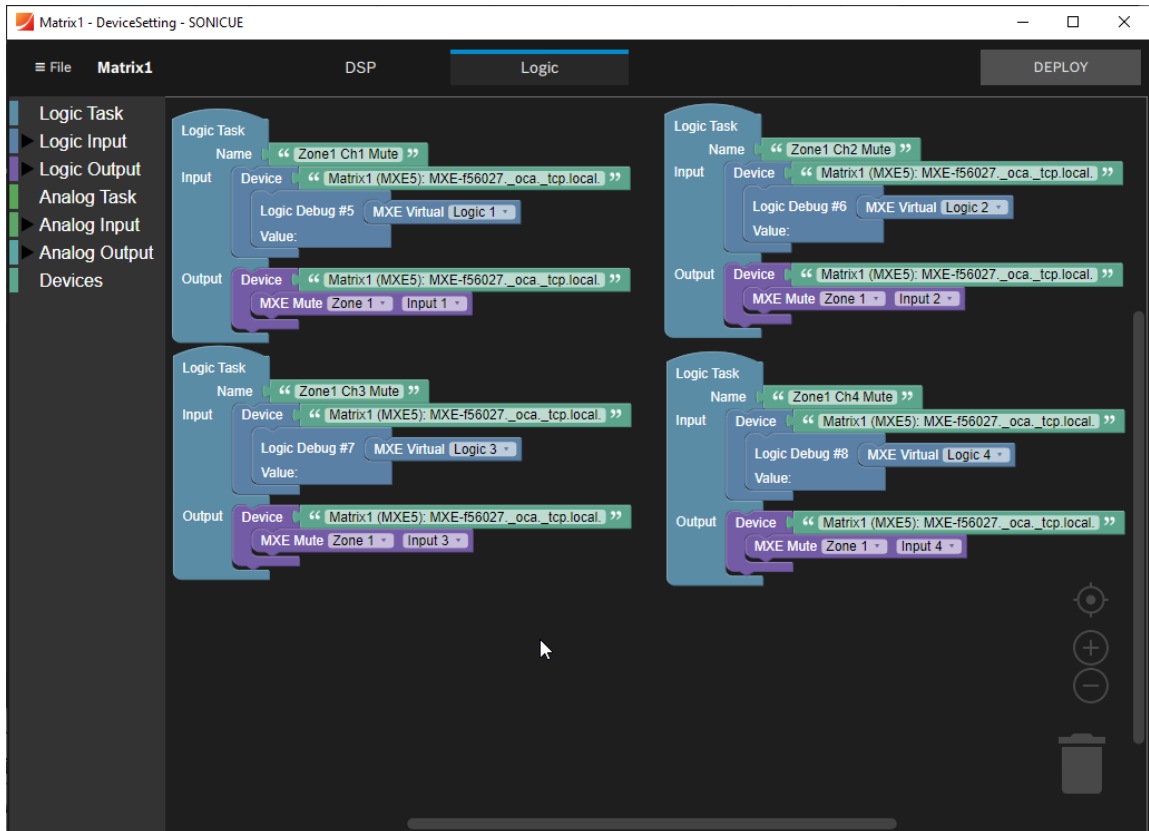
Input: Device "Matrix1 (MXE5): MXE-f56027\_oca\_tcp.local"

Logic Debug #8: MXE Virtual Logic 4

Value:

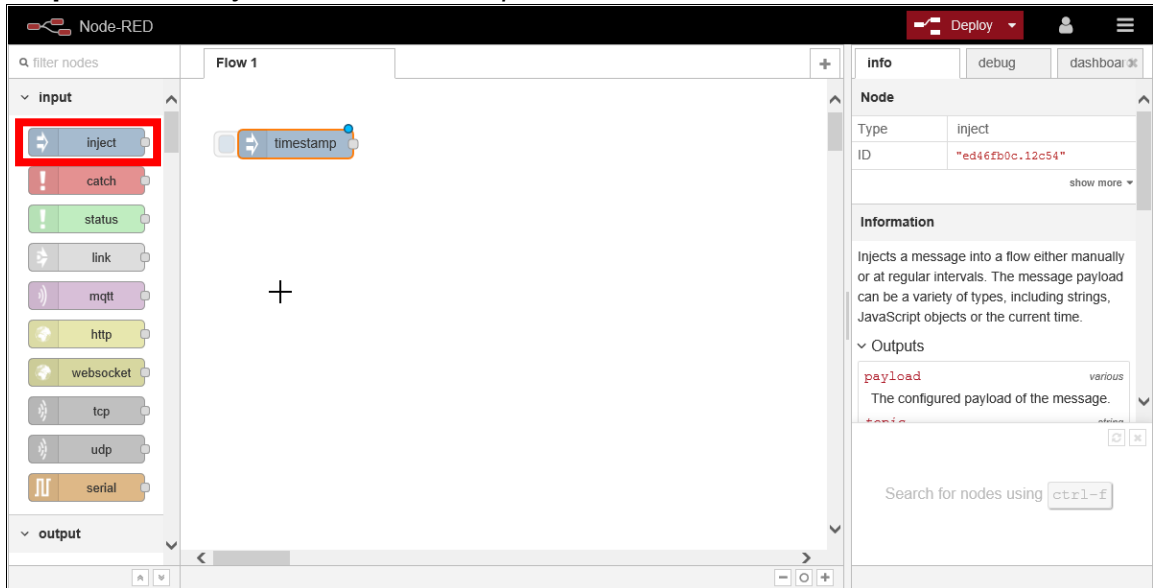
Output: Device "Matrix1 (MXE5): MXE-f56027\_oca\_tcp.local"

MXE Mute Zone 1 Input 4



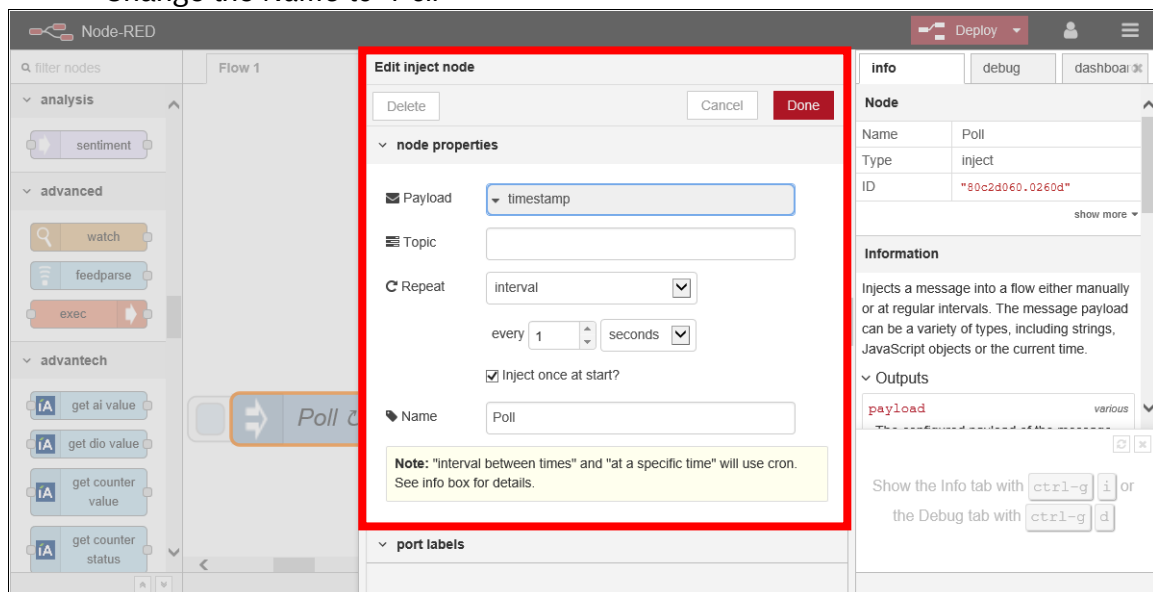
### 4.3.3. Node-RED configuration (covered in detail step-by-step)

#### Step 1: Add an *inject* node from the *input* folder



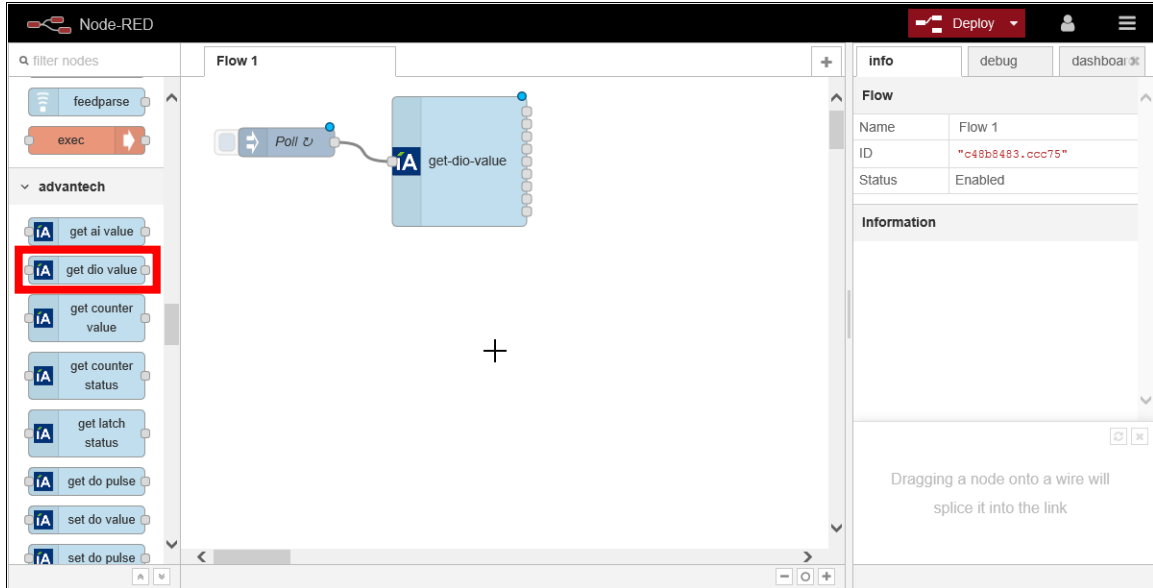
#### Step 2: Double click the inject node labeled as *timestamp* to open the *Edit inject node* window and make the following changes:

- Set *Repeat* to *interval*, every 1 seconds
- Check the *Inject once at start* checkbox
- Change the *Name* to “Poll”



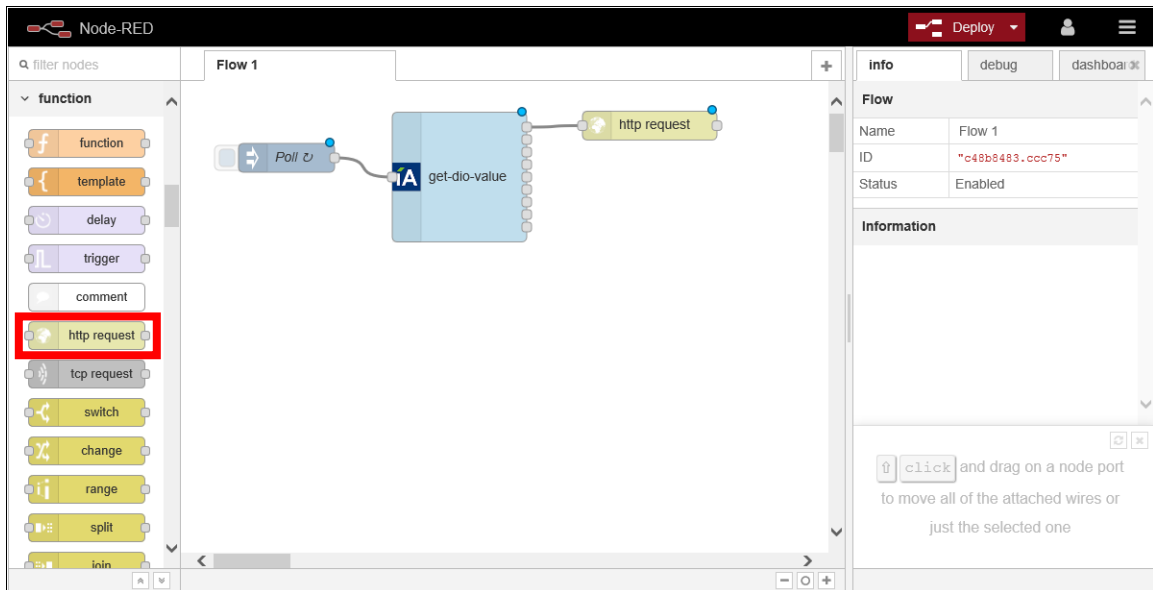


**Step 3:** Add a *get-dio-value* node from the *advantech* folder and connect it to the inject node labeled *Poll* as shown in the screenshot below



The screenshot shows the Node-RED interface. On the left, the 'advantech' folder is expanded, and the 'get-dio-value' node is highlighted with a red box. In the center workspace, a 'Poll' inject node is connected to the 'get-dio-value' node. The right sidebar shows the 'info' tab for 'Flow 1', with fields for Name, ID, and Status.

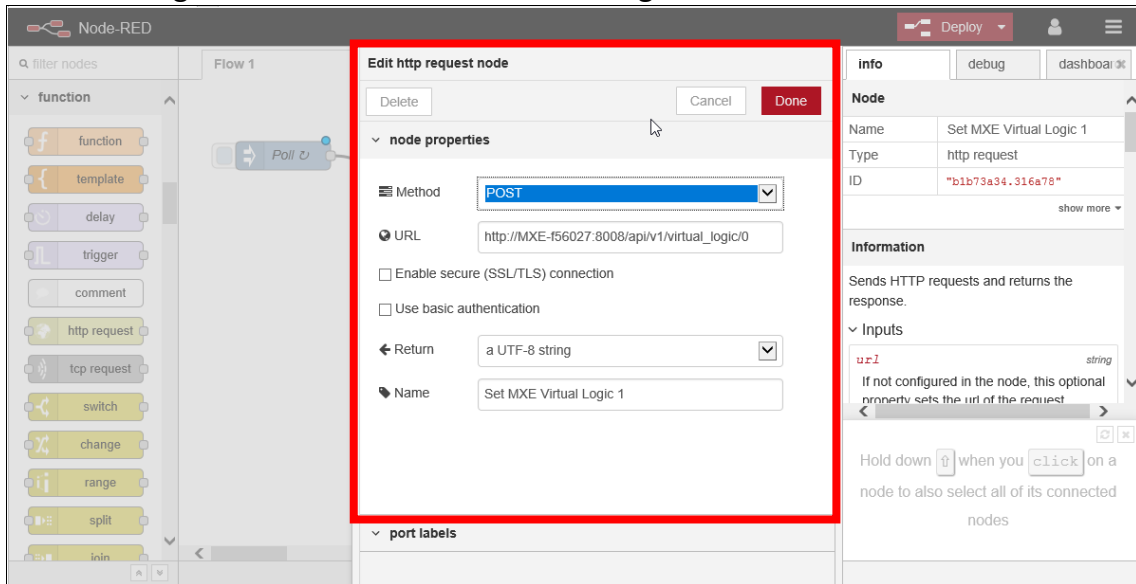
**Step 4:** Add a *http request* node from the *function* folder and connect it to the *get-dio-value* node as shown in the screenshot below



The screenshot shows the Node-RED interface. On the left, the 'function' folder is expanded, and the 'http request' node is highlighted with a red box. In the center workspace, the 'http request' node is connected to the 'get-dio-value' node. The right sidebar shows the 'info' tab for 'Flow 1', with fields for Name, ID, and Status.

**Step 5:** Double click the *http request* node to open the *Edit http request node* window and make the following changes (the example MXE is named “MXE-f56027”):

- Set *Method* to *POST*
- As *URL* enter “*http://MXE-f56027:8008/api/v1/virtual\_logic/0*”
- Change the *Name* to “*Set MXE Virtual Logic 1*”



**Step 6:** Copy the *http request* node labeled as *Set MXE Virtual Logic 1* and paste it three times.

Make the following changes to the first node that has been pasted (remember: the example MXE is named “MXE-f56027” -> you need to use the name of “your” MXE):

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

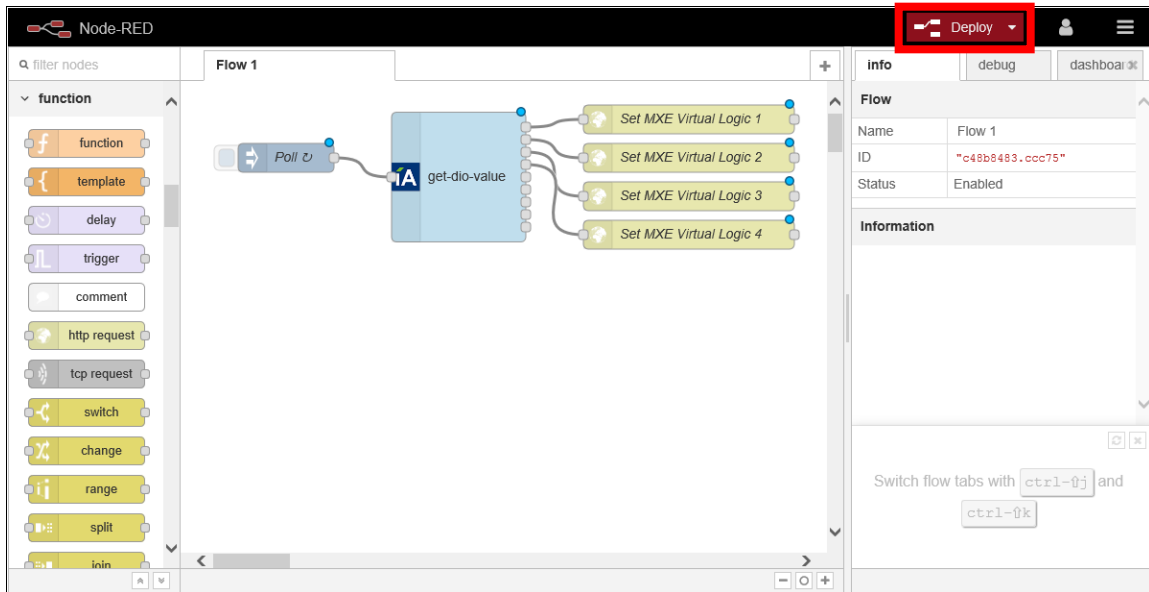
Make the following changes to the second node that has been pasted:

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

Make the following changes to the third node that has been pasted:

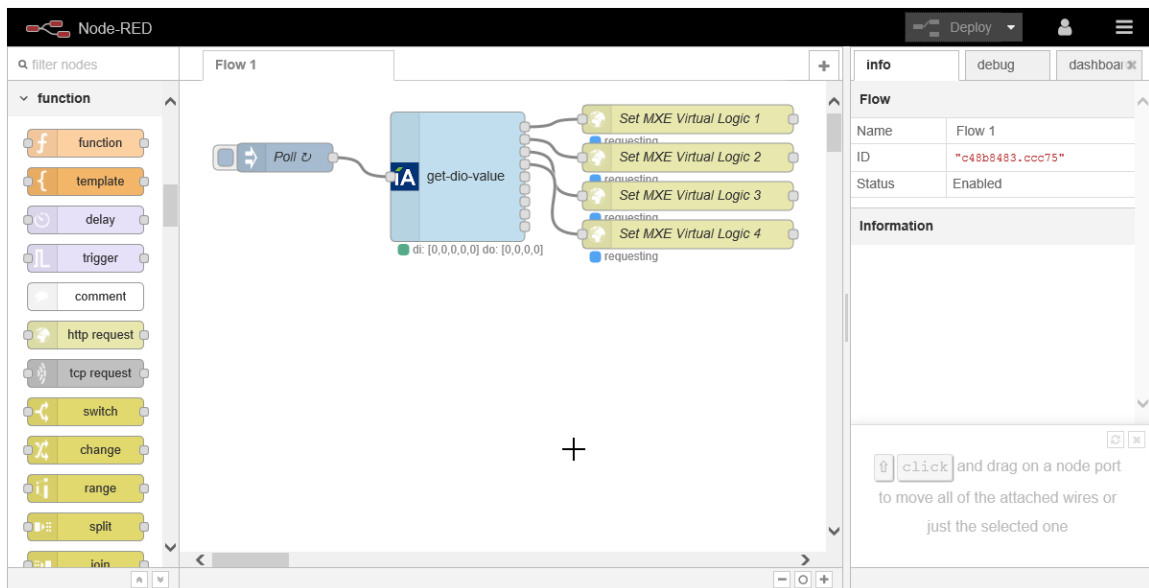
- Change *URL* to “*http://MXE-f56027:8008/api/v1/virtual\_logic/3*”
- Change the *Name* to “*Set MXE Virtual Logic 4*”

When you are finished the flow *Flow 1* should look as follows and you can finally click the **Deploy** button



The flow gets alive:

- The status of the ADAM-6750's digital inputs and outputs is displayed below the *get-dio-value* node and the http request nodes start *requesting*



**DONE!**