IOT : UTILISATION DE NODE RED (APPLICATION : SUPERVISION DE MODULE ADAM ADVANTECH)[1]

Objectifs : Être capable de générer rapidement une IHM pour afficher/contrôler un système connecté.
Etre capable d'interfacer des modules ADAM ou des IoT.

1 By SB ver20191110

1. INTRODUCTION

Node-RED a été crée par IBM en 2013 afin de facilité la création d'objets connectés.
C'est un langage graphique (sous jacent en javascript) qui permet de créer des liens entre les flux de données entrant et sortant.
Il est gratuit et nécessite : NodeJS (javascript du coté serveur)

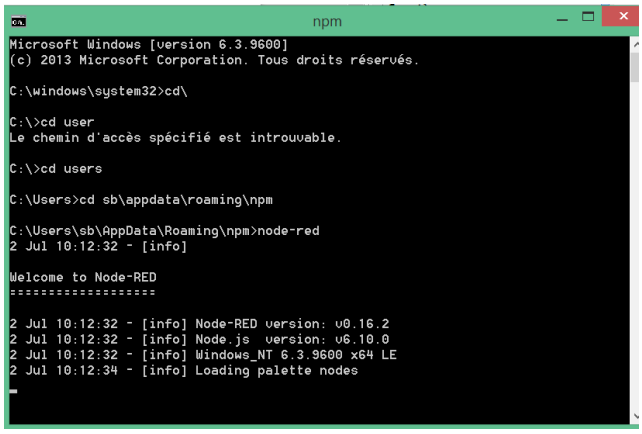
2. INSTALLATION DE NODE RED

2.1. Installer sous windows

Il faut installer : Node JS puis Node-Red
Installer une version LTS de NodeJS.
Installer Node-red en utilisant la ligne de commande sous windows dans une console en mode administrateur :
"npm install node-red" (à vérifier)

2.2. Lancement sous Windows

Lancer la console windows en mode administrateur
Aller dans le dossier : c:\users\sb\appdata\roaming\npm
Lancer la commande node-red
Autoriser l'accès



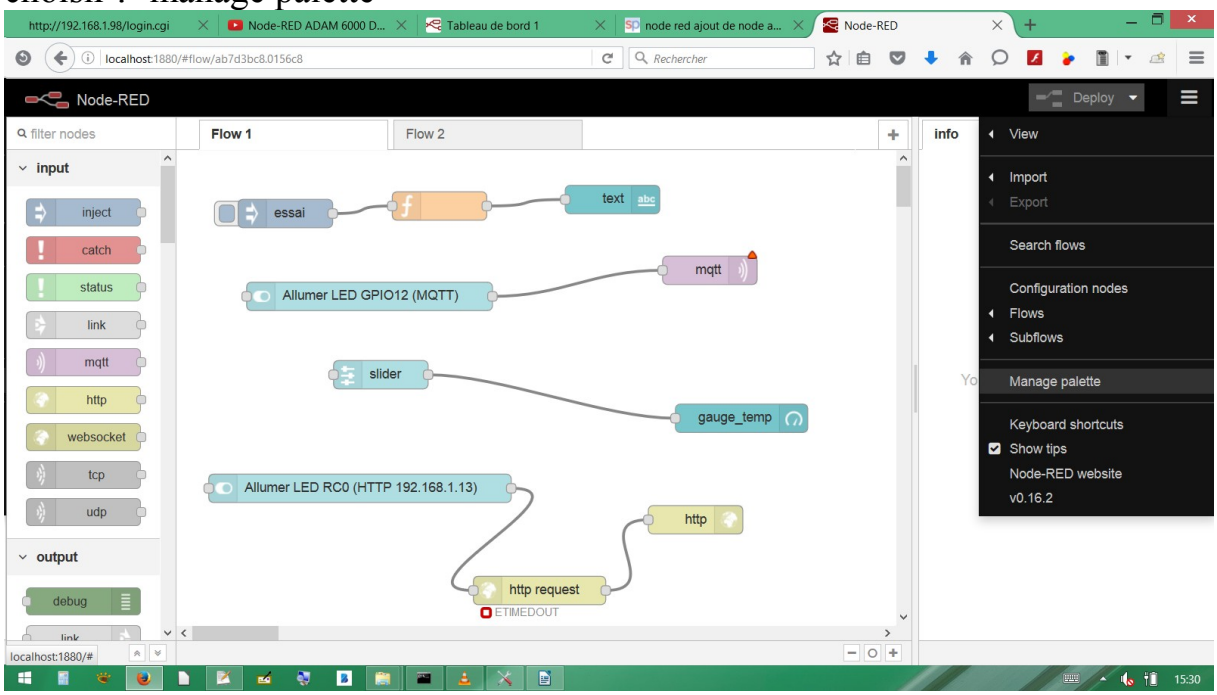
Allez avec un navigateur à l'ip :http://localhost:1880 : l'interface de node-red fonctionne....

IP du dashboard pour test :

http://localhost:1880/ui/#

2.3. Installation de nouveaux NODE

Pour installer de nouveaux "node" dans la palette, utiliser le menu de droit et choisir : "manage palette"



Dans "manage palette", aller dans l'onglet "install" et lancer une recherche : exemple : serial (pour bluetooth) ou modbustcp (pour adam)

Installer les nodes qui vous intéressent.
Les nodes apparaissent alors dans la palette.

3. EXEMPLE UTILISANT UN MODULE ADAM 6060

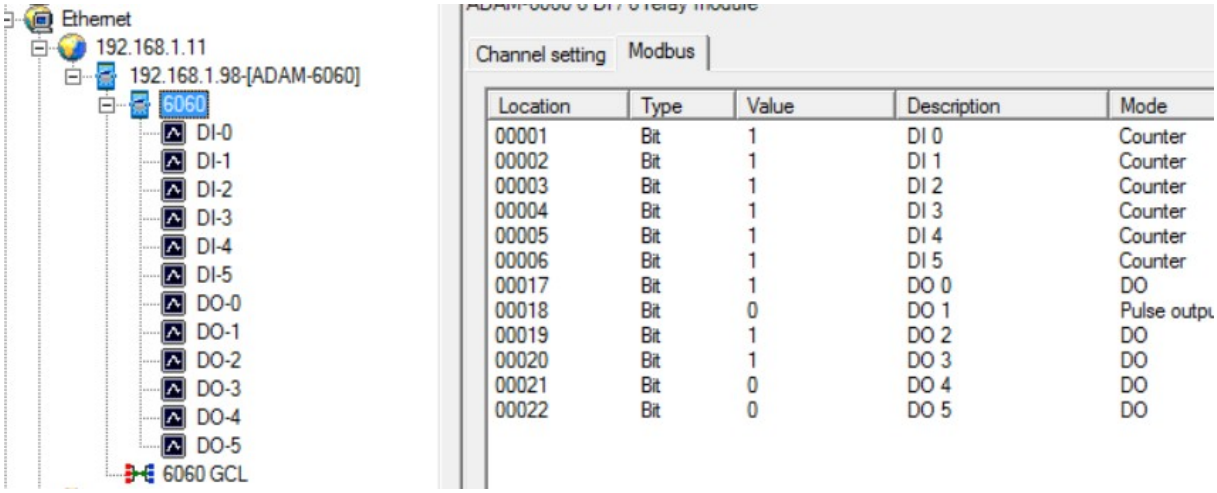
3.1. Cahier des charges

Créer une interface graphique (IHM) pour afficher les données transmises par les modules ADAM par modbus.
Matériel :
ADAM6060, PC avec Node Red, Tablette ou smartphone

3.2. Configuration du module à l'aide de ADAM/APAX .NET de Advantech

Le module ADAM est branché sur le réseau.
L'appli ADAM/APAX permet de connaître son IP par une recherche (loupe).
Notez l'IP : 192.168.1.98

Aller dans l'onglet "Modbus" :



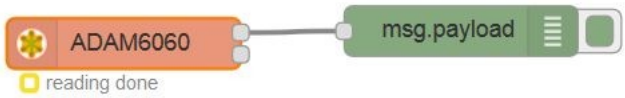
Les entrées/sorties sont numérotées :
de 1 à 6 pour les DI (data in)
de 17 à 22 pour les DO (data out)

ATTENTION : sous NodeRed les numérotations sont à décaler de -1 car elles commencent à 0.

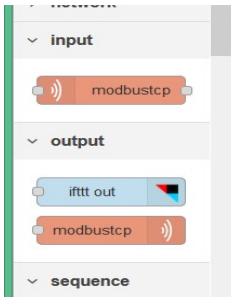
3.3. Création de flow sous NODE RED

3.3.1 Réception des données en ModBus :

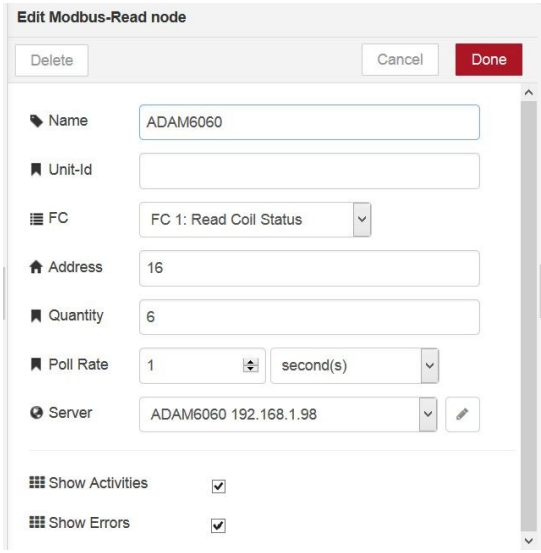
On veut créer le flow suivant :

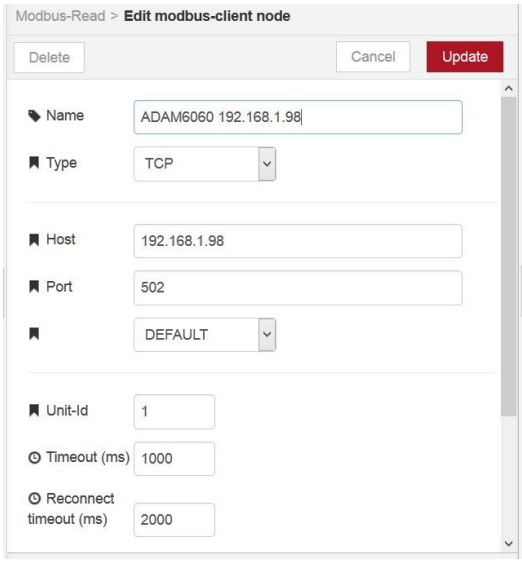


Le node ADAM6060 est un node MODBUS READ :



Le node ADAM est configuré comme suit :



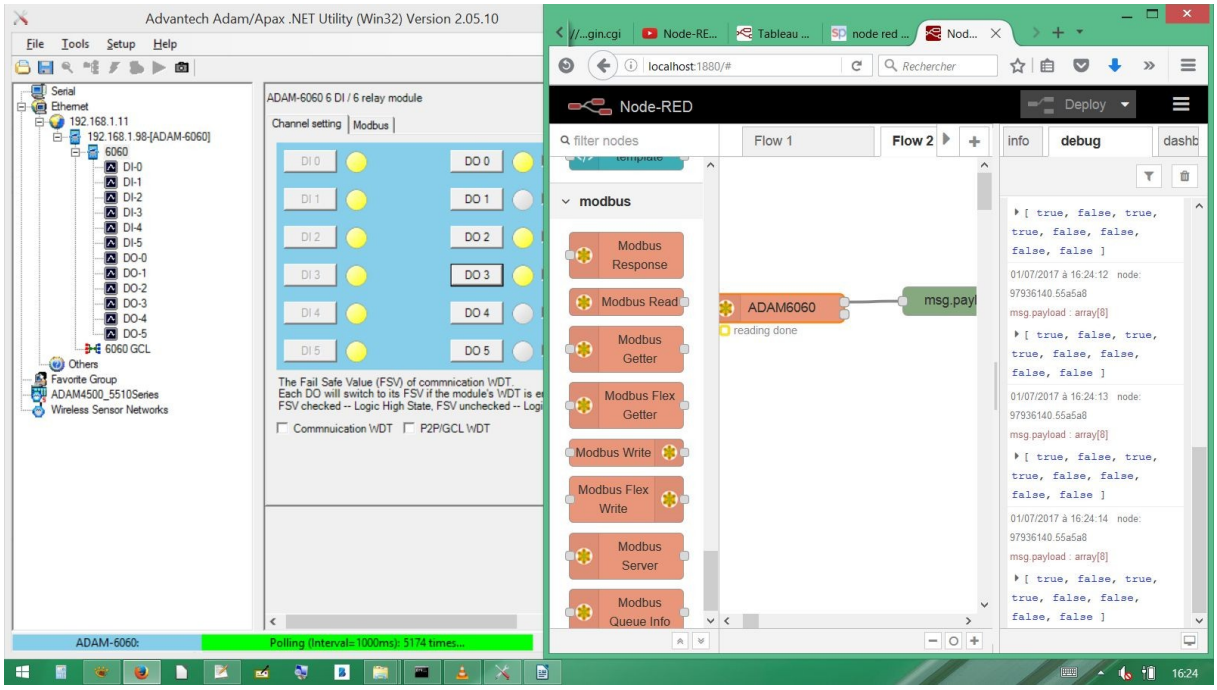


Modbus-Read node :
Name : nom à choisir indifféremment
Unit-id : rien
FC : choisir entre les 4 possibilités
Adresse : première adresse à lire : ICI on lit DO17 donc on met 16 (car numérotation à partir de 0 sur NodeRed
Quantity : nombre d'adresse consécutives à lire (ici DO17 à DO22 = 6)
Poil rate : temps de rafraîchissement : 1s
Server : faire édit puis configurer le client comme suit ...

Modbus-Client node :
Name : nom à choisir indifféremment
Type : choisir TCP
Host : correspond à l'adresse du ADAM.
Port : correspond au port sur lequel émet l'ADAM en modbus.

3.3.2 Test de la connexion

On place un node "debug" afin de visualiser dans la partie droite "debug" ce qui arrive en sortie du nœud.



On fait : "deploy" afin de déployer le flow et on va à l'adresse : localhost:1880/ui pour visualiser.
On voit sur la droite la valeur des DO controlée par l'appli ADAM/APAX.

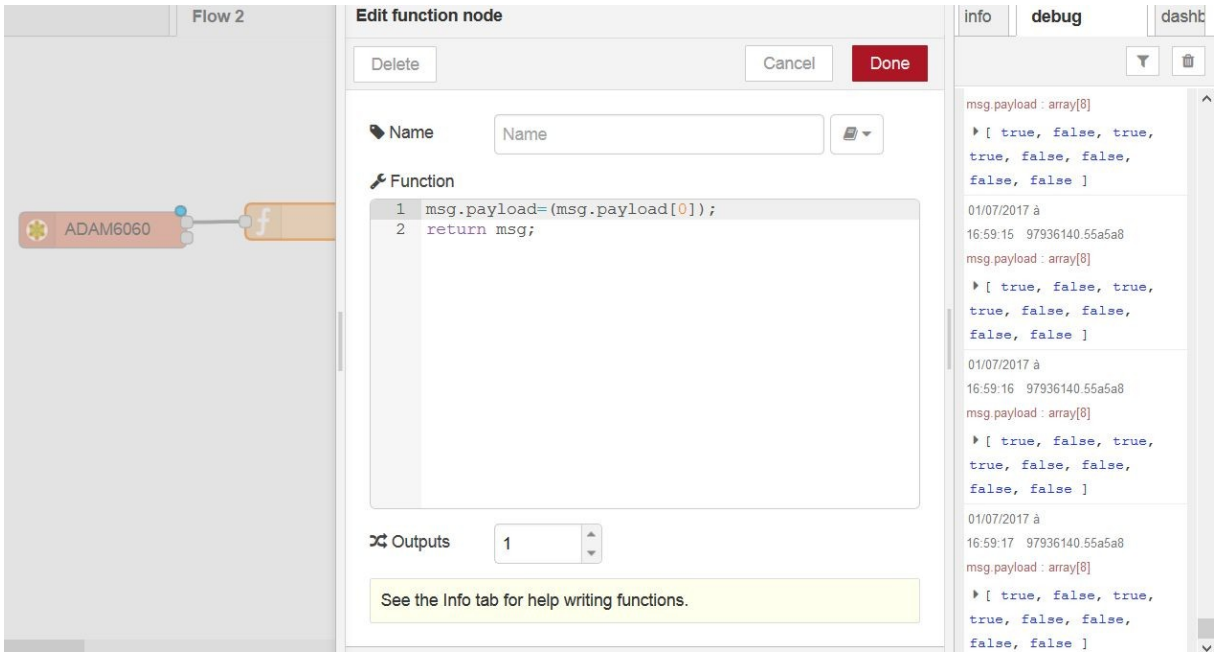
Une fois ce test ok on peut continuer sinon trouver l'erreur :) .

3.4. Création d'un dashboard (Tableau de bord - IHM)

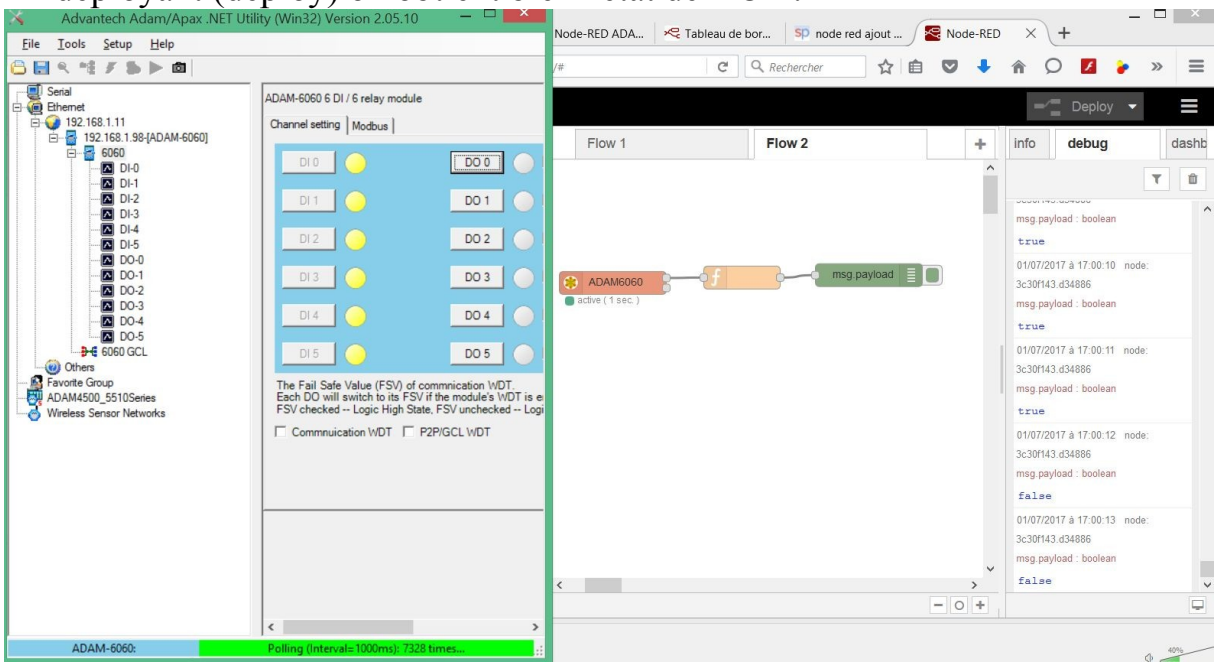
3.4.1 Extraction d'une seule donnée.

On utilise une fonction :
on extrait du message utile (msg.payload[0]) la première donnée.

Explication de la fonction :
msg.payload est l'argument d'entrée : il contient 6 valeurs boolean (voir debug)
msg.payload est l'argument de sortie de la fonction : on lui injecte (msg.payload[0]) afin de ne sortir que le premier élément du message d'entrée.

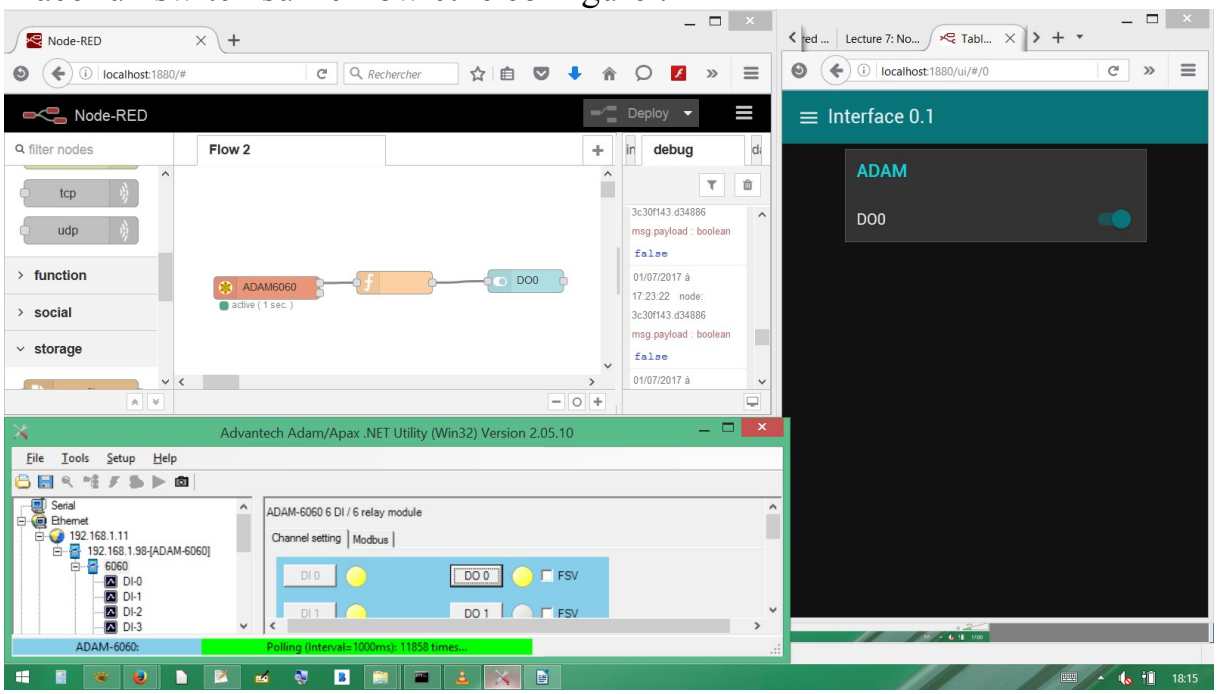


En déployant (deploy) on obtient bien l'état de DO1 :



3.4.2 Affichage sur un "switch" dans le Dashboard

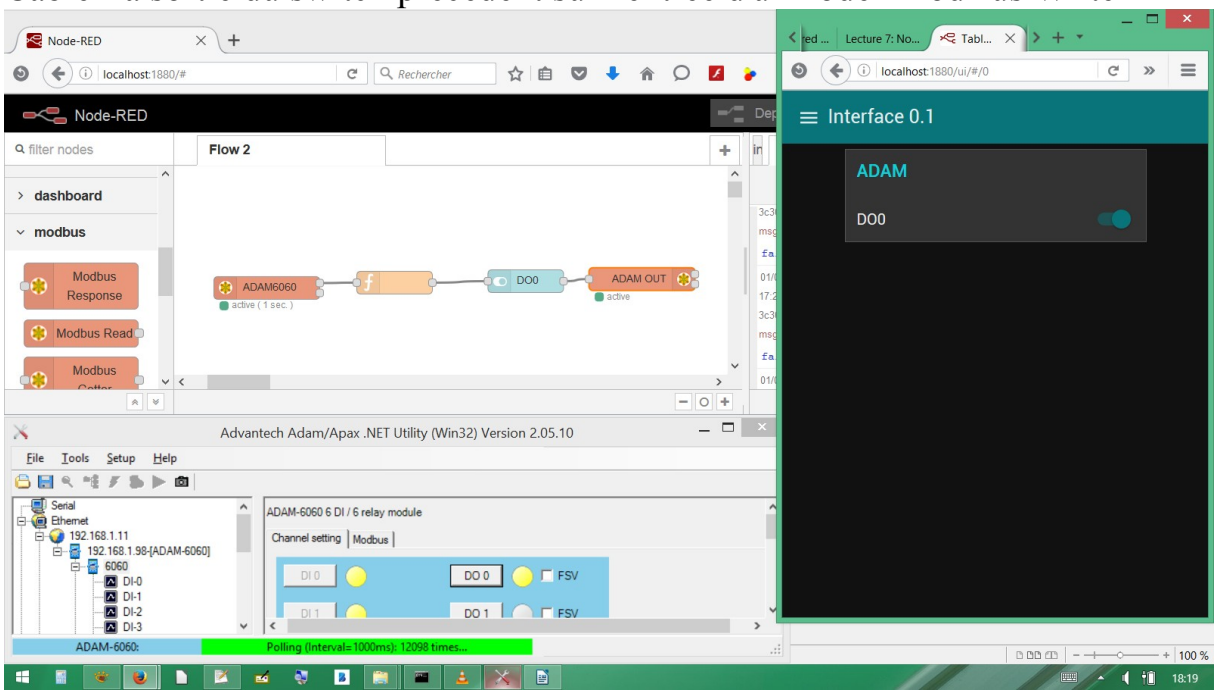
Placer un switch sur le flow et le configurer.



Le switch recopie l'état de DO1 du ADAM6060.

3.4.3 Affichage et contrôle par un "switch"

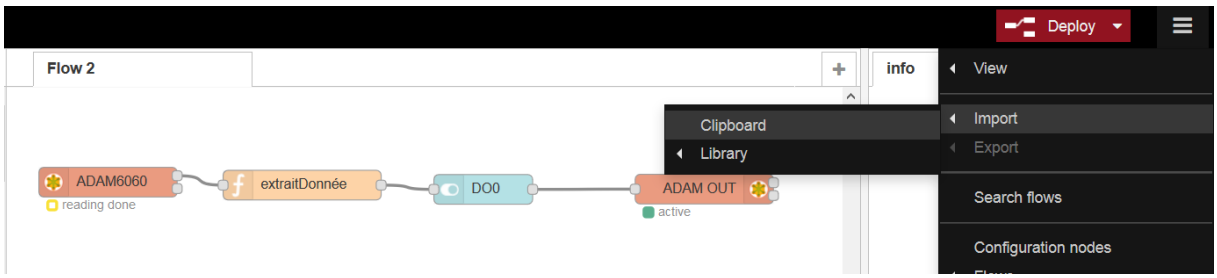
Câbler la sortie du switch précédent sur l'entrée d'un node "ModBus Write"



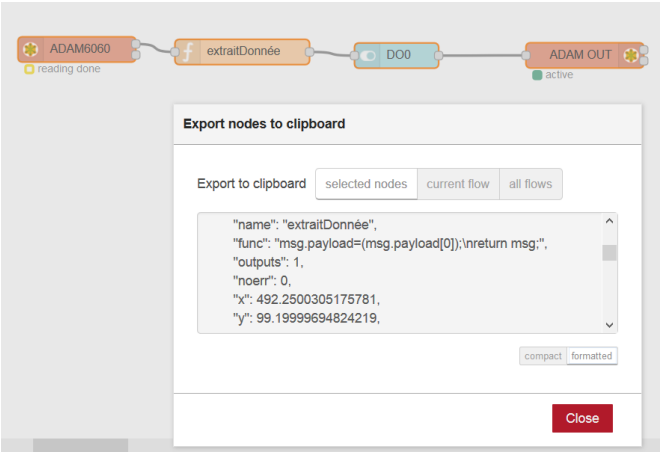
Le switch recopie l'état de DO1 ET peut aussi le contrôler. L'utilisation de l'appli devient inutile.

3.5. Sauvegarde des flows.

Sélectionner le flow à la souris.
Aller dans le menu + export + clipboard



Sélectionner le mode de sauvegarde : compacte ou formatted



Faire "ctrl+C" au clavier
Ouvrir "Notepad++" et coller le clipboard.
Enregistrer le fichier texte ainsi crée.

Les sauvegarde sont des descriptions en JSON.

```
[
  {
    "id": "9d8d75ff.0746e",
    "type": "tab",
    "label": "Flow 2"
  },
  {
    "id": "7959f1fb.ec3d7",
    "type": "modbus-read",
    "z": "9d8d75ff.0746e",
    "name": "ADAM6060",
    "showStatusActivities": true,
    "showErrors": false,
    "unitid": "",
    "dataType": "Coil",
    "adr": "16",
    "quantity": "6",
    "rate": "1",
    "rateUnit": "s",
    "server": "55a5591c.d1174",
    "x": 295.7499694824219,
    "y": 96.20001220703125,
    "wires": [
      [
        "45d2e85b.b58bf"
      ],
      []
    ]
  },
  {
    "id": "45d2e85b.b58bf",
    "type": "function",
    "z": "9d8d75ff.0746e",
    "name": "extraitDonnée",
    "func": "msg.payload=(msg.payload[0]);\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 492.2500305175781,
    "y": 99.19999694824219,
    "wires": [
      [
        "59dc6558.aa3c44"
      ]
    ]
  },
  {
    "id": "59dc6558.aa3c44",
    "type": "ui_switch",
    "z": "9d8d75ff.0746e",
    "name": "",
    "label": "DO0",
    "group": "1b1ef75e.d35891",
    "order": 0,
    "width": 0,
    "height": 0,
    "passthru": false,
    "topic": "",
    "style": "",
    "onvalue": "true",
    "onvalueType": "bool",
    "onicon": "",
    "oncolor": "",
    "offvalue": "false",
    "offvalueType": "bool",
    "officon": "",
    "offcolor": "",
    "x": 675.625,
    "y": 103.5,
    "wires": [
      [
        "c45bea63.c33528"
      ]
    ]
  },
  {
    "id": "c45bea63.c33528",
    "type": "modbus-write",
    "z": "9d8d75ff.0746e",
    "name": "ADAM OUT",
    "showStatusActivities": false,
    "showErrors": false,
    "unitid": "",
    "dataType": "Coil",
    "adr": "16",
    "quantity": "1",
    "server": "55a5591c.d1174",
    "x": 899.375,
    "y": 103.24998474121094,
    "wires": [
      [],
      []
    ]
  },
  {
    "id": "55a5591c.d1174",
    "type": "modbus-client",
    "z": "",
    "name": "ADAM6060 192.168.1.98",
    "clienttype": "tcp",
    "bufferCommands": true,
    "stateLogEnabled": false,
    "tcpHost": "192.168.1.98",
    "tcpPort": "502",
    "tcpType": "DEFAULT",
    "serialPort": "/dev/ttyUSB",
    "serialType": "RTU-BUFFERD",
    "serialBaudrate": "9600",
    "serialDatabits": "8",
    "serialStopbits": "1",
    "serialParity": "none",
    "serialConnectionDelay": "100",
    "unit_id": "1",
    "commandDelay": "1",
    "clientTimeout": "1000",
    "reconnectTimeout": "2000"
  },
  {
    "id": "1b1ef75e.d35891",
    "type": "ui_group",
    "z": "",
    "name": "ADAM",
    "tab": "61c0211a.ac1e6",
    "order": 1,
    "disp": true,
    "width": "6"
  },
  {
    "id": "61c0211a.ac1e6",
    "type": "ui_tab",
    "z": "",
    "name": "Interface 0.1",
    "icon": "home",
    "order": 1
  }
]
```

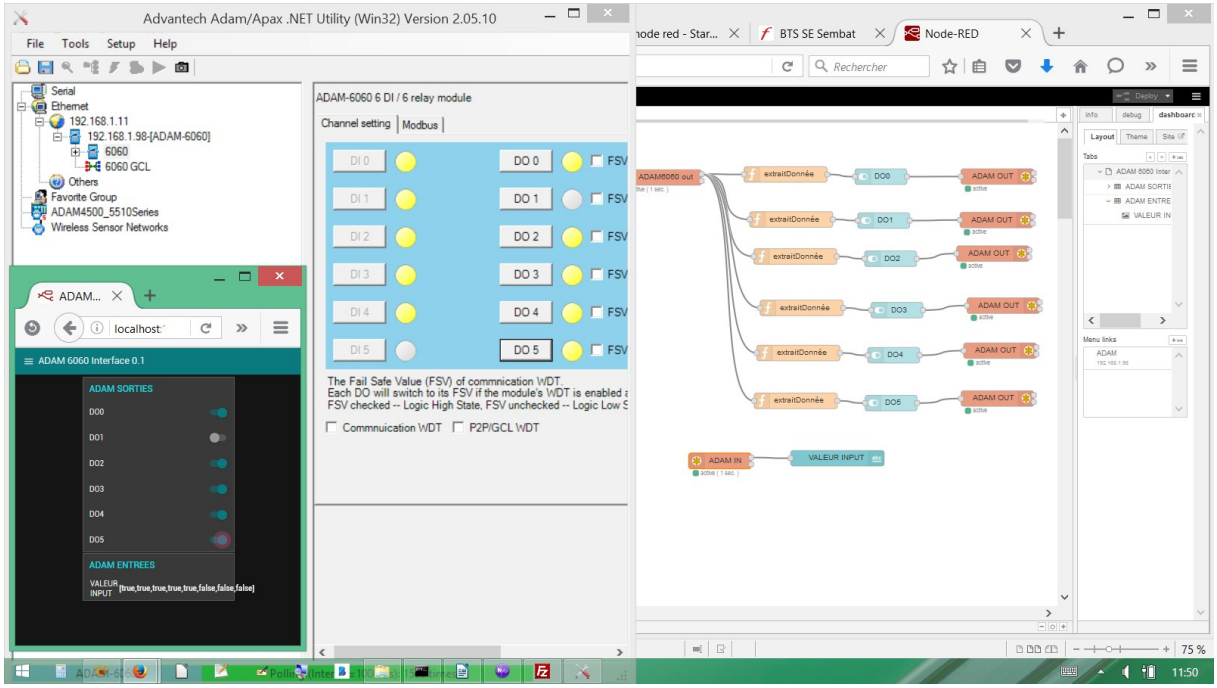
3.6. Accès aux IHM, depuis un smartphone ou une tablette

Avec un navigateur compatible, allez à l'adresse ip du serveur NodeRed (PC ou nanoPC sur lequel vous avez installé le logiciel)
Exemple : http://192.168.1.11:1880/ui

4. EXERCICES

4.1. Exercice 1 : Contrôle d'un module Adam en modbus

Créer une IHM afin de contrôler toutes les sorties et visualiser les entrées du module ADAM6060.

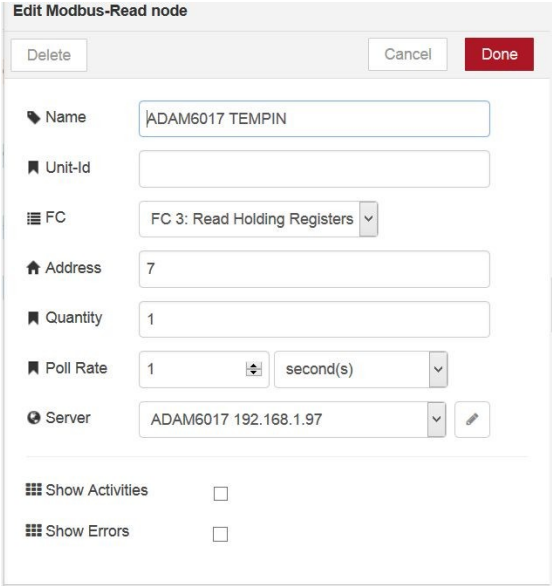


Pour les input on utilise un node "text" qui affiche le msg.payload (message utile).
Rmq : Sur mon ADAM j'ai cabler DO5 sur DI5 afin de pouvoir faire changer facilement une entrée.

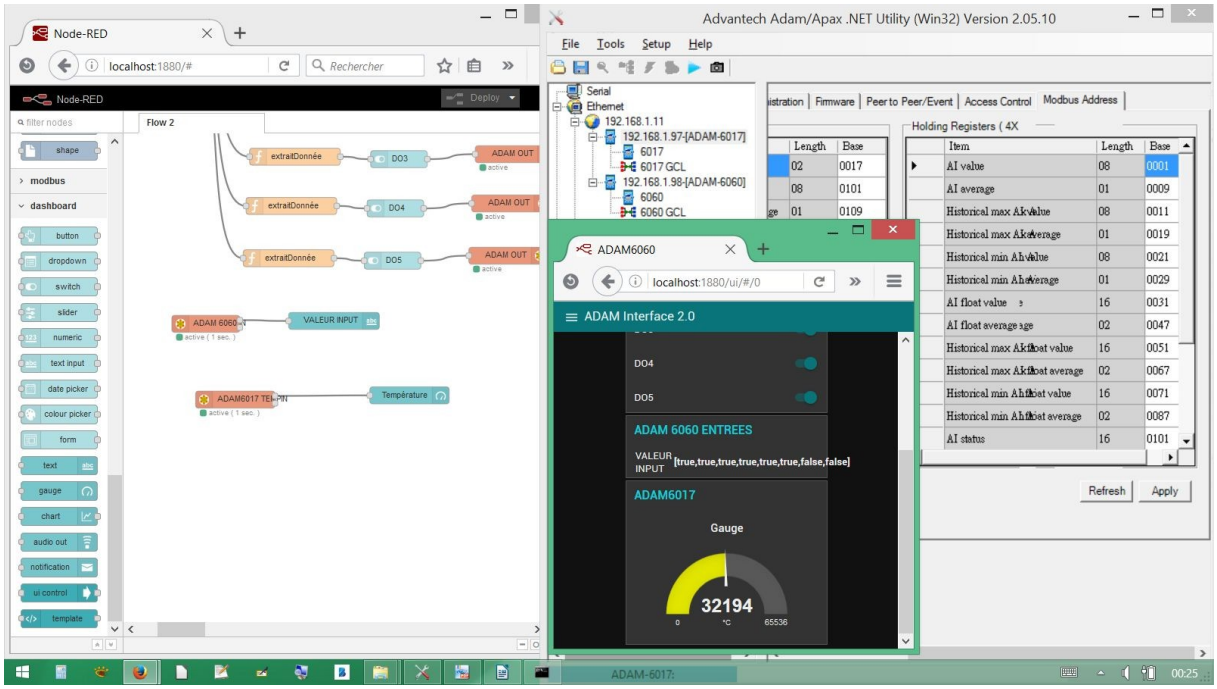
4.2. Exercice 2 : Contrôle de deux modules adam avec une IHM node-red en modbus

On souhaite contrôler un module ADAM6060 (activation d'un relais) et un module 6017 (mesure d'une température avec un lm35).

On souhaite visualiser les Aivalues et surtout le AI7 (adresses modbus 8 sur ADAM donc 7 sur Node-Red). Rmq : on configure Read Holding Registers dans les commandes modbus du node.



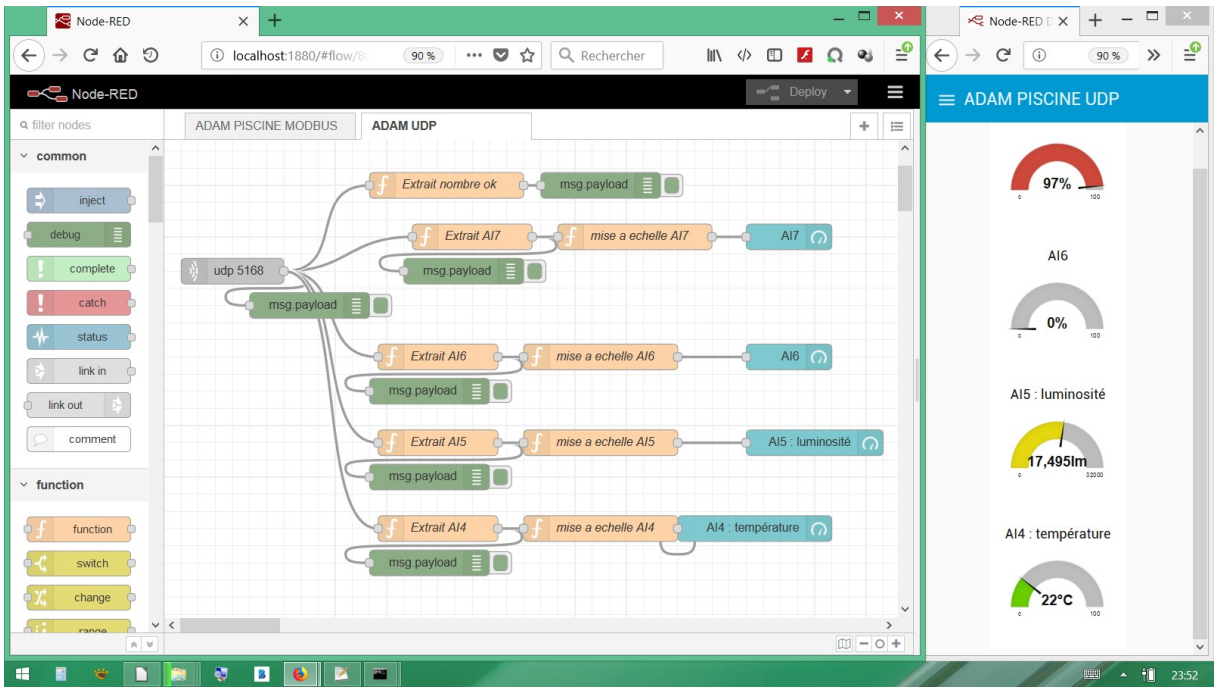
D'ou le résultat :



Reste à faire la mise à l'échelle en fonction de la grandeur mesurée.

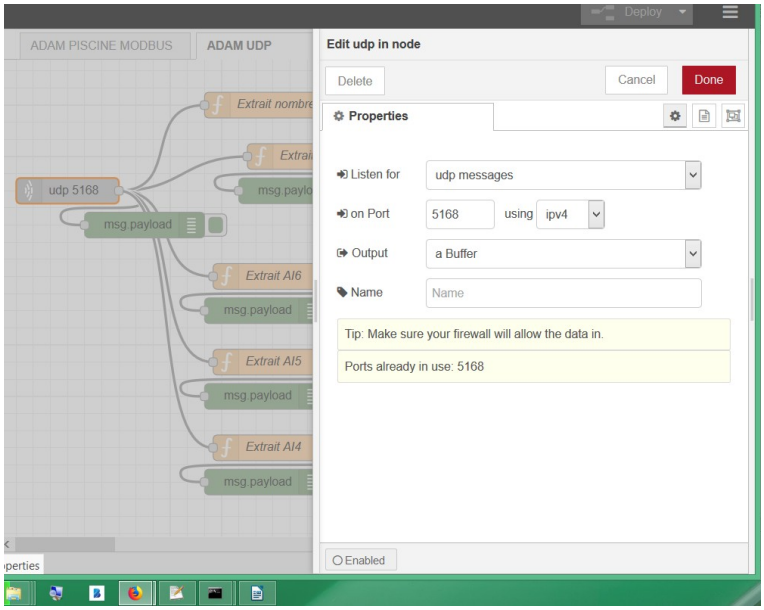
4.3. Exercice 3 : Réception protocole UDP

4.3.1 IHM en node-red



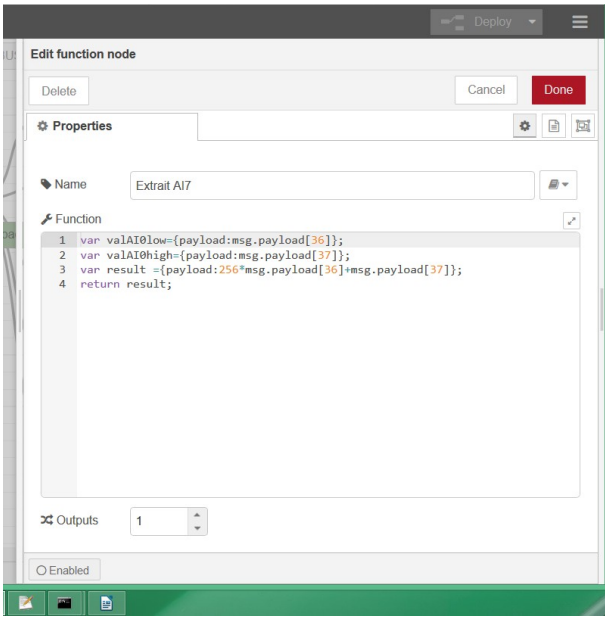
4.3.2 Détails

Node UDP :



Utiliser 'a buffer' en output car un string ne permet pas de récupérer facilement les données non ACSII.

Fonction : extrait AI7



Les cases 36 et 37 correspondent à l'emplacement des données de AI7 dans le flux UDP des ADAM6017 (cf doc. ADAM streaming)

4.3.3 Solution : flow à importer


```
[
  {
    "id": "864171cc.c022b8",
    "type": "tab",
    "label": "ADAM UDP",
    "disabled": false,
    "info": ""
  },
  {
    "id": "bbe33e33.dc1508",
    "type": "debug",
    "z": "864171cc.c022b8",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "x": 170,
    "y": 300,
    "wires": []
  },
  {
    "id": "7e64cfac.9e8fd8",
    "type": "udp in",
    "z": "864171cc.c022b8",
    "name": "",
    "iface": "",
    "port": "5168",
    "ipv": "udp4",
```

```
        "multicast": "false",
        "group": "",
        "datatype": "buffer",
        "x": 80,
        "y": 260,
        "wires": [
          [
            "bbe33e33.dc1508",
            "508e5ac3.474534",
            "597966eb.7f4e2",
            "62bad0c.963b6b",
            "d7c6661d.db3c1",
            "e1eb5e88.5d9178"
          ]
        ]
    },
    {
        "id": "31376131.edfb66",
        "type": "debug",
        "z": "864171cc.c022b8",
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "x": 350,
        "y": 260,
        "wires": []
    },
    {
        "id": "508e5ac3.474534",
        "type": "function",
        "z": "864171cc.c022b8",
        "name": "Extrait AI7",
        "func": "var valAI0low={payload:msg.payload[36]};\nvar valAI0high={payload:msg.payload[37]};\nvar
result ={payload:256*msg.payload[36]+msg.payload[37]};\nreturn result;",
        "outputs": 1,
        "noerr": 0,
        "x": 360,
        "y": 220,
        "wires": [
          [
            "31376131.edfb66",
            "40e19e8d.2a21e8"
          ]
        ]
    },
    {
        "id": "e929247f.ca37f8",
        "type": "ui_gauge",
        "z": "864171cc.c022b8",
        "name": "",
        "group": "1f8fddae.f90832",
        "order": 0,
        "width": 0,
        "height": 0,
        "gtype": "gage",
```

```
    "title": "AI7",
    "label": "",
    "format": "{{value|number:0}}%",
    "min": 0,
    "max": "100",
    "colors": [
      "#00b500",
      "#e6e600",
      "#ca3838"
    ],
    "x": 730,
    "y": 220,
    "wires": []
  },
  {
    "id": "597966eb.7f4e2",
    "type": "function",
    "z": "864171cc.c022b8",
    "name": "Extrait nombre ok",
    "func": "var valAI={payload:msg.payload[22]};\nreturn valAI;",
    "outputs": 1,
    "noerr": 0,
    "x": 330,
    "y": 160,
    "wires": [
      [
        "9cef3f8b.c53e18"
      ]
    ]
  },
  {
    "id": "9cef3f8b.c53e18",
    "type": "debug",
    "z": "864171cc.c022b8",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "x": 510,
    "y": 160,
    "wires": []
  },
  {
    "id": "40e19e8d.2a21e8",
    "type": "function",
    "z": "864171cc.c022b8",
    "name": "mise a echelle AI7",
    "func": "msg.payload=(msg.payload-32767)/327.67;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 550,
    "y": 220,
    "wires": [
      [
        "e929247f.ca37f8"
      ]
    ]
```

```
    },
    {
        "id": "b24befd6.067778",
        "type": "debug",
        "z": "864171cc.c022b8",
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "x": 310,
        "y": 400,
        "wires": []
    },
    {
        "id": "62bad0c.963b6b",
        "type": "function",
        "z": "864171cc.c022b8",
        "name": "Extrait AI6",
        "func": "var valAI0low={payload:msg.payload[34]};\nvar valAI0high={payload:msg.payload[35]};\nvar
result ={payload:256*msg.payload[34]+msg.payload[35]};\nreturn result;",
        "outputs": 1,
        "noerr": 0,
        "x": 320,
        "y": 360,
        "wires": [
            [
                "b24befd6.067778",
                "eb526872.0265f"
            ]
        ]
    },
    {
        "id": "77e1d5a4.e023f4",
        "type": "ui_gauge",
        "z": "864171cc.c022b8",
        "name": "",
        "group": "1f8fddae.f90832",
        "order": 0,
        "width": 0,
        "height": 0,
        "gtype": "gage",
        "title": "AI6",
        "label": "",
        "format": "{{value|number:0}}%",
        "min": 0,
        "max": "100",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "x": 730,
        "y": 360,
        "wires": []
    },
    {
```

```
        "id": "eb526872.0265f",
        "type": "function",
        "z": "864171cc.c022b8",
        "name": "mise a echelle AI6",
        "func": "msg.payload=(msg.payload-32767)/327.67;\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 510,
        "y": 360,
        "wires": [
            [
                "77e1d5a4.e023f4"
            ]
        ]
    },
    {
        "id": "7bea40ad.9479c",
        "type": "debug",
        "z": "864171cc.c022b8",
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "x": 310,
        "y": 500,
        "wires": []
    },
    {
        "id": "d7c6661d.db3c1",
        "type": "function",
        "z": "864171cc.c022b8",
        "name": "Extrait AI5",
        "func": "var valAI0low={payload:msg.payload[32]};\nvar valAI0high={payload:msg.payload[33]};\nvar
result ={payload:256*msg.payload[32]+msg.payload[33]};\nreturn result;",
        "outputs": 1,
        "noerr": 0,
        "x": 320,
        "y": 460,
        "wires": [
            [
                "7bea40ad.9479c",
                "c9f8a599.328648"
            ]
        ]
    },
    {
        "id": "91ed1f26.814ee8",
        "type": "ui_gauge",
        "z": "864171cc.c022b8",
        "name": "",
        "group": "1f8fddae.f90832",
        "order": 0,
        "width": 0,
        "height": 0,
        "gtype": "gage",
        "title": "AI5 : luminosité",
```

```
        "label": "",
        "format": "{{value|number:0}}lm",
        "min": 0,
        "max": "32000",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "x": 760,
        "y": 460,
        "wires": []
    },
    {
        "id": "c9f8a599.328648",
        "type": "function",
        "z": "864171cc.c022b8",
        "name": "mise a echelle AI5",
        "func": "msg.payload=(msg.payload-32767);\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 510,
        "y": 460,
        "wires": [
            [
                "91ed1f26.814ee8"
            ]
        ]
    },
    {
        "id": "dcf9c317.89a58",
        "type": "debug",
        "z": "864171cc.c022b8",
        "name": "",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "x": 310,
        "y": 600,
        "wires": []
    },
    {
        "id": "e1eb5e88.5d9178",
        "type": "function",
        "z": "864171cc.c022b8",
        "name": "Extrait AI4",
        "func": "var valAI0low={payload:msg.payload[30]};\nvar valAI0high={payload:msg.payload[31]};\nvar
result ={payload:256*msg.payload[30]+msg.payload[31]};\nreturn result;",
        "outputs": 1,
        "noerr": 0,
        "x": 320,
        "y": 560,
        "wires": [
            [
                "dcf9c317.89a58",
                "53a321b6.f7ce68"
```

```
        ]
      ]
    },
    {
      "id": "73081dfb.6df0f4",
      "type": "ui_gauge",
      "z": "864171cc.c022b8",
      "name": "",
      "group": "1f8fddae.f90832",
      "order": 0,
      "width": 0,
      "height": 0,
      "gtype": "gage",
      "title": "AI4 : température",
      "label": "",
      "format": "{{value|number:0}}°C",
      "min": 0,
      "max": "100",
      "colors": [
        "#00b500",
        "#e6e600",
        "#ca3838"
      ],
      "x": 690,
      "y": 560,
      "wires": []
    },
    {
      "id": "53a321b6.f7ce68",
      "type": "function",
      "z": "864171cc.c022b8",
      "name": "mise a echelle AI4",
      "func": "msg.payload=(msg.payload-32767)/32;\nreturn msg;",
      "outputs": 1,
      "noerr": 0,
      "x": 510,
      "y": 560,
      "wires": [
        [
          "73081dfb.6df0f4"
        ]
      ]
    },
    {
      "id": "1f8fddae.f90832",
      "type": "ui_group",
      "z": "",
      "name": "ADAM6017 PISCINE UDP",
      "tab": "a606236d.1a74e8",
      "disp": false,
      "width": "3"
    },
    {
      "id": "a606236d.1a74e8",
      "type": "ui_tab",
      "z": "864171cc.c022b8",
      "name": "ADAM PISCINE UDP",
      "icon": "dashboard"
    }
```

]

5. EXERCICE : RÉCEPTION D'UNE RÉQUÊTE HTTP POST

Objectif : recevoir les données envoyées par un formulaire HTML en POST et afficher les données reçues.

Sources : https://cookbook.nodered.org/http/post-form-data-to-a-flow

5.1. Post form data to a flow

5.1.1 Problem

You want to post form data to a flow.

5.1.2 Solution

Use the `HTTP In` node to listen for POST requests that have their `Content-Type` set to `application/x-www-form-urlencoded` and access the form data as properties of `msg.payload`.

Example

```
[{"id":"5b98a8ac.a46758","type":"http
in","z":"3045204d.cfbae","name":"","url":"/hello-
form","method":"post","swaggerDoc":"","x":120,"y":820,"wires":
[["bba61009.4459f"]]},
{"id":"bba61009.4459f","type":"template","z":"3045204d.cfbae","name":"page"
,"field":"payload","fieldType":"msg","format":"handlebars","syntax":"mustac
he","template":"<html>\n    <head></head>\n    <body>\n        <h1>Hello {{
payload.name  }}!</h1>\n         </body>\n</html>","x":290,"y":820,"wires":
[["6ceb930a.93146c"]]},{"id":"6ceb930a.93146c","type":"http
response","z":"3045204d.cfbae","name":"","x":430,"y":820,"wires":[]}]

[~]$ curl -X POST -d "name=Nick" http://localhost:1880/hello-form
<html>
    <head></head>
    <body>
        <h1>Hello Nick!</h1>
    </body>
</html>
```

5.1.3 Discussion

HTML Forms can be used to send data from the browser back to a server. If configured to `POST` the data, the browser will encode the data held in the `<form>` using a `content-type` of `application/x-www-form-urlencoded`.

For example, when a form that looks like this is submitted:

```
<form action="http://localhost:1880/hello-form" method="post">
  <input name="name" value="Nick">
  <button>Say hello</button>
</form>
```

it results in the request:

```
POST / HTTP/1.1
Host: localhost:1880
Content-Type: application/x-www-form-urlencoded
Content-Length: 9

name=Nick
```

When the `HTTP In` node receives such a request, it parses the body of the request and makes the form data available under `msg.payload`:

```
var name = msg.payload.name;
```

5.2. Test personnel

5.2.1 Méthodologie :

1. création d'une page html contenant un formulaire envoyant des données : température=temp et un texte=text'
destination : localhost:1880 (serveur node-red) ou localhost:1880

2. création d'un flow node-red permettant de recevoir les données
création d'un dashboard

3. test à partir d'un PC dans le réseau.

5.2.2 Questions à se poser :

Où est hébergée la page HTML qui envoie le POST ? :
solution 1 : sur site web WAMP,
solution 2 : utiliser un IoT qui envoie un POST (exemple : VoteurWifi)

5.3. solution 1 : WAMP

5.3.1 Préliminaire

Cette solution est facile à mettre en œuvre sur un unique PC sur lequel on installe WAMP et node-red.

5.3.2 Création page web

On crée une page web simple :

Attention : les "name" des <input> du formulaire sont utilisés par le POST et donc par le flow node-red.

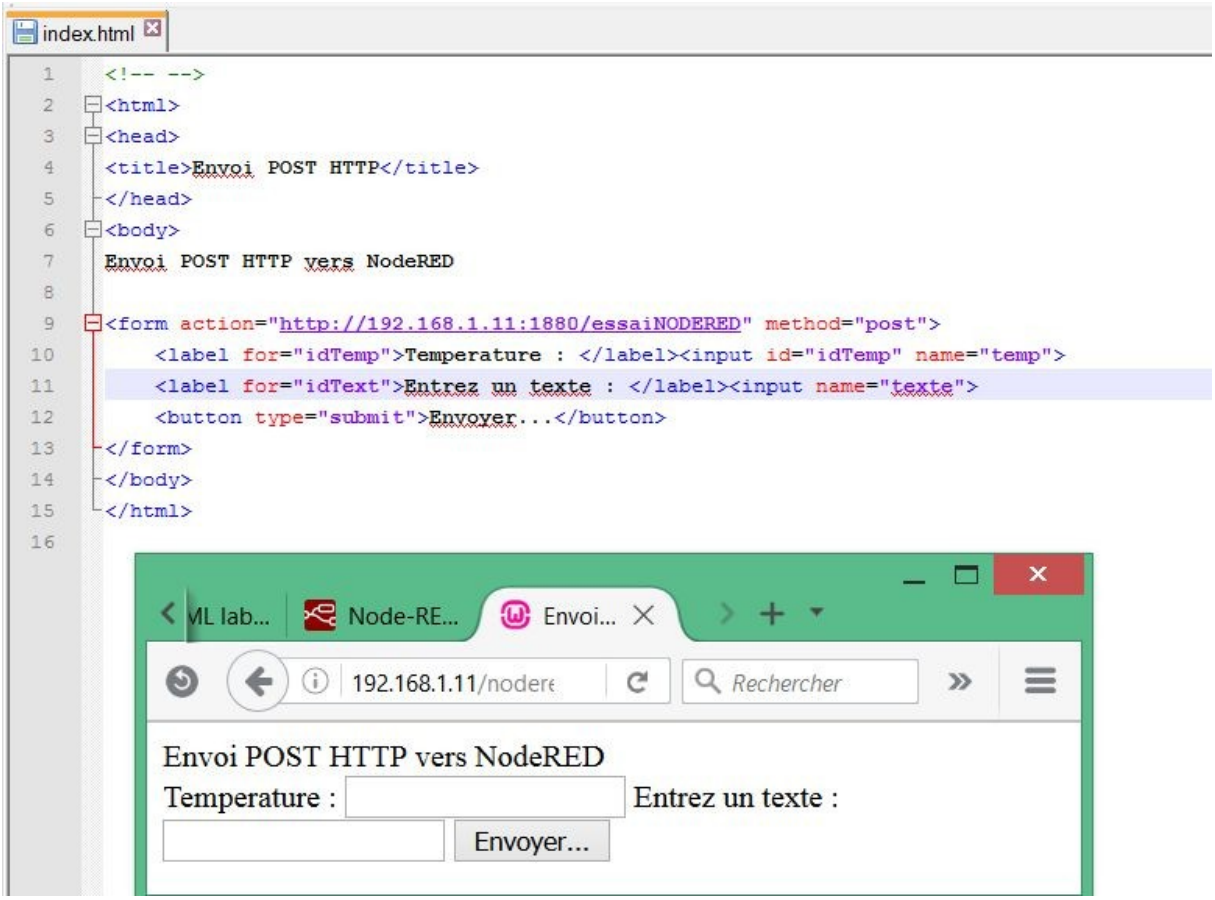


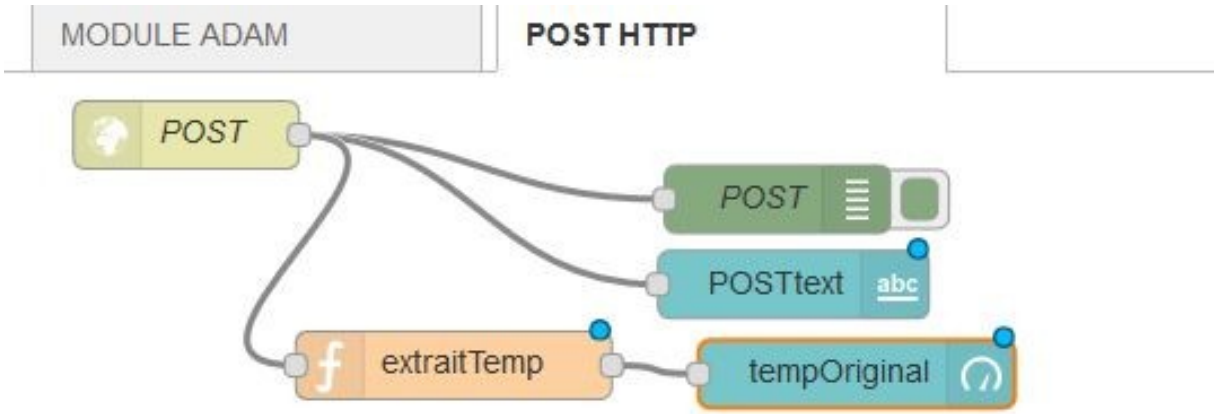
Illustration 1: Page HTML hébergée sur Wamp

On retrouve 2 entrées de formulaire :
une pour la température (name='temp')
une pour un texte (name='texte')

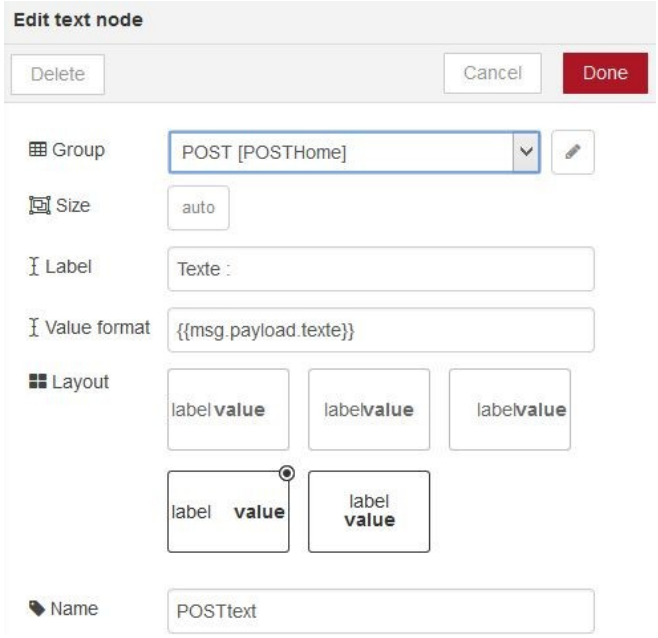
Cette page est hébergée en local sur le PC (IP 192.168.1.11) dans le dossier wamp\www\noderedPOST

Lancer Wamp afin d'accèder à cette page.

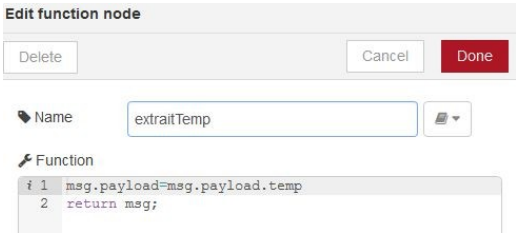
5.4. Création du flow node-red



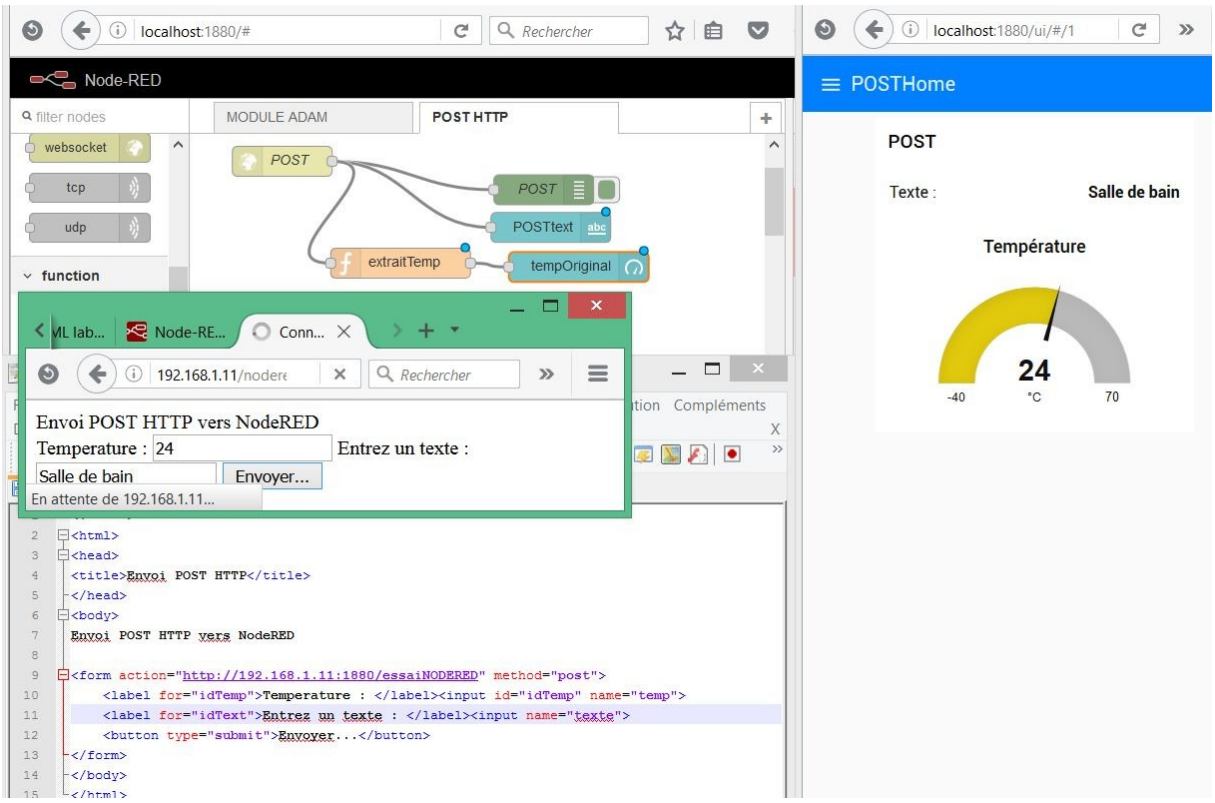
On crée un nouveau flow 'POST HTTP'
On insère un HTTP request : POST
On insère un DEBUG : POST
On insère une sortie Text :



On insère une sortie GAUGE précédée de la fonction 'extraitTemp' afin d'extraire la donnée du POST :



Le résultat est donnée ci dessous :



Ok en local testé.

5.5. Test en mode réseau

A l'aide d'un PC ou d'un smartphone connectez vous sur le réseau contenant : le serveur node-red et Wamp.

Ok testé en réseau

5.6. Solution 2 : Utilisation du voteurwifi

On souhaite créer une IHM avec node-red pour afficher le vote du "voteurwifi"

1. Création d'un réseau wifi sur le PC hôte du serveur node-red (plus pratique que d'utiliser un réseau autre)
2. Création de l'ihm avec node-red
3. Modification de programme voteurwifi.c (IP et port du serveur)
4. Test

5.6.1 Création du réseau wifi

On crée le réseau en configurant le PC en hotspot :

1. Avec l'invite de commande en mode "administrateur"
Tapez :

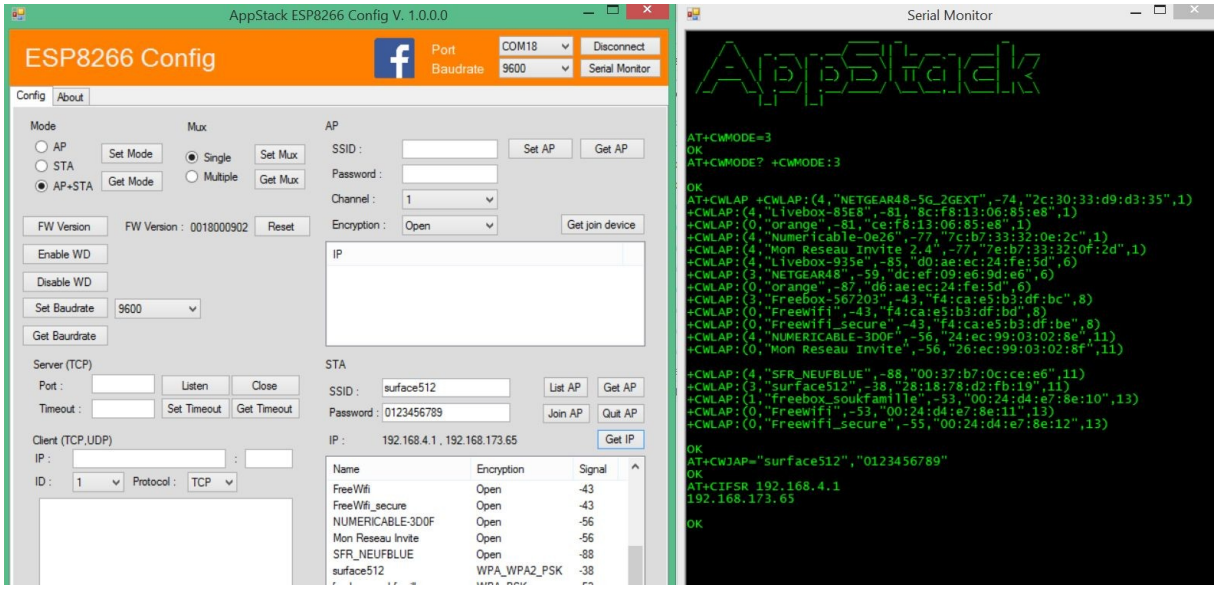
netsh wlan set hostednetwork mode =allow ssid="Connectify-sb" key="0123456789"

(le ssid et le pass sont choisis afin de ne pas modifier mes programmes ESP8266 en mode client:)

2. Démarrer le hotspot en rendant le point d'accès visible :
dans l'invite de commande en mode admin :
Tapez :

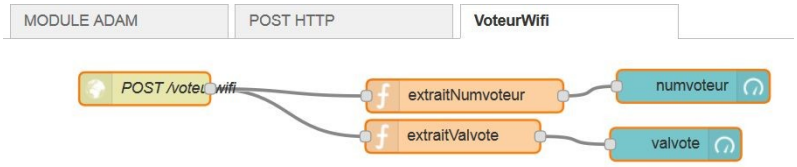
netsh wlan start hostednetwork

On configure l'ESP8266 pour accéder à ce réseau



5.6.2 Création de l'ihm avec node-red

Le flow :

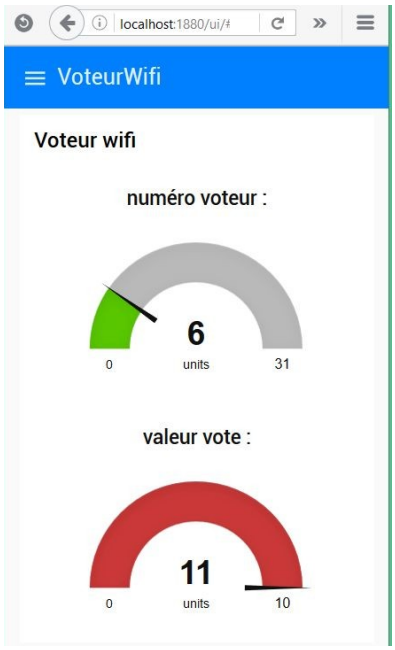


Le code du flow :


```
[
  {
    "id": "d0900776.692da8",
    "type": "http in",
    "z": "e81d573e.995da",
    "name": "POST /voteurwifi",
    "url": "/voteurwifi",
    "method": "post",
    "swaggerDoc": "",
    "x": 147.49998474121094,
    "y": 130,
    "wires": [
      [
        "3b5eaa0d.2a9aae",
        "effddf83.526908"
      ]
    ]
  },
  {
    "id": "dc358b6d.6f2a7",
    "type": "ui_gauge",
    "z": "e81d573e.995da",
    "name": "numvoteur",
    "group": "da562f1f.529858",
    "order": 0,
    "width": 0,
    "height": 0,
    "gtype": "gage",
    "title": "numéro voteur :",
    "label": "units",
    "format": "{{value}}",
    "min": 0,
    "max": "31",
    "colors": [
      "#00b500",
      "#e6e600",
      "#ca3838"
    ],
    "x": 649.5,
    "y": 127.80000305175781,
    "wires": []
  },
  {
    "id": "a689d7eb.383f2",
    "type": "ui_gauge",
    "z": "e81d573e.995da",
    "name": "valvote",
    "group": "da562f1f.529858",
    "order": 0,
    "width": 0,
    "height": 0,
    "gtype": "gage",
    "title": "valeur vote :",
    "label": "units",
    "format": "{{value}}",
    "min": 0,
    "max": 10,
    "colors": [
      "#00b500",
```

```
        "#e6e600",
        "#ca3838"
      ],
      "x": 633.5,
      "y": 180.80003356933594,
      "wires": []
    },
    {
      "id": "3b5eaa0d.2a9aae",
      "type": "function",
      "z": "e81d573e.995da",
      "name": "extraitValvote",
      "func": "msg.payload=msg.payload.val_vote\nreturn msg;",
      "outputs": 1,
      "noerr": 0,
      "x": 429.5,
      "y": 173.6000213623047,
      "wires": [
        [
          "a689d7eb.383f2"
        ]
      ]
    },
    {
      "id": "effddf83.526908",
      "type": "function",
      "z": "e81d573e.995da",
      "name": "extraitNumvoteur",
      "func": "msg.payload=msg.payload.num_voteur\nreturn msg;",
      "outputs": 1,
      "noerr": 0,
      "x": 440.5,
      "y": 136.59999084472656,
      "wires": [
        [
          "dc358b6d.6f2a7"
        ]
      ]
    },
    {
      "id": "da562f1f.529858",
      "type": "ui_group",
      "z": "",
      "name": "Voteur wifi",
      "tab": "17de3bcb.c17394",
      "disp": true,
      "width": "6"
    },
    {
      "id": "17de3bcb.c17394",
      "type": "ui_tab",
      "z": "",
      "name": "VoteurWifi",
      "icon": "dashboard"
    }
]
```


Le visuel :


localhost:1880/ui/#
VoteurWifi
Voteur wifi
numéro voteur :
6
0
units
31
valeur vote :
11
0
units
10

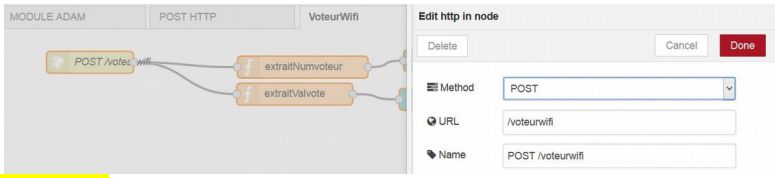
5.6.3 Modification du programme voteurwifi.c

Il faut envoyer un POST à 192.168.173.1:1880 (IP du PC serveur node-red)
Il faut utiliser les variables : "num_voteur" et "val_vote"

Le code .C :

```
void EnvoiRequetePOST(int8 mesure1,mesure2){
//Envoi la requete POST au module par commande AT : la requete contient les données mesure1 et mesure2 (ici
mesure1 = NUM_VOTEUR mesure2 = VAL_VOTE
//ok testé

    printf("AT+CIPSTART=\"TCP\",\"192.168.173.1\",1880\r\n");
    testOK();
    //test LINKED
    while(getc()!='L'){};
    while(getc()!='i'){};
    while(getc()!='n'){};
    while(getc()!='k'){};
    while(getc()!='e'){};
    while(getc()!='d'){};

   delay_ms(1000);

   printf("AT+CIPSEND=146\r\n");//=xxx avec xxx le nombre d'octet à envoyer
   printf("POST /voteurwifi HTTP/1.1\r\n");//29octets-2\=27
   printf("Host: 192.168.173.1\r\n");//23-2\=21
   printf("Content-Type: application/x-www-form-urlencoded\r\n");//51-2\ = 49
   printf("Content-Length: 27\r\n\r\n");//26-4\ = 22 //27=taille de la  ligne suivante
   printf("num_voteur=%3u&val_vote=%3u",mesure1,mesure2);//27 donc total = 159

   printf("AT+CIPCLOSE\r\n");//fermeture connection pas de test car retour varié !
   delay_ms(1000);//
   printf("AT+CWQAP\r\n");testOK();//désactive la connexion

}//fin EnvoiRequetePOST()
```

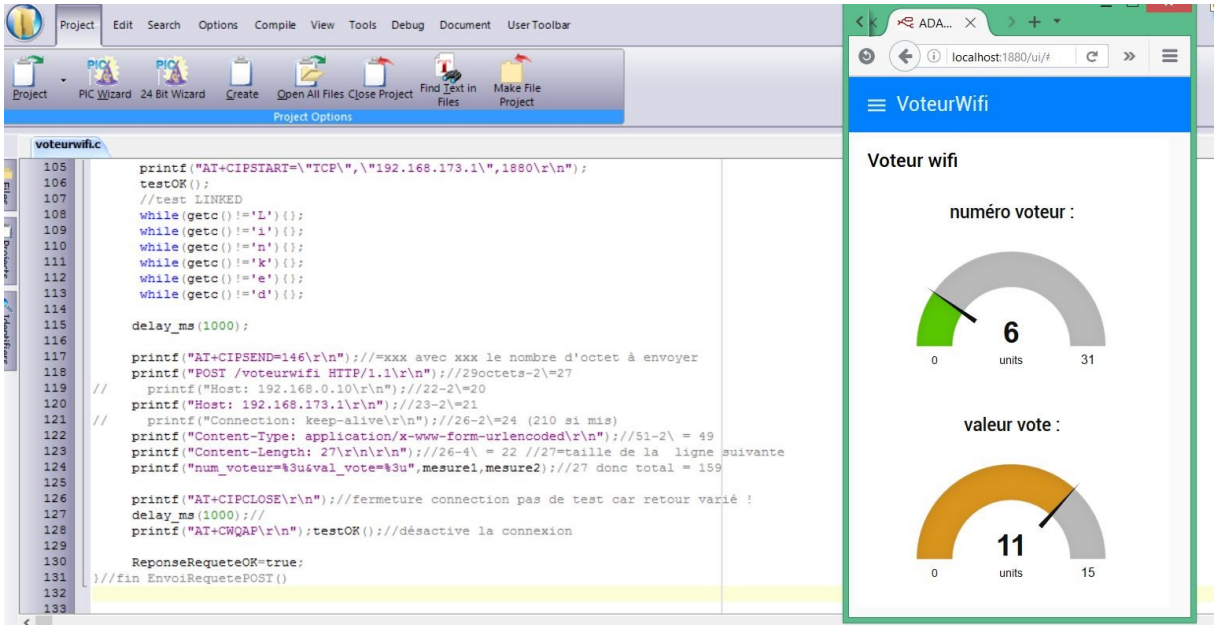
Remarque :



printf("POST /voteurwifi HTTP/1.1\r\n");//29octets-2\=27 ce code correspond à la configuration du node HTTP.

5.6.4 Test complet

Après avoir programmé le VoteurWifi
Après avoir lancer le serveur node-red
Après avoir créer le réseau wifi : SSID : surface512 key : 0123456789
En appuyant sur le voteurwifi on obtient bien l'affichage des informations sur l'IHM :



OK testé.

6. RESSOURCES :

https://fred.sensetecnic.com/
http://noderedguide.com
https://projetsdiy.fr/