



Lab 4: Smart Home Monitoring using OM2M, Node-RED, and Wyliodrin

物聯網技術與應用(英) IoT/M2M Technologies and Applications

國立交通大學資訊工程系

Department of Computer Science

National Chiao Tung University

November 30, 2018

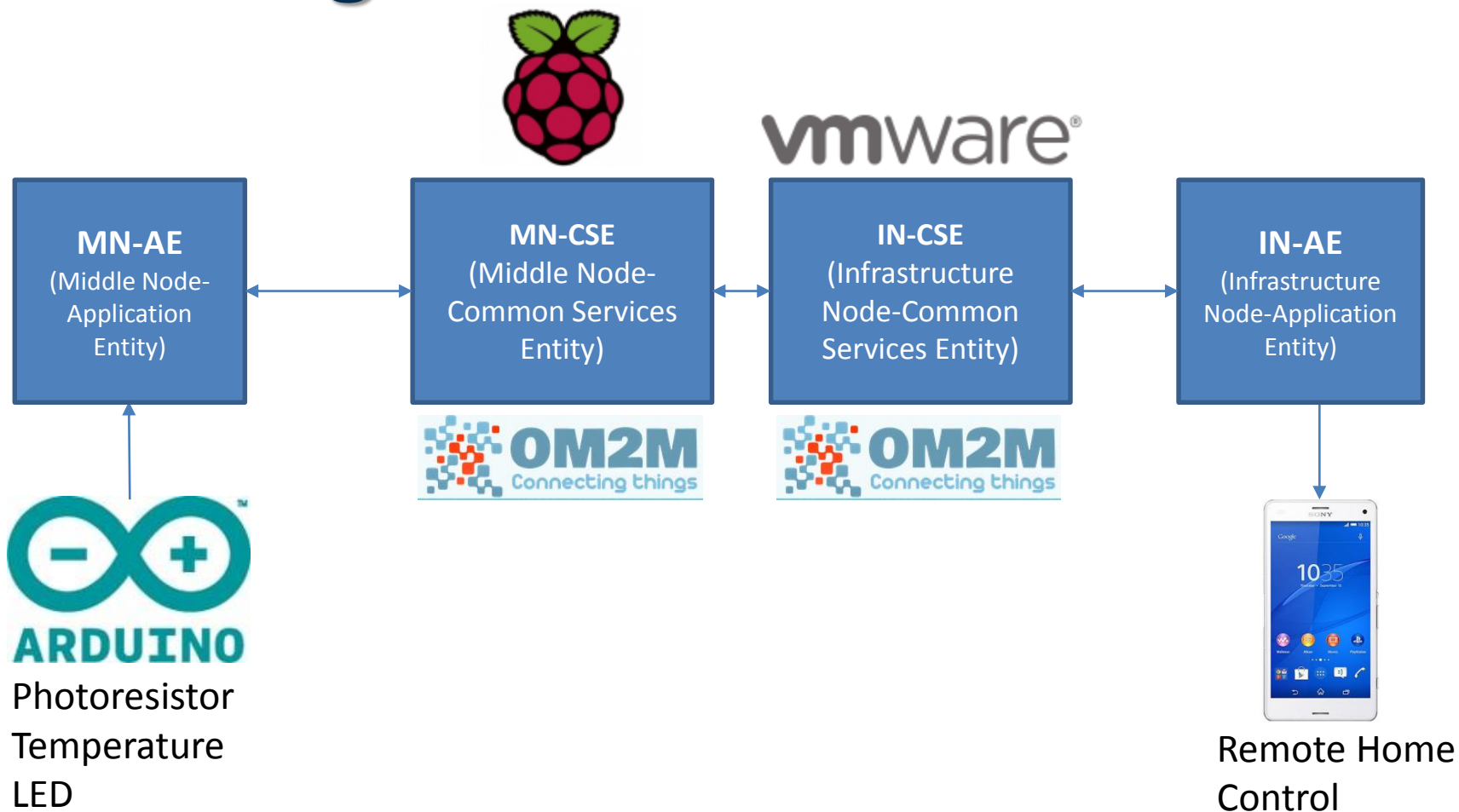
Outline

- High Level Architecture.
- Configuring IN-CSE (VM) and MN-CSE (Raspberry Pi). *(Checkpoint 1)*
- Creating a Middle Node-Application Entity (MN-AE) with HTTP Server Capabilities using Node-Red. *(Checkpoint 2)*
- Sending Smart Home Data from Raspberry Pi to MN-AE using Node-Red. *(Checkpoint 3)*
- Subscribing to Smart Home Data and Extracting Sensor Values using Node-Red. *(Checkpoint 4)*

Attention!

Please, start your virtual machine, connect your Raspberry Pi to the power source now, and connect your Arduino to your Raspberry Pi as you did in Lab 2.

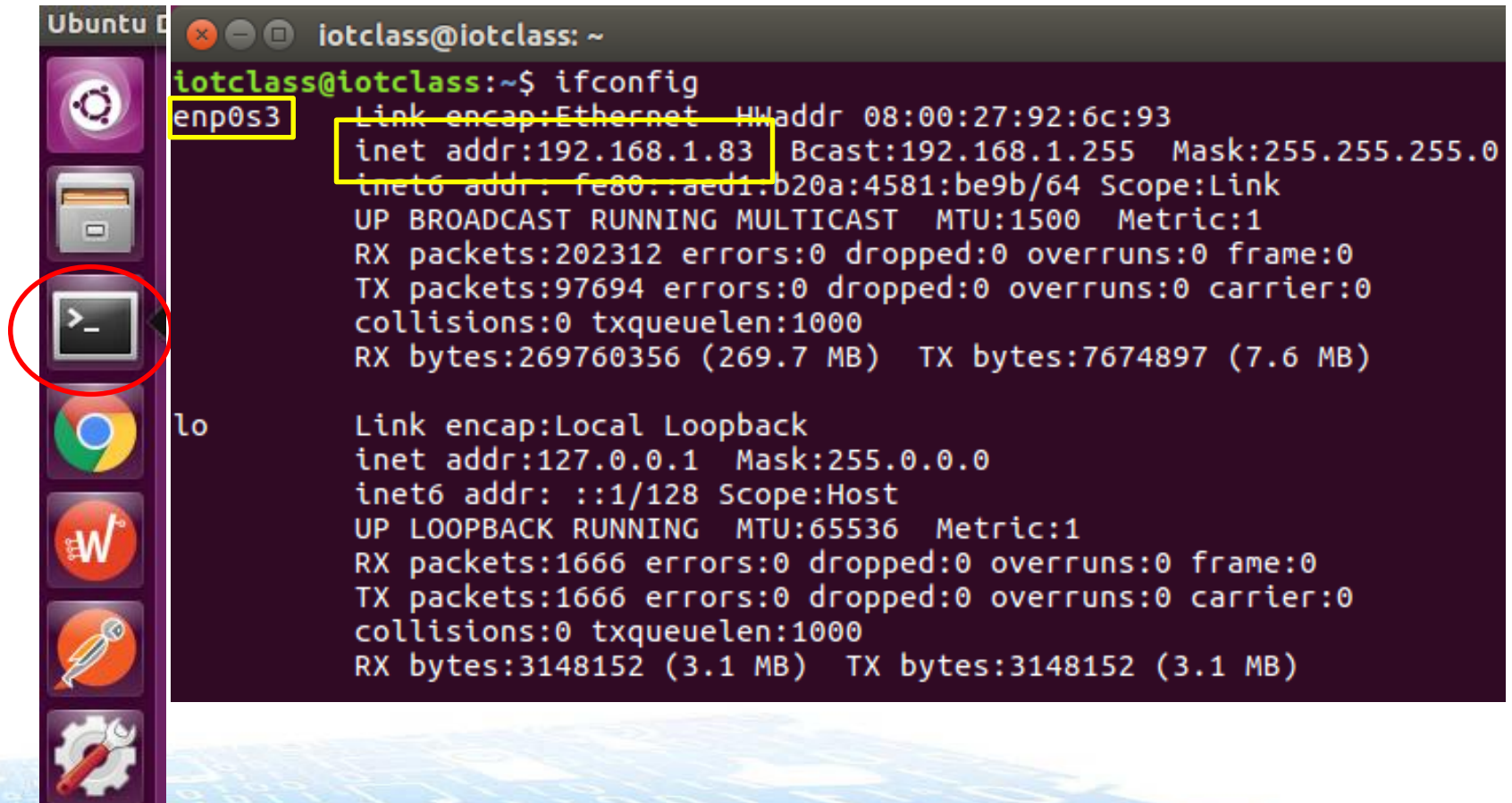
High Level Architecture



CONFIGURING IN-CSE

Finding out your VM IP Address

1. Open a terminal window (Click on the icon shown below).
2. Execute “ifconfig”.
3. **YOUR.VM.IP.ADDRESS** is under the first network interface information.



```
iotclass@iotclass: ~  
iotclass@iotclass:~$ ifconfig  
enp0s3: Link encap:Ethernet HWaddr 08:00:27:92:6c:93  
        inet addr:192.168.1.83 Bcast:192.168.1.255 Mask:255.255.255.0  
        inet6 addr: fe80::aedi:b20a:4581:be9b/64 Scope:Link  
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
        RX packets:202312 errors:0 dropped:0 overruns:0 frame:0  
        TX packets:97694 errors:0 dropped:0 overruns:0 carrier:0  
        collisions:0 txqueuelen:1000  
        RX bytes:269760356 (269.7 MB) TX bytes:7674897 (7.6 MB)  
  
lo: Link encap:Local Loopback  
      inet addr:127.0.0.1 Mask:255.0.0.0  
      inet6 addr: ::1/128 Scope:Host  
      UP LOOPBACK RUNNING MTU:65536 Metric:1  
      RX packets:1666 errors:0 dropped:0 overruns:0 frame:0  
      TX packets:1666 errors:0 dropped:0 overruns:0 carrier:0  
      collisions:0 txqueuelen:1000  
      RX bytes:3148152 (3.1 MB) TX bytes:3148152 (3.1 MB)
```

Changing IN-CSE Configuration (1)

1. In another terminal window, execute:
2. `cd om2m/in-cse/configuration/`
3. `nano config.ini`

```
iotclass@ubuntu: ~/om2m/in-cse/configuration
iotclass@ubuntu:~$ cd om2m/in-cse/configuration/
iotclass@ubuntu:~/om2m/in-cse/configuration$ nano config.ini
iotclass@ubuntu:~/om2m/in-cse/configuration$
```


Changing IN-CSE Configuration (2)

4. Set org.eclipse.om2m.cseBaseAddress=**YOUR.VM.IP.ADDRESS**
5. Save the changes (Press CTRL + X, then press Y, finally press ENTER).

```
iotclass@iotclass: ~/om2m/IN-CSE/configuration
GNU nano 2.5.3      File: config.ini

#This configuration file was written by: org.eclipse.equinox.internal.framework$
#Fri Jul 08 10:21:12 CEST 2016
log4j.configuration=file\:./log4j.configuration
org.eclipse.equinox.http.jetty.http.port=8080
org.eclipse.om2m.dbReset=false
org.eclipse.om2m.cseBaseContext=/
org.eclipse.om2m.globalContext=
osgi.bundles=reference\:file\:javax.servlet_3.1.0.v20140303-1611.jar@4,referenc$
org.eclipse.om2m.cseBaseProtocol.default=http
org.eclipse.om2m.cseBaseName=tn-name
org.eclipse.om2m.cseBaseAddress=192.168.1.83
eclipse_p2_profile=DefaultProfile
org.eclipse.om2m.dbUrl=jdbc\:h2\:./database/indb
osgi.framework.extensions=
org.eclipse.om2m.webInterfaceContext=/webpage
osgi.bundles.defaultStartLevel=4
org.eclipse.om2m.dbUser=om2m
osgi.framework=file\:plugins/org.eclipse.osgi_3.10.2.v20150203-1939.jar
org.eclipse.om2m.guestRequestingEntity=quest\:quest

[ Read 29 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File ^\ Replace  ^U Uncut Text ^T To Spell  ^_ Go To Line
```

Line 11.

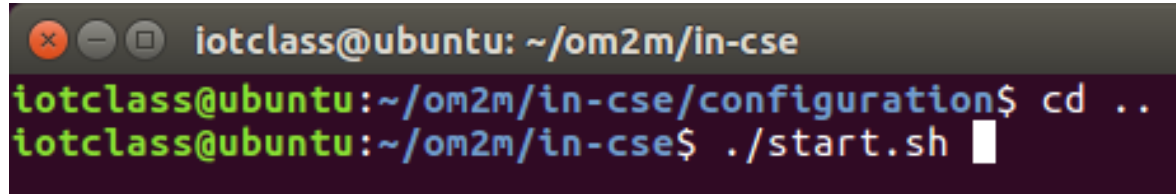
YOUR.VM.IP.ADDRESS

Start OM2M IN-CSE

- Enter the following commands:

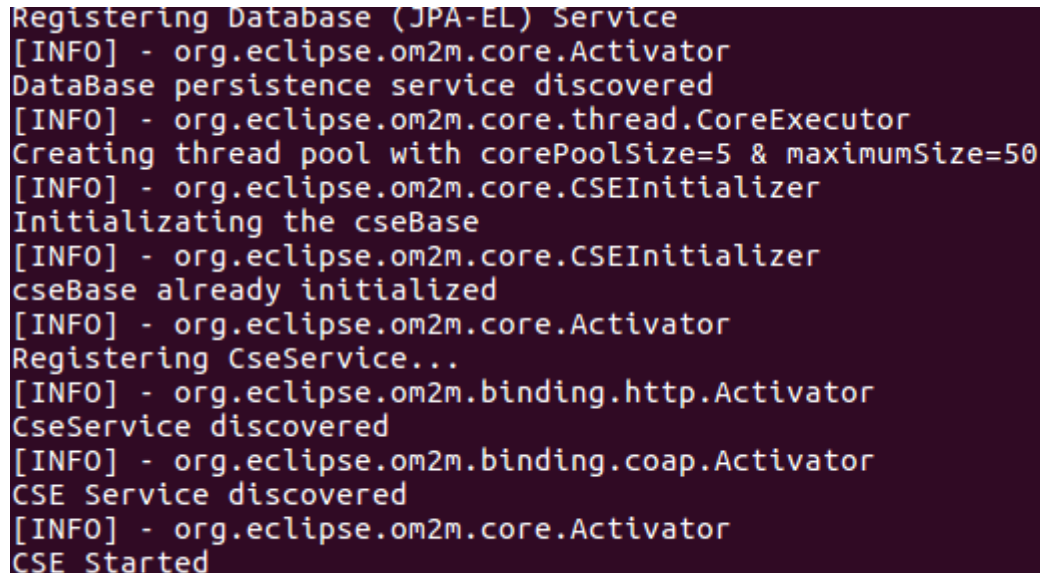
```
$ cd ..
```

```
$ ./start.sh
```

A screenshot of a terminal window showing the execution of commands. The prompt is "iotclass@ubuntu: ~/om2m/in-cse". The first command is "cd .." and the second is "./start.sh".

```
iotclass@ubuntu: ~/om2m/in-cse
iotclass@ubuntu:~/om2m/in-cse/configuration$ cd ..
iotclass@ubuntu:~/om2m/in-cse$ ./start.sh
```

- After starting it successfully, you will see “CSE Started” in your terminal.

A screenshot of a terminal window showing the output of the ./start.sh command. The output includes several log messages from the Eclipse OM2M core, indicating that the database service, thread pool, and CSE service have been successfully initialized and started.

```
Registering Database (JPA-EL) Service
[INFO] - org.eclipse.om2m.core.Activator
DataBase persistence service discovered
[INFO] - org.eclipse.om2m.core.thread.CoreExecutor
Creating thread pool with corePoolSize=5 & maximumSize=50
[INFO] - org.eclipse.om2m.core.CSEInitializer
Initializing the cseBase
[INFO] - org.eclipse.om2m.core.CSEInitializer
cseBase already initialized
[INFO] - org.eclipse.om2m.core.Activator
Registering CseService...
[INFO] - org.eclipse.om2m.binding.http.Activator
CseService discovered
[INFO] - org.eclipse.om2m.binding.coap.Activator
CSE Service discovered
[INFO] - org.eclipse.om2m.core.Activator
CSE Started
```

Verify your IN-CSE

Browse to <http://YOUR.VM.IP.ADDRESS:8080/webpage>.

Notice that your IN-CSE is using an IP address now instead of Localhost.



Logout

OM2M CSE Resource Tree

<http://192.168.1.51:8080/~in-cse>



– in-name
└ acp_admin

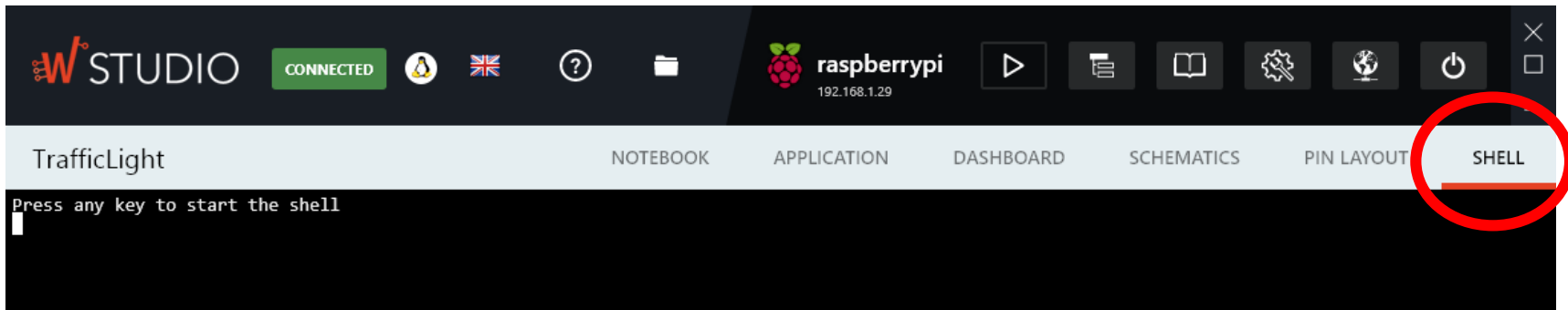


Attribute	Value
m	in-name
ty	5
ri	/in-cse
ct	20180703T152122

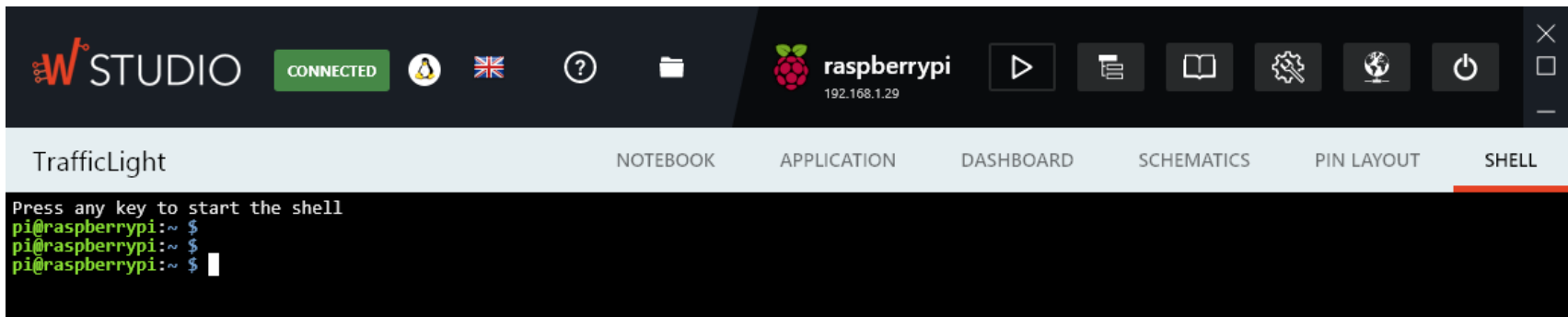
CONFIGURING MN-CSE

Changing MN-CSE Configuration (1)

1. Open a new terminal window and launch Wylidrin Studio. (\$./wylidrin.AppImage)
2. Connect to your Raspberry Pi.
3. Click on the “Shell” tab.

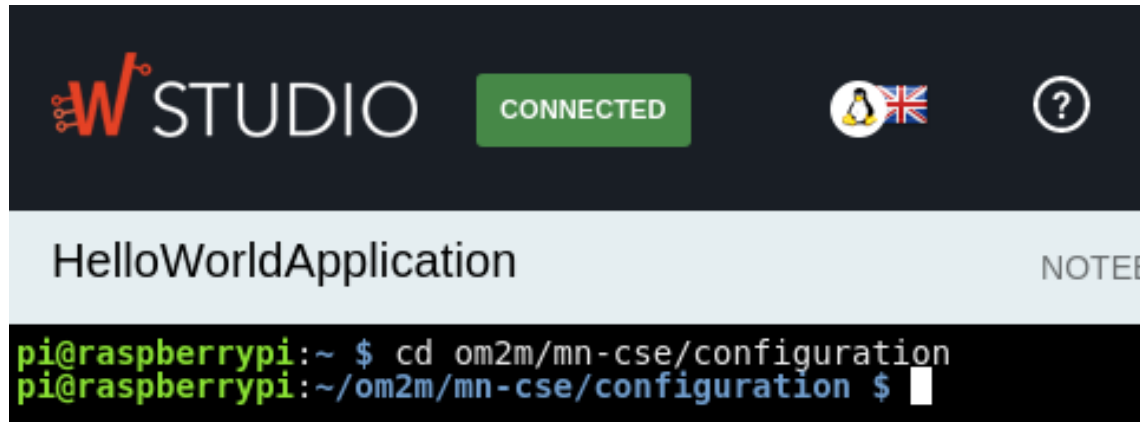


4. Press ENTER.



Changing MN-CSE Configuration (2)

4. Execute the following commands:
5. `cd om2m/mn-cse/configuration`

A screenshot of the WS Studio interface. The top bar shows the "WS STUDIO" logo, a green "CONNECTED" button, a penguin icon with a UK flag, and a help icon. Below the bar, the text "HelloWorldApplication" is displayed on the left and "NOTEBOOK" on the right. The terminal window shows the command `cd om2m/mn-cse/configuration` being executed, with the prompt changing from `pi@raspberrypi:~` to `pi@raspberrypi:~/om2m/mn-cse/configuration`.

```
WS STUDIO CONNECTED [penguin icon] [UK flag icon] [help icon]
HelloWorldApplication NOTEBOOK
pi@raspberrypi:~ $ cd om2m/mn-cse/configuration
pi@raspberrypi:~/om2m/mn-cse/configuration $
```

6. `nano config.ini`

A screenshot of the WS Studio interface, similar to the previous one. The top bar and application name are the same. The terminal window now shows the command `nano config.ini` being entered at the prompt `pi@raspberrypi:~/om2m/mn-cse/configuration`.

```
WS STUDIO CONNECTED [penguin icon] [UK flag icon] [help icon]
HelloWorldApplication NOTEBOOK
pi@raspberrypi:~/om2m/mn-cse/configuration $ nano config.ini
```

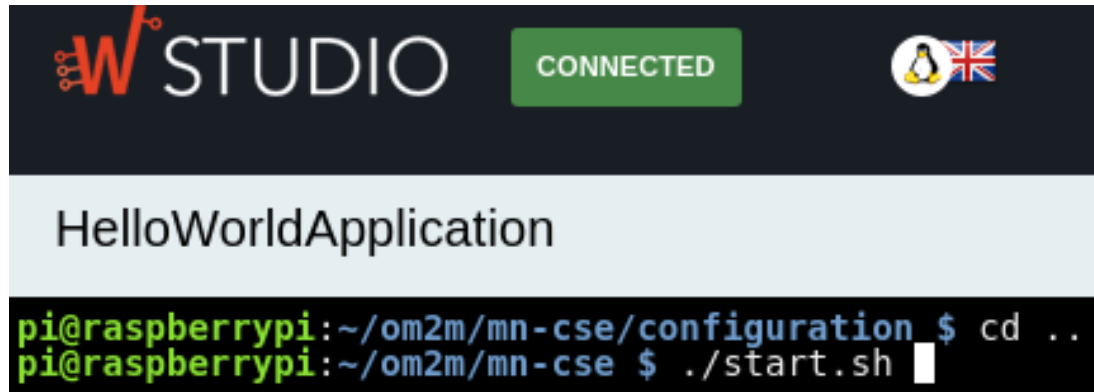
Changing MN-CSE Configuration (3)


7. Set org.eclipse.om2m.cseBaseName=mn-name-pi
8. Set org.eclipse.om2m.cseBaseAddress=YOUR.RASPBERRY.IP.ADDRESS
9. Set org.eclipse.om2m.cseBaseId=mn-cse-pi
10. Set org.eclipse.om2m.remoteCseAddress=YOUR.VM.IP.ADDRESS
11. Save the changes (Press CTRL + X, then press Y, finally press ENTER).

```
org.eclipse.om2m.remoteCsePort=8080
org.eclipse.om2m.cseBaseContext=/
org.eclipse.om2m.globalContext=
osgi.bundles=reference\:file\:javax.servlet_3.1.0.v20140303-1611.jar@4,reference\:file\:javax.xml_1.
org.eclipse.om2m.cseBaseProtocol.default=http
Line 13: org.eclipse.om2m.cseBaseName=mn-name mn-name-pi
Line 14: org.eclipse.om2m.cseBaseAddress=127.0.0.1 YOUR.RASPBERRY.IP.ADDRESS
eclipse.p2.profile=DefaultProfile
org.eclipse.om2m.dbUrl=jdbc\:h2\:./database/mndb
osgi.framework.extensions=
org.eclipse.om2m.webInterfaceContext=/webpage
osgi.bundles.defaultStartLevel=4
org.eclipse.om2m.dbUser=om2m
org.eclipse.om2m.maxNrOfInstances=1000
osgi.framework=file\:plugins/org.eclipse.osgi_3.10.2.v20150203-1939.jar
org.eclipse.om2m.guestRequestingEntity=guest\:guest
org.eclipse.om2m.remoteCseName=in-name
Line 25: org.eclipse.om2m.cseBaseId=mn-cse mn-cse-pi
org.eclipse.om2m.remoteCseContext=/
org.eclipse.om2m.dbDriver=org.h2.Driver
Line 29: org.eclipse.om2m.remoteCseAddress=127.0.0.1 YOUR.VM.IP.ADDRESS
org.eclipse.om2m.adminRequestingEntity=admin\:admin
org.eclipse.om2m.cseType=MN
org.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JLogger
org.eclipse.om2m.cseAuthentication=true
eclipse.p2.data.area=@config.dir/./p2
org.eclipse.om2m.coap.port=5684
org.eclipse.om2m.dbPassword=om2m
```


Starting MN-CSE on Raspberry

12. Execute “cd ..”.
13. Execute “./start.sh”.



```
W STUDIO CONNECTED 
```

```
HelloWorldApplication
```

```
... pi@raspberrypi:~/om2m/mn-cse/configuration $ cd ..
```

```
... pi@raspberrypi:~/om2m/mn-cse $ ./start.sh
```

This process will take a few minutes, it is normal due to the computational capacity of Pi.
In the end, you should see a message “Successfully registered to in-cse” as shown below.

```
]
[INFO] - org.eclipse.om2m.core.CSEInitializer
Successfully registered to in-cse
[INFO] - org.eclipse.om2m.core.remotecse.RemoteCseService
addRemoteCseAndPublish(cseId=/mn-cse-pi/csr-809637383, name=in-name)
[INFO] - org.eclipse.om2m.core.remotecse.RemoteCseService
post Event to inform about RemoteCSE creation (cseId=in-cse, cseName=in-name)
```


Verify your MN-CSE

Browse to <http://YOUR.RASPBERRY.IP.ADDRESS:8282/webpage>.

Notice that your MN-CSE is using an IP address now instead of Localhost.

Also notice the registration of “in-name” which corresponds to the IN-CSE.



Logout

OM2M CSE Resource Tree

<http://192.168.1.125:8282/~mn-cse-pi>



```
– mn-name-pi
  |
  |– acp_admin
  |
  |→ in-name
```



Attribute	Value
rn	mn-name-pi
ty	5
ri	/mn-cse-pi

CHECKPOINT 1!

CREATING MN-AE AND WEB SERVICE

Launching Node-Red

Use a new terminal to launch Node-Red:

\$ node-red

```
ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ node-red

Welcome to Node-RED
=====

2 Aug 20:11:59 - [info] Node-RED version: v0.11.1
2 Aug 20:11:59 - [info] Node.js version: v0.12.7
2 Aug 20:11:59 - [info] Loading palette nodes
2 Aug 20:12:03 - [warn] -----
2 Aug 20:12:03 - [warn] Failed to register 1 node type
2 Aug 20:12:03 - [warn] Run with -v for details
2 Aug 20:12:03 - [warn] -----
2 Aug 20:12:03 - [info] Settings file : /usr/local/lib/node_modules/node-red/se
ttings.js
2 Aug 20:12:03 - [info] User directory : /home/ubuntu/.node-red
2 Aug 20:12:03 - [info] Flows file : /home/ubuntu/.node-red/flows_ubuntu-Virtual
Box.json
2 Aug 20:12:03 - [info] Server now running at http://127.0.0.1:1880/
2 Aug 20:12:03 - [info] Starting flows
2 Aug 20:12:03 - [info] Started flows
```

Before creating new flows in Node-Red (1)

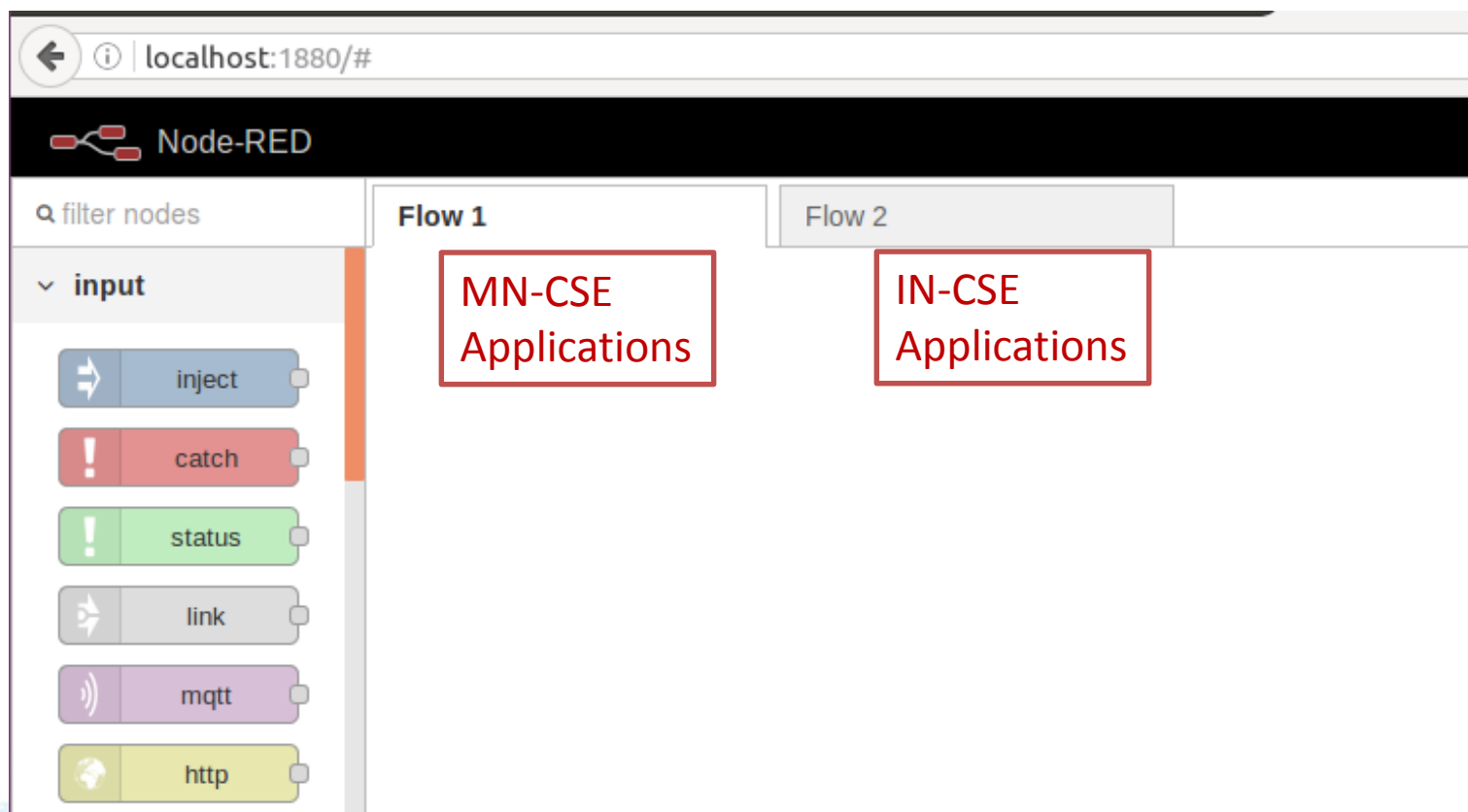
Open <http://localhost:1880> in a browser.

Add a new workspace.



Before creating new flows in Node-Red (2)

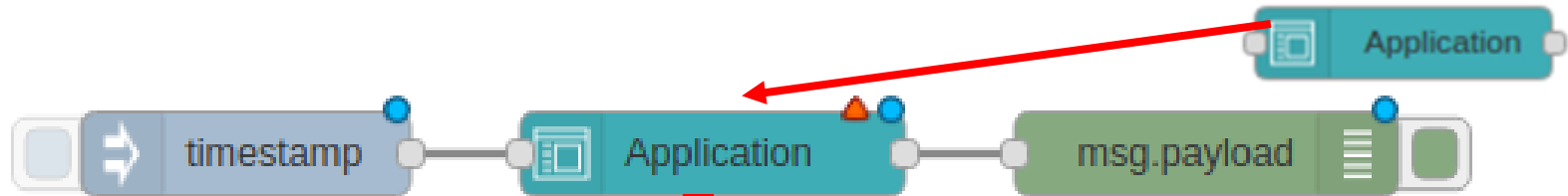
We suggest you to use different workspaces for different applications.
For example, one workspace for MN-CSE and another one for IN-CSE applications.



Creating a MN-AE with Node-RED

Create a new IDE OM2M Application according to the pictures.

IDE OM2M



Edit Application node

Delete

Cancel

node properties

Platform: MN-CSE-PI

Application: SMART_HOME

Point of Access: poa

Announce: ☒

Label	Value
Location	pi

Name: Smart Home Application in Pi

Edit Application node > Edit xN_CSE node

Delete

Cancel

Update

Platform: MN-CSE-PI

URLBase: http://192.168.1.125:8282/~mn-cse-pi/mn-name-1

Username: admin

Password:

admin

URLBase = http://**YOUR.RASPBERRY.IP.ADDRESS**:8282/~mn-cse-pi/mn-name-pi

Reminder:

Deploy and Trigger the Flows

Remember to push the Deploy button in Node-Red every time you made changes.



After successfully deploying, trigger you new flows.



Creating a MN-AE with Node-RED

Deploy the flows in Node-Red and trigger the flow.

Verify that your application has been creating by accessing OM2M GUI.



Logout

OM2M CSE Resource Tree

<http://192.168.1.125:8282/~ /mn-cse-pi/CAE958638652>



– mn-name-pi

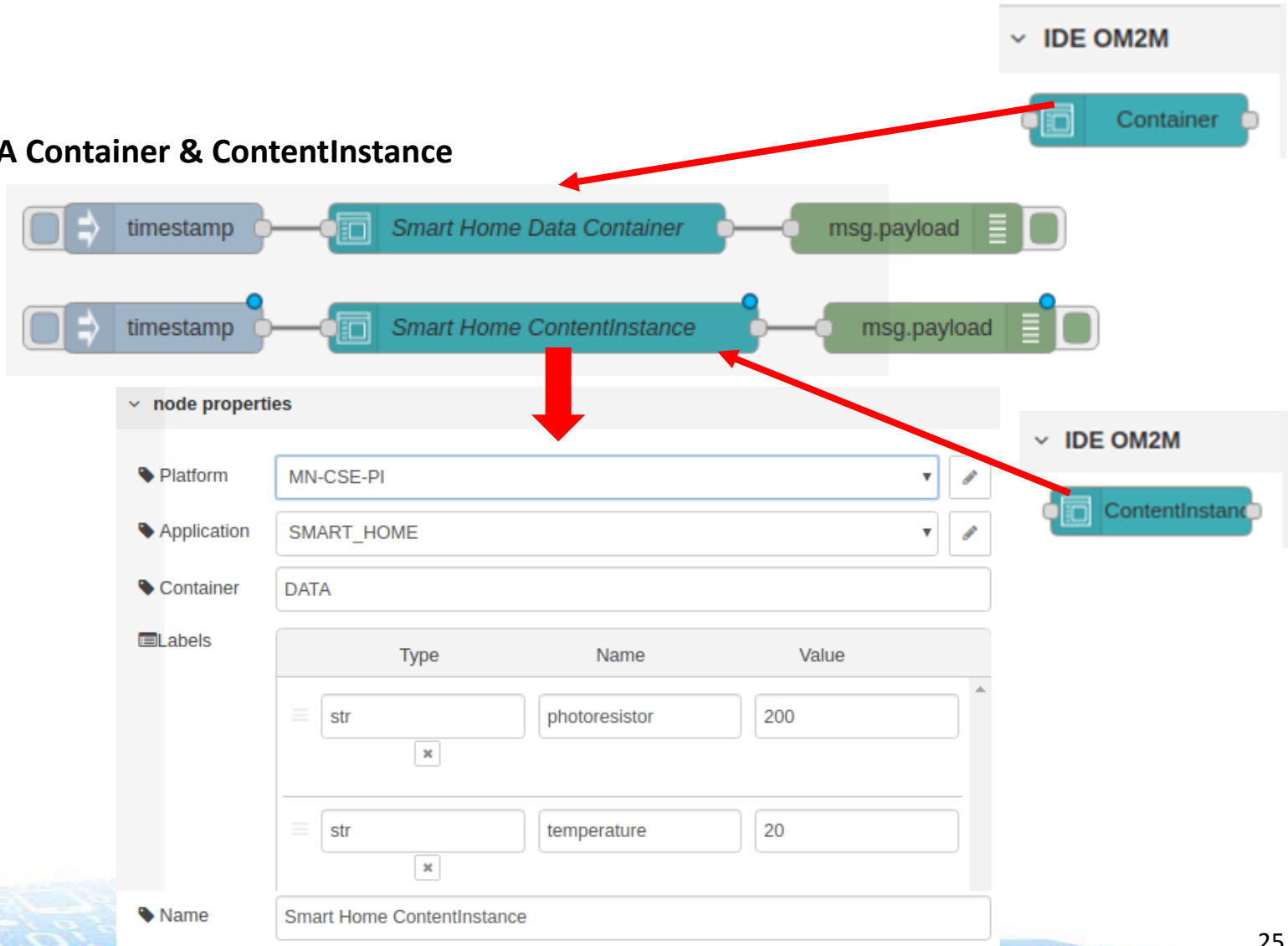
- acp_admin
- SMART_HOME**
- in-name

Attribute	Value
rn	SMART_HOME
ty	2
ri	/mn-cse-pi/CAE958638652
pi	/mn-cse-pi
ct	20180703T094547
lt	20180703T094547
...	...

Creating a MN-AE with Node-RED

Create the following resources under your “SMART_HOME” application.

DATA Container & ContentInstance



IDE OM2M

Container

timestamp → Smart Home Data Container → msg.payload

timestamp → Smart Home ContentInstance → msg.payload

node properties

Platform: MN-CSE-PI

Application: SMART_HOME

Container: DATA

Labels

Type	Name	Value
str	photoresistor	200
str	temperature	20

Name: Smart Home ContentInstance

IDE OM2M

ContentInstance

Creating a MN-AE with Node-RED

Deploy the flows in Node-Red and trigger the flows.

Verify that your DATA container and ContentInstance have been created.

Logout

OM2M CSE Resource Tree

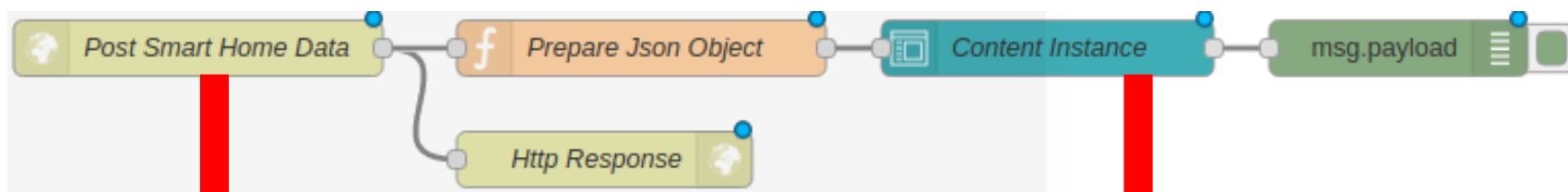
<http://192.168.1.125:8282/~mn-cse-pi/cin-239932425>

- mn-name-pi
 - acp_admin
 - SMART_HOME
 - DATA
 - cin_239932425
 - in-name

Attribute	Value						
rn	cin_239932425						
ty	4						
ri	/mn-cse-pi/cin-239932425						
pi	/mn-cse-pi/cnt-224633082						
ct	20180703T112557						
lt	20180703T112557						
st	0						
cnf	message						
cs	126						
con	<table border="1"> <thead> <tr> <th>Attribute</th><th>Value</th></tr> </thead> <tbody> <tr> <td>photoresistor</td><td>200</td></tr> <tr> <td>temperature</td><td>20</td></tr> </tbody> </table>	Attribute	Value	photoresistor	200	temperature	20
Attribute	Value						
photoresistor	200						
temperature	20						

Creating a MN-AE with Node-RED

Create a “postSmartHomeData” webservice to receive data from real sensors.



Method POST

☐ Accept file uploads?

URL /postSmartHomeData

Name Post Smart Home Data

node properties

Platform MN-CSE-PI

Application SMART_HOME

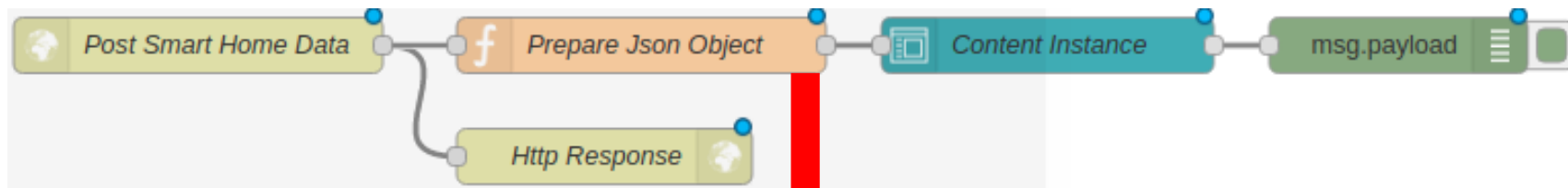
Container DATA

Labels

Type	Name	Value
------	------	-------

Creating a MN-AE with Node-RED

Create a “postSmartHomeData” webservice to receive data from real sensors.



Name

Prepare Json Object

Function

CODE (Copy & Paste in Node Red)

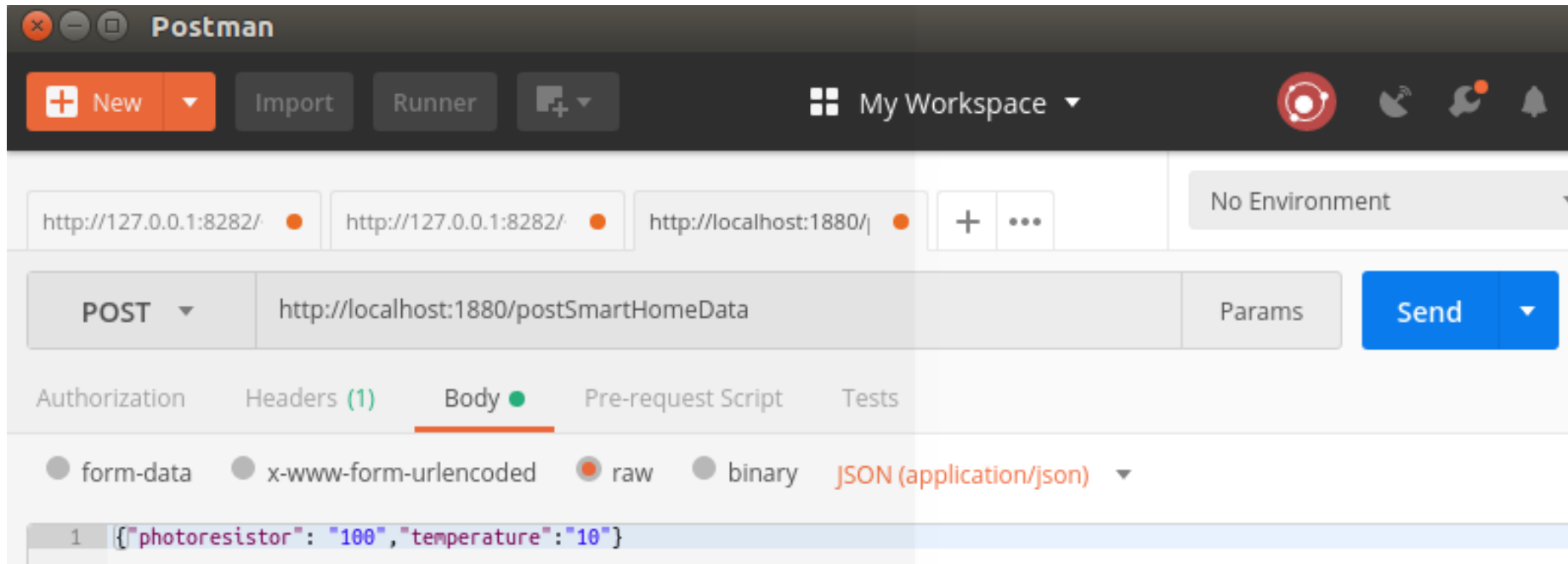
```
var data = [{
  type:'str',
  label:'photoresistor',
  value: msg.payload.photoresistor
},
{
  type:'str',
  label:'temperature',
  value: msg.payload.temperature
}];
msg.externalInput =
JSON.stringify(data);
return msg;
```

```
1 - var data = [{
2   type:'str',
3   label:'photoresistor',
4   value: msg.payload.photoresistor
5 - },
6 - {
7   type:'str',
8   label:'temperature',
9   value: msg.payload.temperature
10 - }];
11 msg.externalInput = JSON.stringify(data);
12 return msg;
```

Creating a MN-AE with Node-RED

Deploy the flows in Node-Red.

Use Postman to verify your webservice works correctly.



Creating a MN-AE with Node-RED

Check the OM2M GUI.

Logout

OM2M CSE Resource Tree

<http://192.168.1.125:8282/~mn-cse-pi/cin-952835110>

- mn-name-pi
 - acp_admin
 - SMART_HOME
 - DATA
 - cin_239932425
 - cin_952835110
 - in-name

Attribute	Value						
rn	cin_952835110						
ty	4						
ri	/mn-cse-pi/cin-952835110						
pi	/mn-cse-pi/cnt-224633082						
ct	20180703T114156						
lt	20180703T114156						
st	0						
cnf	message						
cs	126						
con	<table border="1"> <thead> <tr> <th>Attribute</th><th>Value</th></tr> </thead> <tbody> <tr> <td>photoresistor</td><td>100</td></tr> <tr> <td>temperature</td><td>10</td></tr> </tbody> </table>	Attribute	Value	photoresistor	100	temperature	10
Attribute	Value						
photoresistor	100						
temperature	10						

CHECKPOINT 2!

SENDING SMART HOME DATA FROM PI

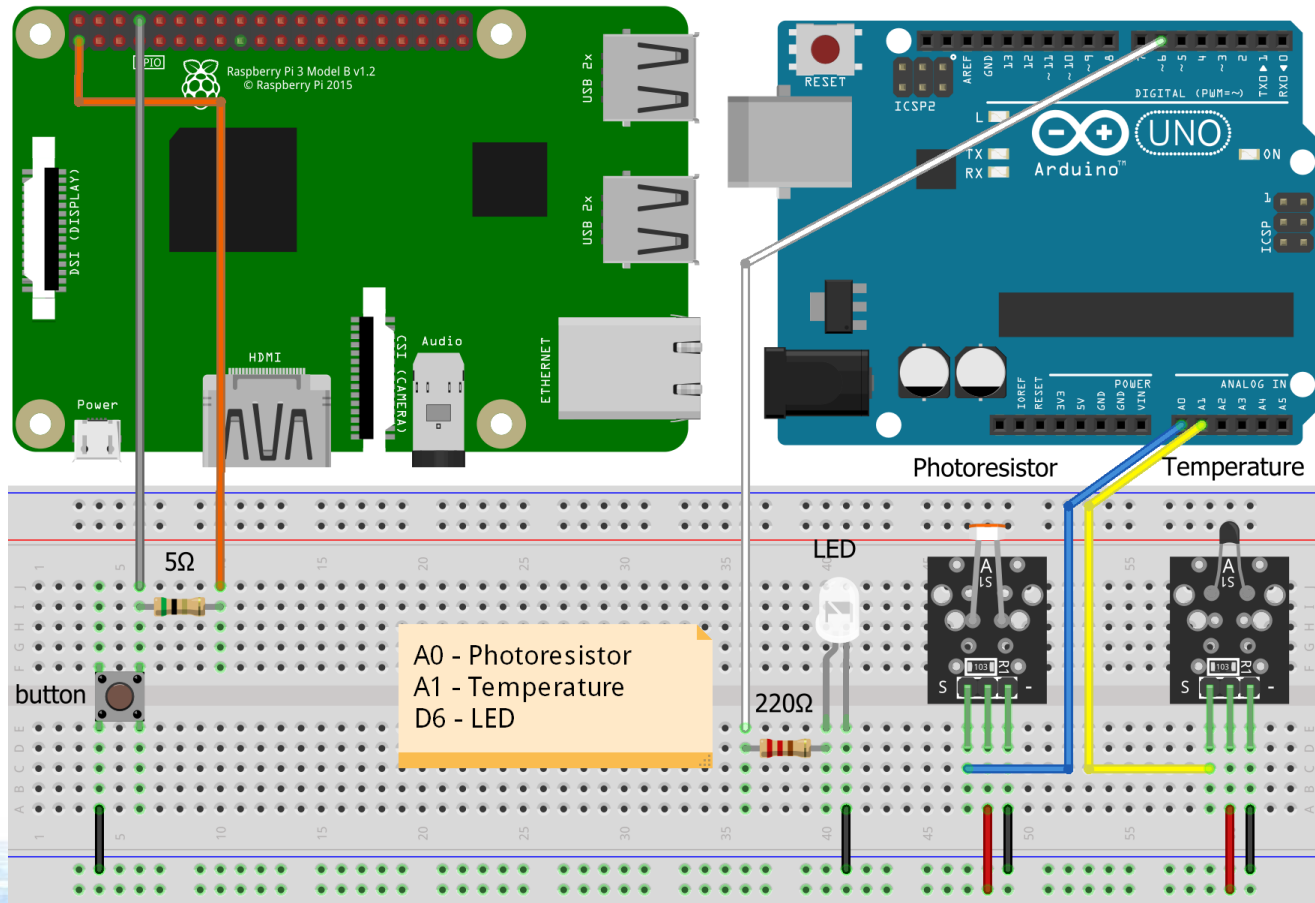
Attention!!

We are going to use the applications created in Labs 1 and 2, and apply some modifications on them.

Make sure your Pi, Arduino, and the Wyliodrin applications from Labs 1 and 2 are ready!

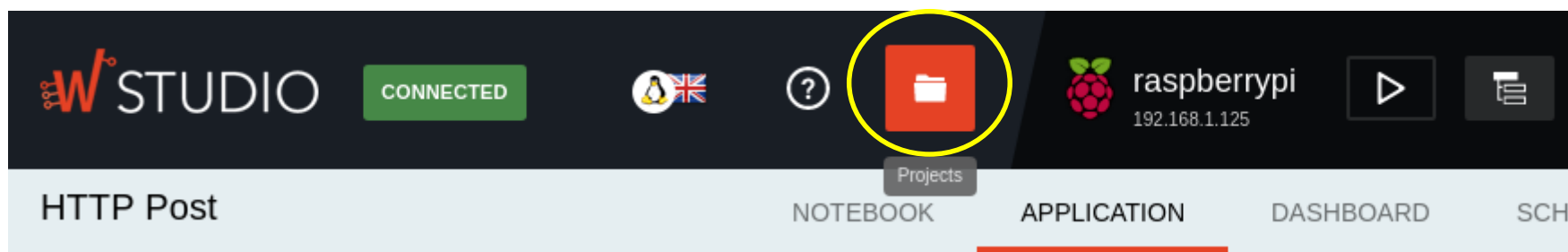
Schematics (Lab 1 + Lab 2)

- Connect all the modules and component according to the schema.
- **Connect Arduino to Raspberry with Arduino USB-cable as you did in Lab 2.**
- **Connect 5V and GND (Ground) to your breadboard as you did in Labs 0, 1, and 2 (Not shown in the diagram!).**
- Ask TA if you are not sure how to connect the components.

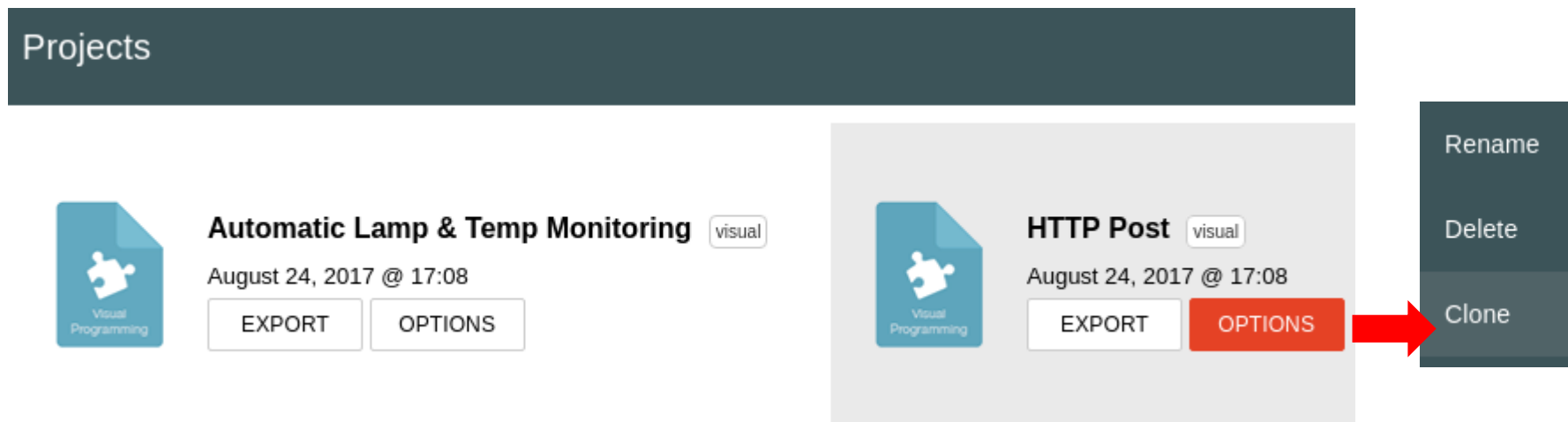


Clone HTTP Post App (1)

- Use Wylidrin to clone your previous “HTTP Post” application.
Click on “Projects”.

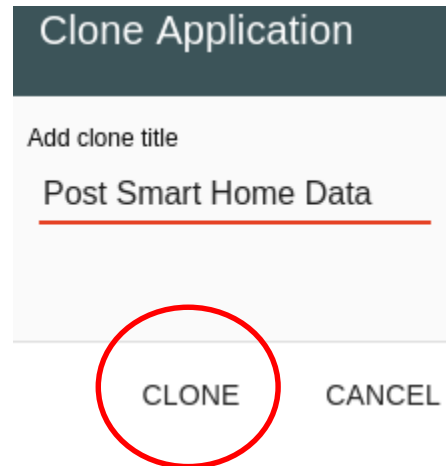


- In the Projects window, click on “Options” -> “Clone”.



Clone HTTP Post App (2)

- Change the application name to “Post Smart Home Data”. Click on “clone”.



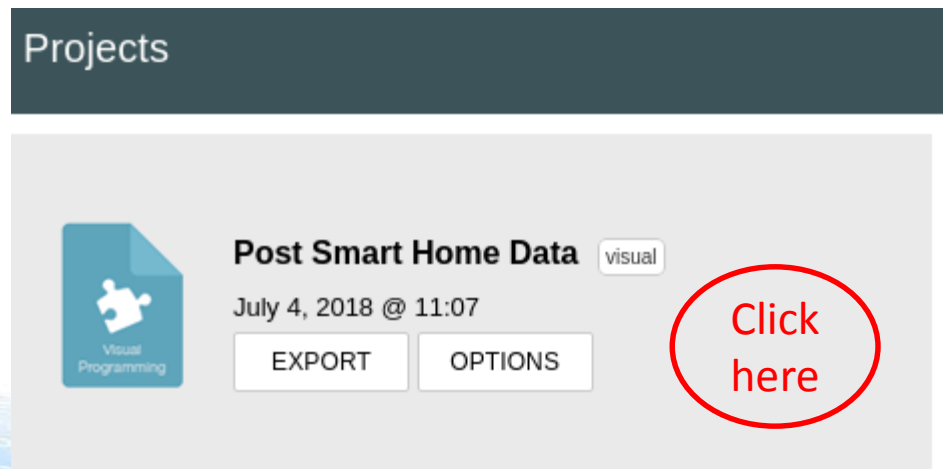
Clone Application

Add clone title


Post Smart Home Data

CLONE CANCEL

- Open the application.



Projects

 **Post Smart Home Data** visual

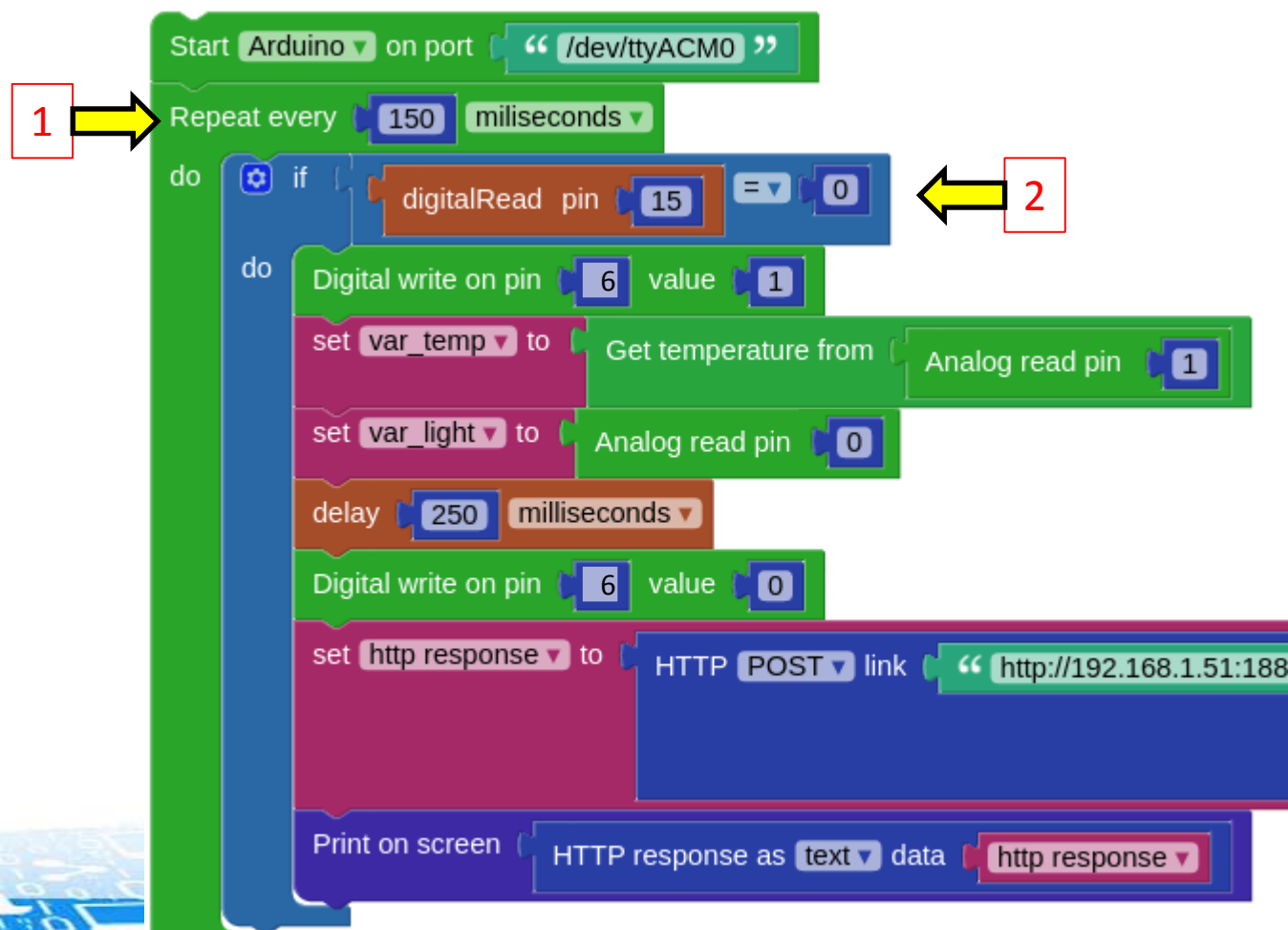
July 4, 2018 @ 11:07

EXPORT OPTIONS

Click here

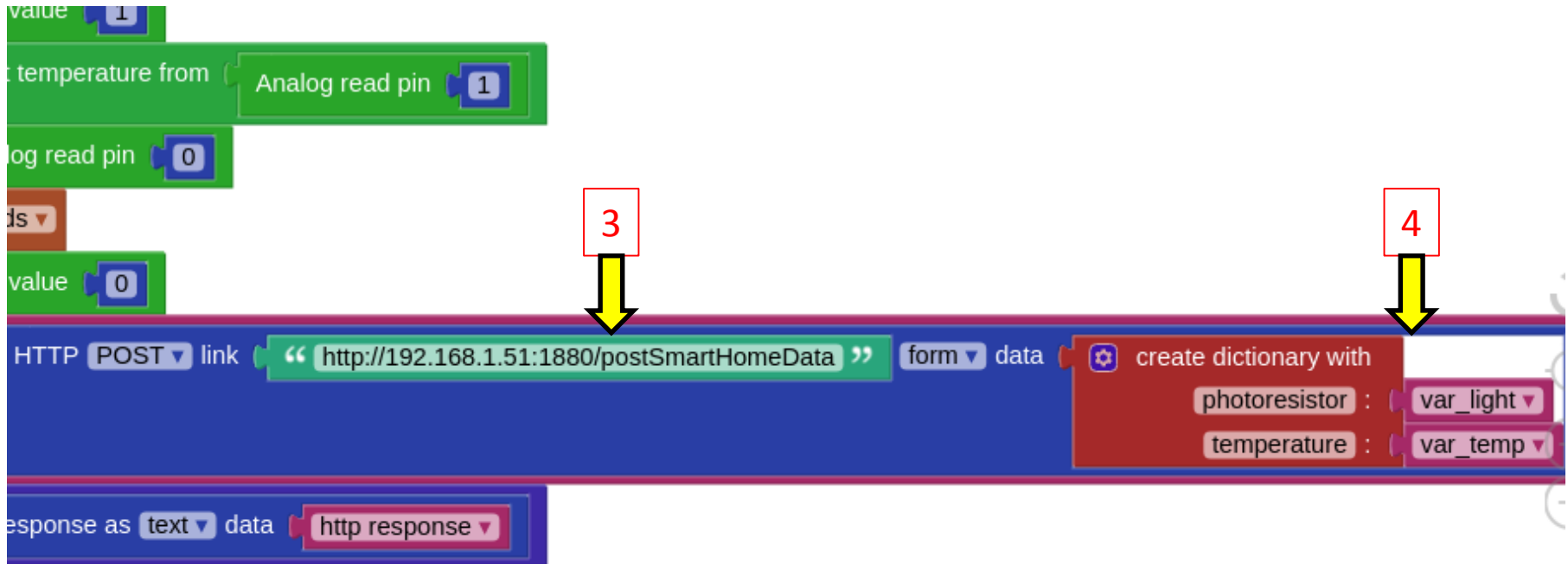
Modify the blocks (1)

1. Change time to 150 milliseconds.
2. Use the switch button to trigger the action (Put previous blocks inside the IF block).



Modify the blocks (2)

3. Set “link” to: <http://YOUR.VM.IP.ADDRESS:1880/postSmartHomeData>.
4. Change dictionary to send “photoresistor”: var_light, “temperature”: var_temp.
5. Run your application now!



The image shows a Scratch code block for an HTTP POST request. The block is divided into several sections: a dropdown menu for the HTTP method (set to POST), a text field for the link, a dropdown menu for the form type (set to data), and a 'create dictionary with' block. The link field contains the text "http://192.168.1.51:1880/postSmartHomeData". The 'create dictionary with' block has two entries: "photoresistor" with the value "var_light" and "temperature" with the value "var_temp". Below the main block is a 'response as' block with a dropdown menu set to 'text' and a label 'http response'. Two yellow arrows with red boxes containing the numbers 3 and 4 point to the link field and the 'create dictionary with' block respectively.

value 1

temperature from Analog read pin 1

log read pin 0

ds

value 0

3

4

HTTP POST link “http://192.168.1.51:1880/postSmartHomeData” form data create dictionary with

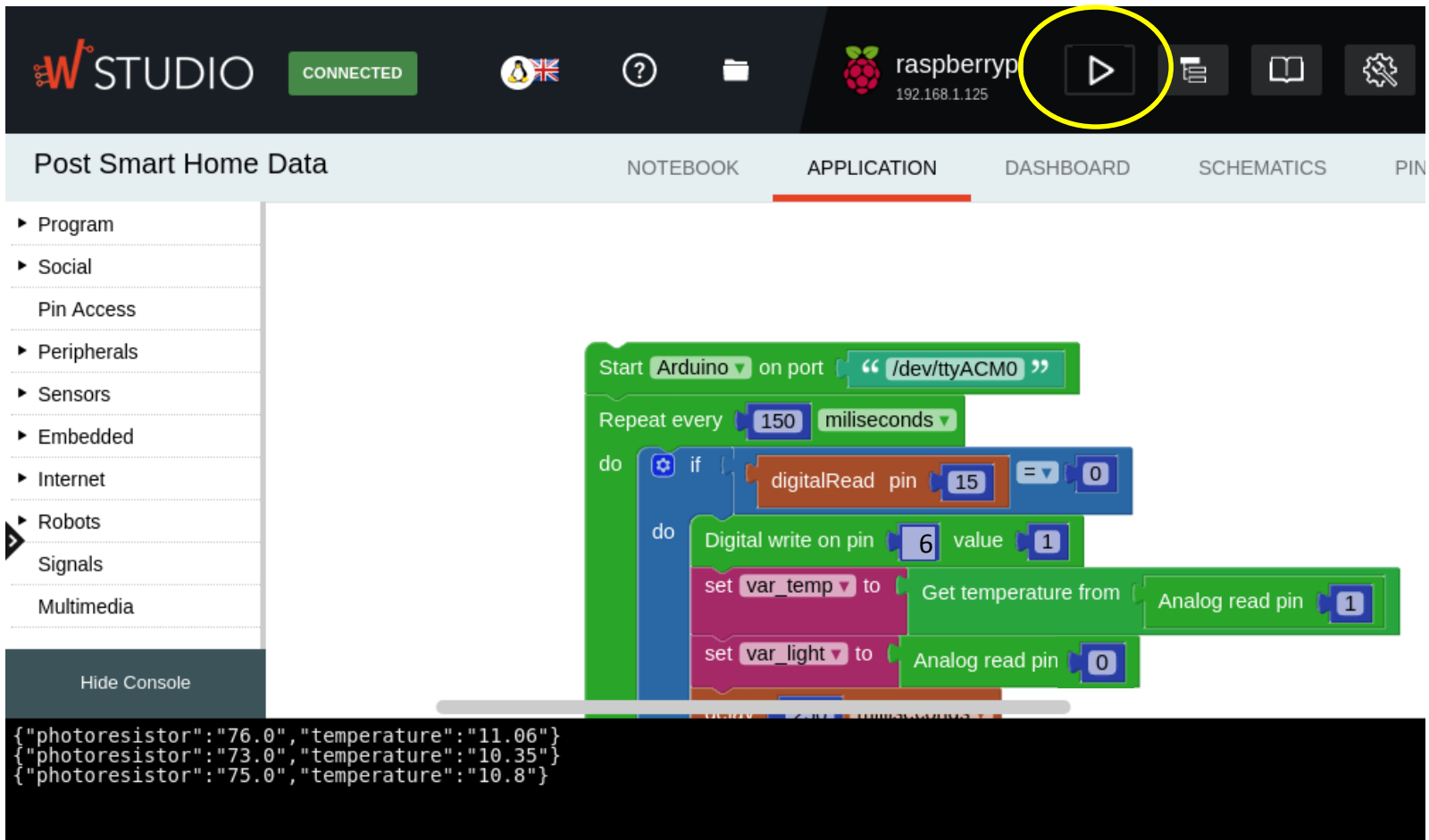
photoresistor : var_light

temperature : var_temp

response as text data http response

Run the application

Push the switch button in your breadboard several times.



W STUDIO CONNECTED

raspberrypi 192.168.1.125

Post Smart Home Data

NOTEBOOK APPLICATION DASHBOARD SCHEMATICS PIN

Program

Social

Pin Access

Peripherals

Sensors

Embedded

Internet

Robots

Signals

Multimedia

Hide Console

```
{
  "photoresistor": "76.0",
  "temperature": "11.06"
}
{
  "photoresistor": "73.0",
  "temperature": "10.35"
}
{
  "photoresistor": "75.0",
  "temperature": "10.8"
}
```

Verify the data is received

Use the OM2M GUI to check if the data is saved correctly.

<http://localhost:8080/~mn-cse-pi/cin-911179417>

– mn-name-pi

– acp_admin

– SMART_HOME

– DATA

– cin_535384691

– cin_710437371

– cin_588005961

– cin_526274670

– cin_181429149

– cin_58870277

– cin_710611188

– cin_22856444

– cin_624882041

– cin_191592019

– cin_345602423

Attribute	Value
rn	cin_911179417
ty	4
ri	/mn-cse-pi/cin-911179417
pi	/mn-cse-pi/cnt-103641824
ct	20180704T091831
lt	20180704T091831
st	0
cnf	message
cs	129
con	

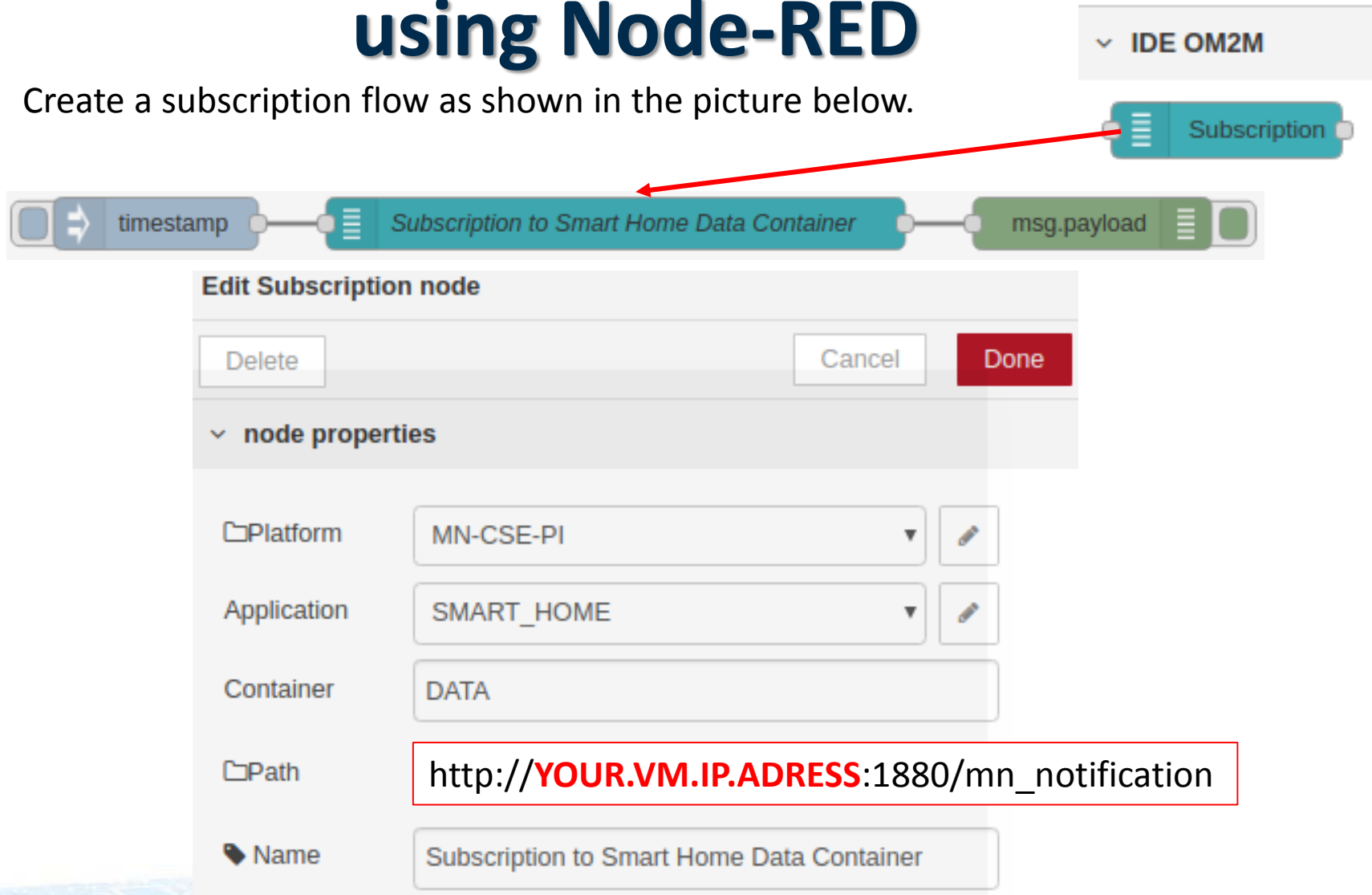
Attribute	Value
photoresistor	75.0
temperature	10.8

CHECKPOINT 3!

SUBSCRIBING TO SMART HOME DATA

Creating a subscription to a MN-AE using Node-RED

Create a subscription flow as shown in the picture below.



The image displays a Node-RED flow and the configuration for a 'Subscription' node. The flow consists of three nodes: a 'timestamp' node, a 'Subscription to Smart Home Data Container' node, and a 'msg.payload' node. A red arrow points from the 'Subscription' node in the top right palette to the 'Subscription to Smart Home Data Container' node in the flow.

The configuration for the 'Subscription' node is shown in the 'Edit Subscription node' dialog:

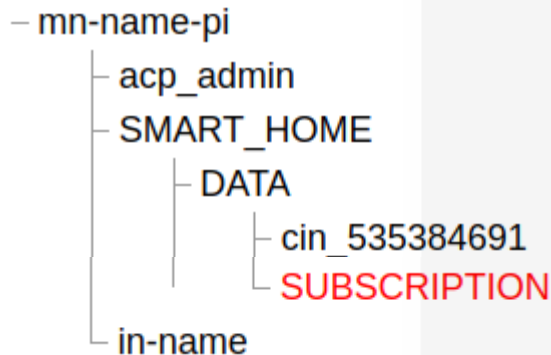
- Platform:** MN-CSE-PI
- Application:** SMART_HOME
- Container:** DATA
- Path:** http://YOUR.VM.IP.ADRESS:1880/mn_notification
- Name:** Subscription to Smart Home Data Container

Creating a subscription to a MN-AE using Node-RED

Deploy the flow and trigger it. Verify the subscription object was created using the OM2M GUI.

OM2M CSE Resource Tree

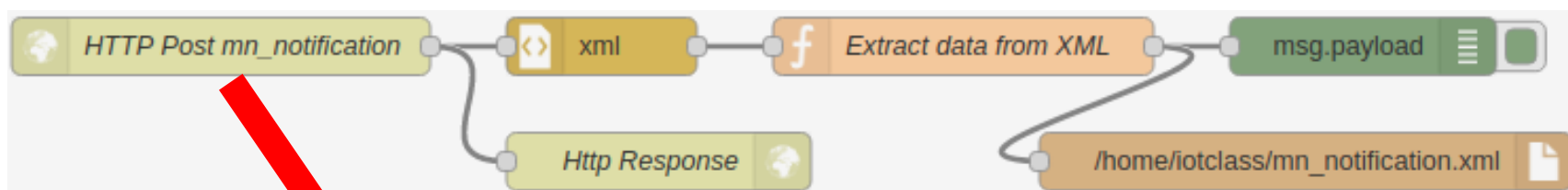
<http://localhost:8080/~mn-cse-pi/cin-324109828>



Attribute	Value
rn	SUBSCRIPTION
ty	23
ri	/mn-cse-pi/sub-395011938
pi	/mn-cse-pi/cnt-103641824
ct	20180704T075839
lt	20180704T075839
acpi	<div>AccessControlPolicyIDs</div> <div>/mn-cse-pi/acp-766302963</div>
nu	<ul style="list-style-type: none"> http://192.168.1.51:1880/mn_notification
nct	2

Creating a subscription to a MN-AE using Node-RED

Create a web service to receive the notification.



Edit http in node

Delete Cancel Done

▼ node properties

Method POST ▼

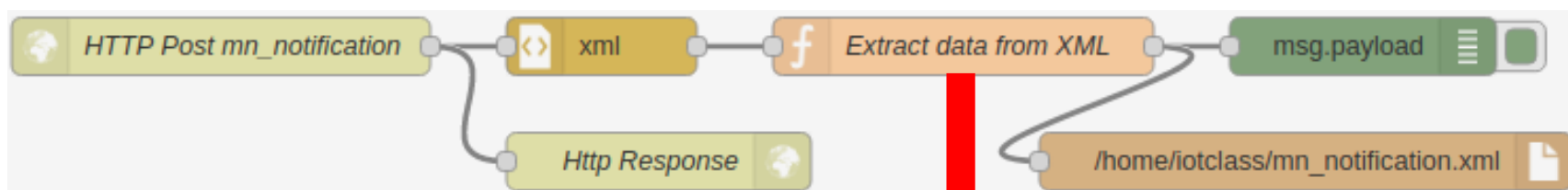
☐ Accept file uploads?

URL /mn_notification

Name HTTP Post mn_notification

Creating a subscription to a MN-AE using Node-RED

Create a web service to receive the notification.



CODE (Copy & Paste in Node Red)

```
var notification =  
msg.payload['m2m:sgn'];  
var nev = notification['nev'][0];  
var rep = nev['rep'][0];  
var con1 = rep['m2m:cin'][0];  
var con = con1['con'][0];  
msg.payload = con;  
return msg;
```

Edit function node

Delete

Cancel

Done

node properties

Name

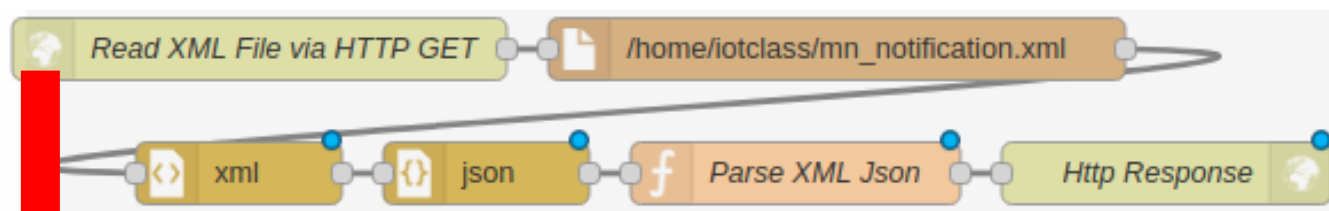
Extract data from XML

Function

```
1 var notification = msg.payload['m2m:sgn'];  
2 var nev = notification['nev'][0];  
3 var rep = nev['rep'][0];  
4 var con1 = rep['m2m:cin'][0];  
5 var con = con1['con'][0];  
6 msg.payload = con;  
7 return msg;
```

Creating a subscription to a MN-AE using Node-RED

Create an additional HTTP GET Web Service to extract the values of photoresistor and temperature from the notification message.



Edit http in node

Delete Cancel Done

▼ node properties

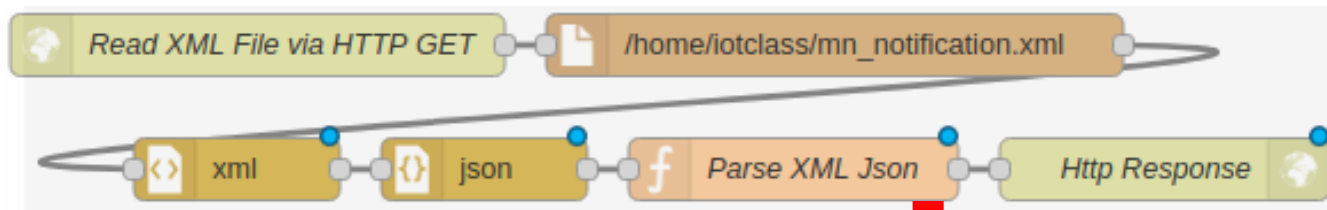
Method GET

URL /mn_notification

Name Read XML File via HTTP GET

Creating a subscription to a MN-AE using Node-RED

Create an additional HTTP GET Web Service to extract the values of photoresistor and temperature from the notification message.



CODE (Copy & Paste in Node Red)

```
var data = JSON.parse(msg.payload);
var object1 = data.obj;
var str1 = object1.str;
var photores = str1[0];
var photores_$ = photores.$;
var photoresistor = photores_$.val;
var temperatur = str1[1];
var temperatur_$ = temperatur.$;
var temperature = temperatur_$.val;
msg.payload = {
  "photoresistor":photoresistor,
  "temperature": temperature
};
return msg;
```

Name

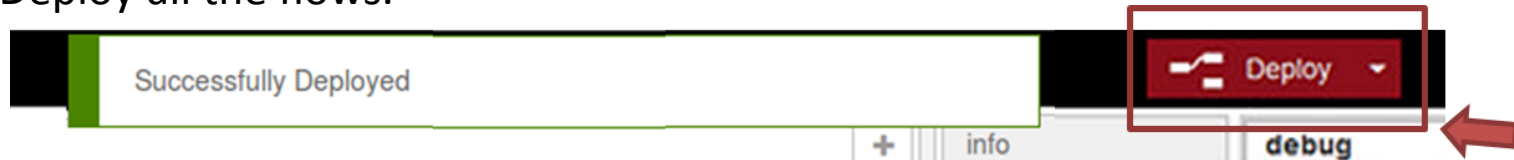
Parse XML Json|

Function

```
1 var data = JSON.parse(msg.payload);
2 var object1 = data.obj;
3 var str1 = object1.str;
4
5 var photores = str1[0];
6 var photores_$ = photores.$;
7 var photoresistor = photores_$.val;
8
9 var temperatur = str1[1];
10 var temperatur_$ = temperatur.$;
11 var temperature = temperatur_$.val;
12
13 msg.payload = {
14   "photoresistor":photoresistor,
15   "temperature": temperature
16 };
17 return msg;
```

Creating a subscription to a MN-AE using Node-RED

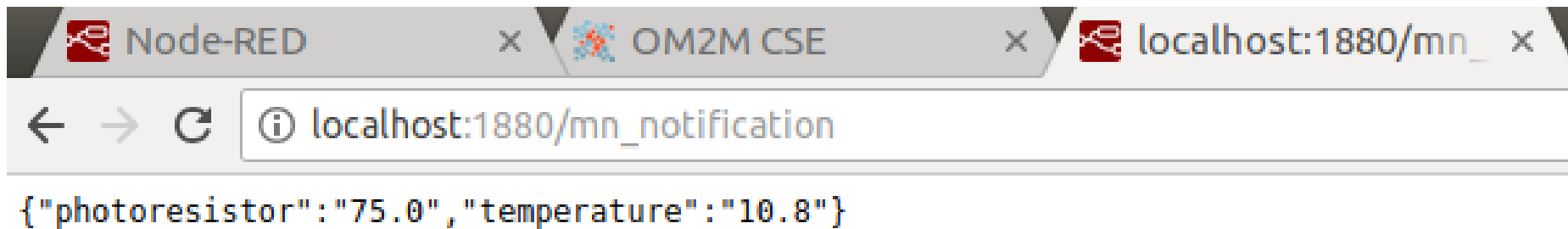
Deploy all the flows.



Push the switch button in your breadboard.

Open the url of the Read XML web service.

http://YOUR.VM.IP.ADDRESS:1880/mn_notification



CHECKPOINT 4!