



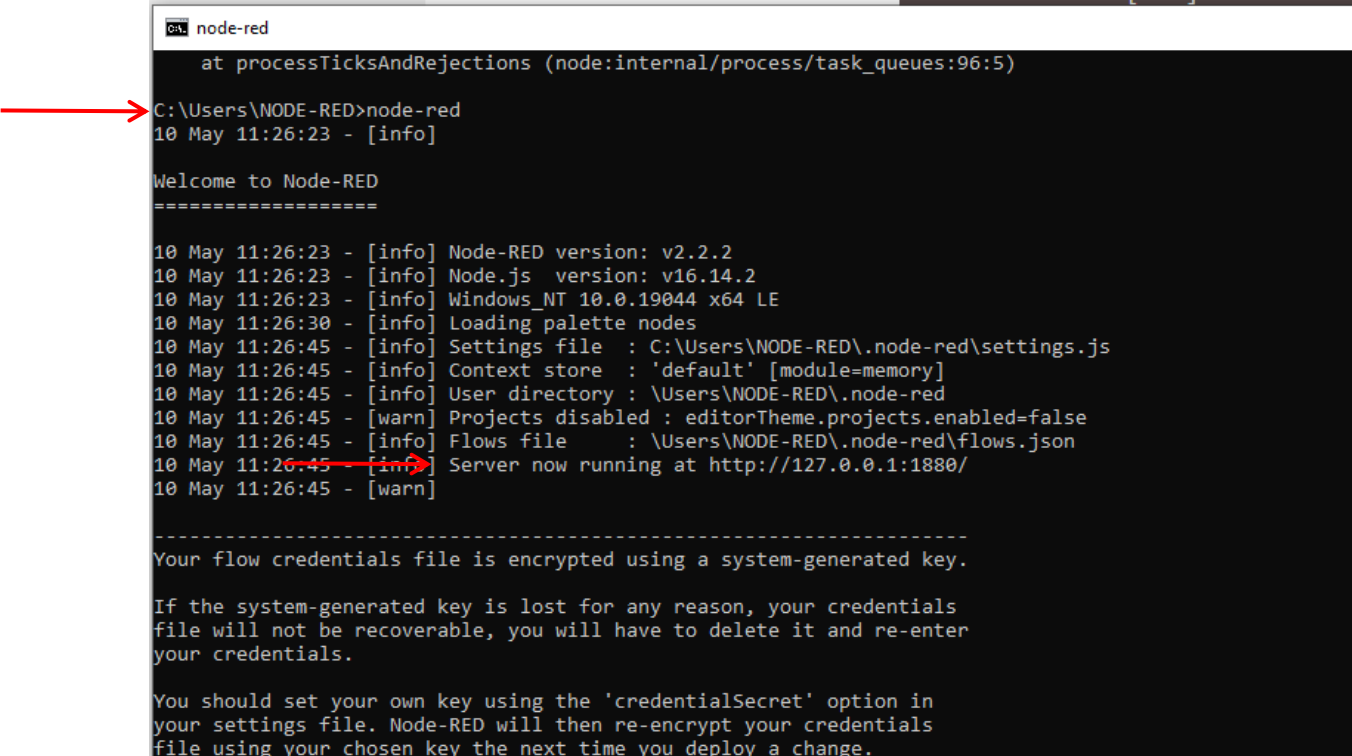
**EDC,PR,UB_ALRAM
CALCULATION
NODE-RED PROJECT
MANUAL**

MALAVIKA K.V



NODE-RED INSTALLATION SETUP

1. First need to install node-red using [Running Node-RED locally : Node-RED \(nodered.org\)](#)
2. Once installed as a global module you can use the node-red command to start Node-RED in your terminal. You can use Ctrl-C or close the terminal window to stop Node-RED.



```
node-red
at processTicksAndRejections (node:internal/process/task_queues:96:5)
C:\Users\NODE-RED>node-red
10 May 11:26:23 - [info]

Welcome to Node-RED
=====

10 May 11:26:23 - [info] Node-RED version: v2.2.2
10 May 11:26:23 - [info] Node.js version: v16.14.2
10 May 11:26:23 - [info] Windows_NT 10.0.19044 x64 LE
10 May 11:26:30 - [info] Loading palette nodes
10 May 11:26:45 - [info] Settings file : C:\Users\NODE-RED\.node-red\settings.js
10 May 11:26:45 - [info] Context store : 'default' [module=memory]
10 May 11:26:45 - [info] User directory : \Users\NODE-RED\.node-red
10 May 11:26:45 - [warn] Projects disabled : editorTheme.projects.enabled=false
10 May 11:26:45 - [info] Flows file : \Users\NODE-RED\.node-red\flows.json
10 May 11:26:45 - [info] Server now running at http://127.0.0.1:1880/
10 May 11:26:45 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

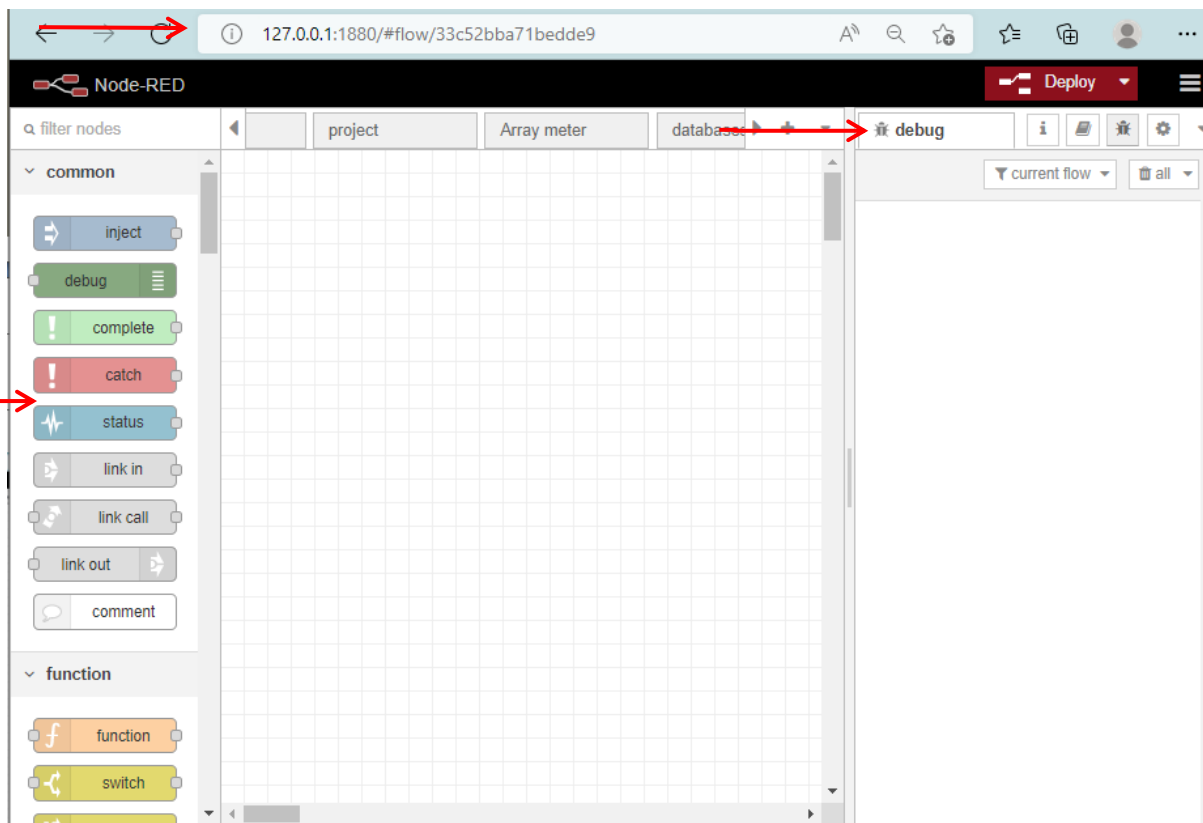
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
```

3. Installation using docker command:

```
# docker run -it -p 1880:1880 -v node_red_data:/data --name mynodered
nodered/node-red
```

4. Access the editor with Node-RED [running](#), open the editor in a web browser. If you are using a browser on the same computer that is running Node-RED, you can access it with the url: <http://localhost:1880>. If you are using a browser on another computer, you will need to use the ip address of the computer running Node-RED: <http://<ip-address>:1880>.
5. A node-red workspace will open on the left side of workspace there will nodes to work on and in right side debug window to show outputs.



EDC PROJECT REQUIREMENTS

STEP 1: Calculate Energy dc by using IDC and UDC values from all blocks save these calculated values in new tag naming as “EDC01” and sent this to an influxdb database in every 1 min

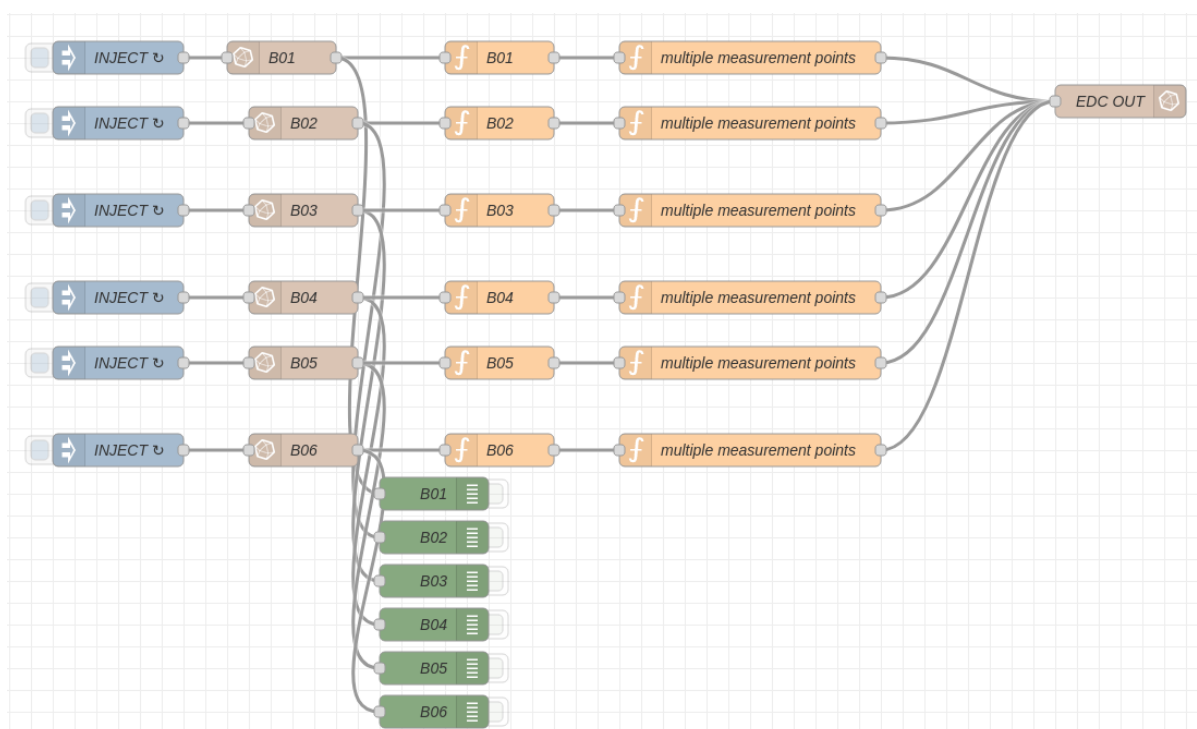
STEP 2: calculated PR calculation using the sent EDC values and give new tag naming like “PR_EDC_01” and sent this to an influxdb database in every 1 min

STEP 3: Find Mean,Max,Standard Deviation of PR_EDC values coming from influxdb database and set UB_ALARM condition according to Standard deviation and send it to influxdb database again.

NODE-RED FLOW FOR STEP 1:

STEP 1: Calculate Energy dc by using IDC and UDC values from all blocks save these calculated values in new tag naming as “EDC01” and sent this to an influxdb database in every 1 min

NODE-RED FLOW FOR STEP 1:



[There is 6 flows for 6 blocks all flows uses same function logic only b= part in influxdb query is different for all 6]

Inject Node:

- 1 The inject plus node allows you to inject messages into a flow, either by clicking the button on the node, or setting a time interval between injects.
- 2 Select the newly added Inject node to see information about its properties and a description of what it does in the [Information sidebar pane](#).
- 3 Drag and drop the node from nodes to workspace.

Edit inject node

Delete

Cancel

Done

Properties

Name

INJECT

+ add

inject now

☒ Inject once after

0.1

seconds, then

Repeat

interval

▼

every

1

▲▼

minutes

▼

☐ Enabled

Here we are setting up the triggering time to every 1 min repeat that will trigger the corresponding flow in the set interval

Influxdb in node:

Click on the pencil icon to add details of the new server

Add Server configuration as follows

Edit influxdb in node > Edit influxdb node

Delete Cancel Update

Properties

Name

Version 1.x

Host iplon-pvindia.com Port 3008

Database SIRICILLA_30MW

Username

Password

☐ Enable secure (SSL/TLS) connection

Give query in the query box

Query used in this flow is

```
SELECT last("value") FROM "scaback_csv" WHERE ("b" = 'B01' AND "f" = ~  
/.*SCB*/.) AND time >= now() - 10m GROUP BY time(1m), "b", "d", "f"  
fill(none);
```

```
SELECT last("value") FROM "scaback_csv" WHERE ("b" = 'B01' AND "f" =  
'UDC') AND time >= now() - 10m GROUP BY time(1m), "b", "d", "f"  
fill(none);
```

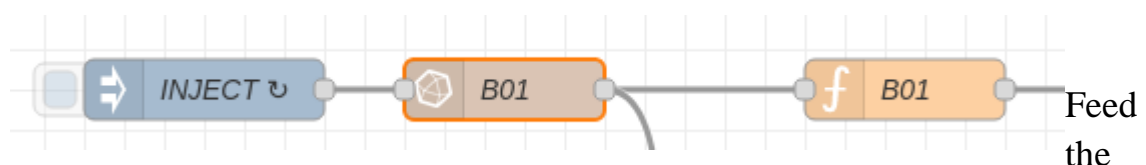
So that we will get output of 2 array

1st array contain details of field IDC from a block all devices in last 10 min

2nd array contain details of field UDC from a block all devices in last 10 min

Function node: The Function node allows JavaScript code to be run against the messages that are passed through it.

The message is passed in as an object called **msg**. By convention it will have a **msg.payload** property containing the body of the message. Other nodes may attach their own properties to the message, and they should be described in their documentation.



javascript code on the message box

Edit function node

Delete Cancel Done

Properties

Name B01

Setup On Start **On Message** On Stop

```

1  let plantObj
2
3  plantObj = []
4
5  for (var j =0; j < msg.payload[1].length;j++)
6    for (var i =0; i < msg.payload[0].length;i++)
7    {
8      var itime= new Date(msg.payload[0][i].time).getTime()
9      var utable= new Date(msg.payload[1][j].time).getTime()
10     var idvice=msg.payload[0][i].d
11     var udevice=msg.payload[1][j].d
12
13     if ((itime==utable)&&(idvice==udevice))
14
15
16     {
17       var time=utable
18       var IDC=msg.payload[0][i].last;
19       var UDC= msg.payload[1][j].last;
20
21       var PDC=IDC*UDC
22       var EDC = (PDC*60)/(60*60*1000);
23
24
25       var field_idc= msg.payload[0][i].f;
26       var field_udc= msg.payload[1][j].f;
27       //node.send(obj);
28       var obj={
29         TIME:new Date(time),
30         IDC,
31         UDC,
32         edc:EDC,
  
```

```
var plantObj = []
```

```

for (var j =0; j < msg.payload[1].length;j++)
for (var i =0; i < msg.payload[0].length;i++)
{
var itime= new Date(msg.payload[0][i].time).getTime();
var utable= new Date(msg.payload[1][j].time).getTime();
  
```



```
var ideoice=msg.payload[0][i].d
var udevice=msg.payload[1][j].d
if ((itime==utime)&&(ideoice==udevice))

{
var time=utime
var IDC=msg.payload[0][i].last;
var UDC= msg.payload[1][j].last;
var PDC=IDC*UDC
var EDC = (PDC*60)/(60*60*1000);
var field_idc= msg.payload[0][i].f;
var field_udc= msg.payload[1][j].f;
//node.send(obj);
var obj={
TIME:new Date(time),
IDC,
UDC,
edc:EDC,
f:["EDC01",
"EDC10",
"EDC11",
"EDC12",
"EDC02",
"EDC03",
"EDC04",
"EDC05",
"EDC06",
"EDC07",
"EDC08",
"EDC09"
],
udevice,
ideoice,
field_idc

}
plantObj.push(obj)
}
}

msg.payload =plantObj

return msg;
```

This javascript code is used to check every message in the array of output getting from influxdb in node using 2 for loop function because we here we have 2 array output and calculate PDC(power) by equation $P=I*V$ in this case $PDC=IDC*UDC$ and EDC(energy for a minute) by using equation $E=(P*60)/(60*60*1000)$ in this case

$EDC = (PDC*60)/(60*60*1000)$; before calculation we have to check the timestamp and device of both array if both are equal only we have to do calculation

create one array inside the loop msg.object to give tags to corresponding EDC values

Function for structure output in influxdb line-protocol

if we need to inject fields, tags and timestamp structure to influxdb we need to structure it in that way for that we use a logic like this

```
var j=0;
for (var i=0; i < msg.payload.length ;i++)
{
  var f=msg.payload[i].f

  var field=msg.payload[i].f[j];
  j=j+1;
  if(j==f.length)
  {j=0;}
  var obj={ };
  obj.payload=
  [
    {
      measurement: "energydc_test",
      fields: {
        value:msg.payload[i].edc
      },
      tags:{
        f:field,
        d:msg.payload[i].device
      },
      timestamp:msg.payload[i].TIME
    }
  ];
  node.send(obj);
```

```
}  
return null;
```

Here we using for loop to iterate the array output then inside the loop itself we are again iterating field array[f] using other loop then setting output object in field,tag,timestamp structure then sending the output as single message object after each iteration use node.send(array) for single message output.

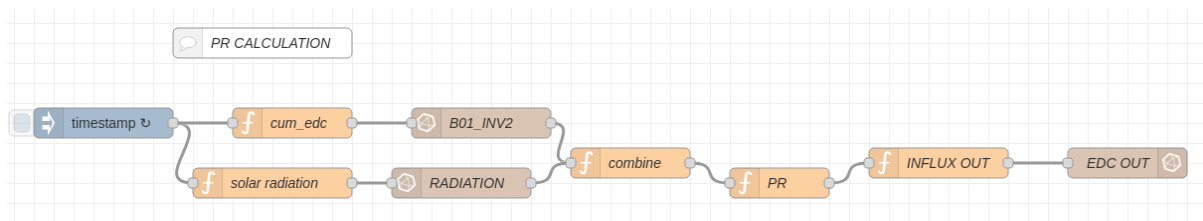
Influxdb batch node:

inside this node just setup server configuration where we want to send this data

The screenshot shows the 'Edit influx batch node' configuration window. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below these is a 'Properties' section with a gear icon. The 'Name' field is set to 'EDC OUT'. The 'Server' field is a dropdown menu showing '[v1.x] iplon-pvindia.com:30086/EDC' with a pencil icon to its right. There is an unchecked checkbox for 'Advanced Query Options'. At the bottom, a yellow tip box states: 'Tip: If no retention policy is specified, autogen will be assumed.'

NODE-RED FLOW FOR STEP-2

STEP 2: calculated PR calculation using the sent EDC values and give new tag naming like “PR_EDC_01” and sent this to an influxdb database in every 1 min



Inject Node:

1. The inject plus node allows you to inject messages into a flow, either by clicking the button on the node, or setting a time interval between injects.
2. Select the newly added Inject node to see information about its properties and a description of what it does in the [Information sidebar pane](#).
3. Drag and drop the node from nodes to workspace.
4. Here we are using one timestamp to trigger both flow

Edit inject node

Delete

Cancel

Done

Properties

Name

INJECT

+ add

inject now

☒ Inject once after

0.1

seconds, then

Repeat

interval

▼

every

1

▲▼

minutes

▼

☐ Enabled

Here we are setting up the triggering time to every 1 min repeat that will trigger the corresponding flow in the set interval

Function node : Here we used function node to give msg.query output to influxdb in node. If we need to query for a variable start and stop time we can use javascript logic to set up start and stop time here we need midnight timestamp[12.00AM] of that day as start time and last timestamp of the day[11.59PM] time as end timestamp.

```
var $today = new Date();
```

```
var d = new Date($today);
```

```
d.setDate($today.getDate());
```

```
d.setHours(0,0,0,0);
```

```
var $tommarow = new Date();
```

```
var d1 = new Date($tommarow);
```

```
d1.setDate($tommarow.getDate());
```

```
d1.setHours(23,59,0,0);
```

```
var tstart=d.getTime();
```

```
var tend= d1.getTime();
```

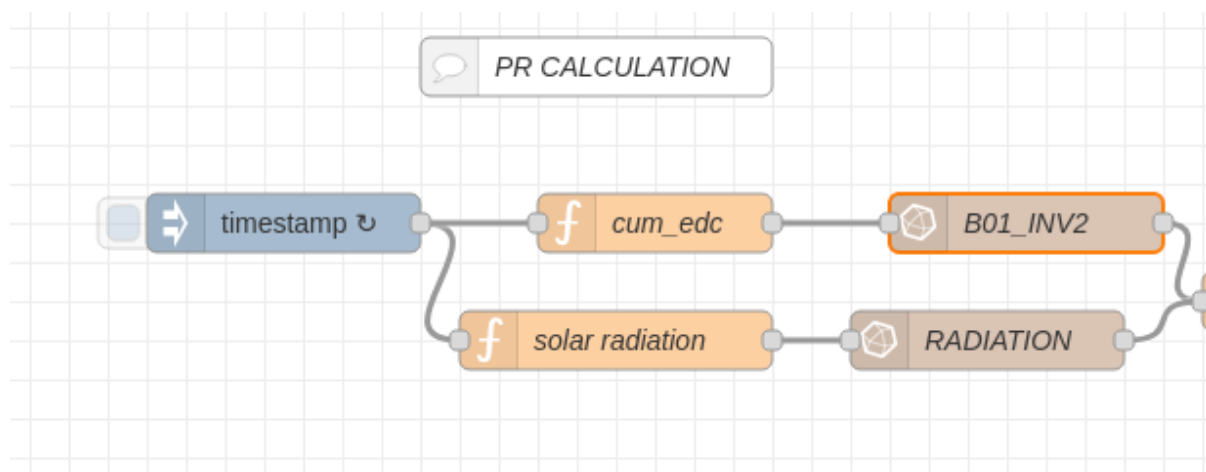
```
var q = "SELECT cumulative_sum(sum(value)) FROM energydc_test WHERE (d = 'B01_INV2' AND f !~  
/. *PR_EDC*/ AND f != 'UB_ALARM') AND time >= " + tstart + "ms and time <= " + tend + "ms GROUP  
BY time(5m),d,f fill(0);";
```

```
msg.query = q
```

```
msg.topic="cum";
```

```
return msg;
```

set msg.topic because we are going to combine both influxdb output in coming node-red



Parellaly one more function node also used for collectind solar radiation data using variable start time and stop timestamp

```
var $today = new Date();

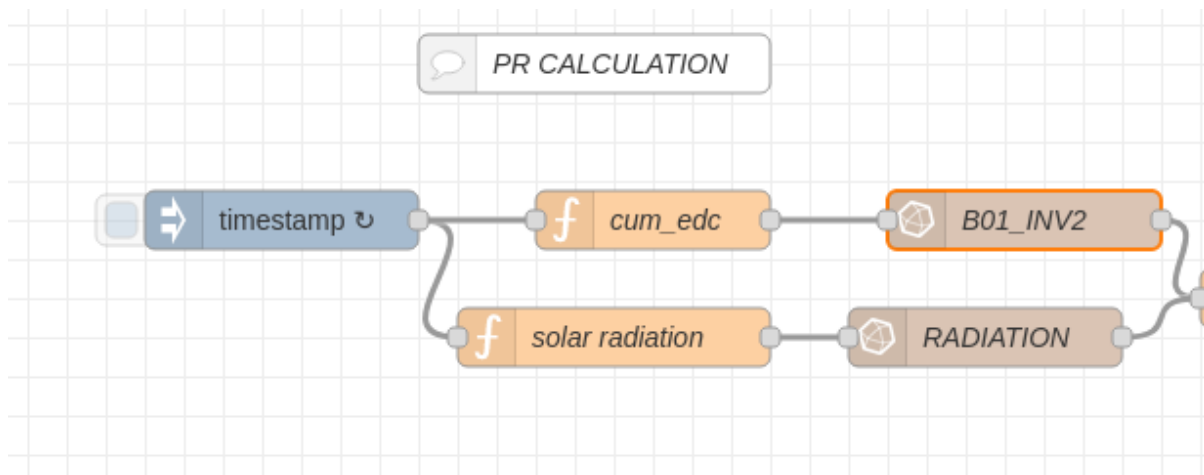
var d = new Date($today);
d.setDate($today.getDate());
d.setHours(0,0,0,0);
var $tommarow = new Date();
var d1 = new Date($tommarow);
d1.setDate($tommarow.getDate());
d1.setHours(23,59,0,0);
var tstart=d.getTime();
var tend= d1.getTime();
var q = "SELECT last(value) FROM scaback_csv WHERE (d = 'CR_EM01' AND f =
'RADIATION_CUM') AND time >= "+ tstart +"ms and time <= "+ tend + "ms GROUP BY time(5m),d,f
fill(0);";
msg.query = q
msg.topic="radiation";

return msg;
```

Influxdb in node ([node-red-contrib-influxdb \(node\)](#) - [Node-RED \(nodered.org\)](#)):

To access this node we have to install the node package(node-red-contrib-influxdb)from manage palette feature(click 3 lines symbol in the top right corner in the node-red workspace)

Nodes to query data from an influxdb time series database. Supports InfluxDb versions 1.x to 2.0.



double click on the influx db node

keep the query box blank if you are giving msg.query through a function node

Edit influxdb in node

Delete

Cancel

Done

Properties

Name

B01_INV2

Server

[v1.x] iplon-pvindia.com:30086/EDC

☐ Raw Output

☐ Advanced Query Options

Query

1

Tip: If no query is set, ensure **msg.query** contains a query.

Then click on the pencil icon to add details of the new server

Edit influxdb in node > Edit influxdb node

Delete Cancel Update

Properties

Name

Version 1.x

Host iplon-pvindia.com Port 3008

Database EDC

Username

Password

☐ Enable secure (SSL/TLS) connection

Parellaly other function node input is given to second influxdb out node which taking data from different database.

Function node for combine outputs from both flows

```
context.data = context.data || { };
switch(msg.topic){
case "cum":
context.data.cum = msg.payload;
msg = null;
break;
case "radiation":
context.data.radiation = msg.payload;
msg = null;
break;
}
if(context.data.cum != null && context.data.radiation != null){
var data_combine = {
```

```
"cum": context.data.cum,  
"radiation": context.data.radiation  
}  
context.data = null  
return {payload: data_combine};  
}
```

Inside function we are writing logic to combine outputs from both flows and merge it to one array of output

function node for calculating PR:

```
let plantObj  
  
plantObj = []  
  
for (var i =0; i < msg.payload.cum.length;i++)  
for( var j =0; j < msg.payload.radiation.length;j++)  
  
{  
var edctime= new Date(msg.payload.cum[i].time).getTime();  
var soltime= new Date(msg.payload.radiation[j].time).getTime();  
if (edctime==soltime)  
{  
var CUM_EDC=msg.payload.cum[i].cumulative_sum  
var CUM_RADIATION=msg.payload.radiation[j].last  
var pr = "PR" + "_" + msg.payload.cum[i].f;  
var PR;  
if (CUM_RADIATION==0)  
{  
PR =0;  
}  
else  
{  
PR = ((CUM_EDC)/(1717.2 *CUM_RADIATION))*100;  
}  
var obj={  
edctime,  
CUM_EDC,  
CUM_RADIATION,  
pr,  
PR,  
field:msg.payload.cum[i].f,  
device:msg.payload.cum[i].d,
```

```

}

plantObj.push(obj)
}
}
msg.payload = plantObj
return msg;

```

This javascript code is used to check every message in the array of output getting from influxdb in node using 2 for loop function because we here we have 2 array output and calculate PR (performance ratio) by equation

$$PR = ((\text{Cumulative EDC}) / (\text{DC Capacity of the device} * \text{cumulative solar radiation})) * 1000$$

in this case $PR = ((\text{CUM_EDC}) / (1717.2 * \text{CUM_RADIATION})) * 100;$

before calculation we have to check the timestamp of both array if both are equal only we have to do calculation and setting PR=0 if Cum radiation=0 using if condition

adding “PR” string with incoming field name(EDC) to create tag for PR_EDC values

Function for structure output in influxdb line-protocol

if we need to inject fields, tags and timestamp structure to influxdb we need to structure it in that way for that we use a logic like this

```

for (var i = 0; i < msg.payload.length; i++)
{
  var obj = {};
  obj.payload = [
    {
      measurement: "energydc_test",
      fields: {
        value: msg.payload[i].PR
      },
      tags: {
        f: msg.payload[i].pr,
        d: msg.payload[i].device
      },
      timestamp: new Date(msg.payload[i].edctime)
    }
  ]
}

```

```
}  
];  
  
node.send(obj);  
}  
  
return null;
```

setting output object in field,tag,timestamp structure then sending the output as single message object after each iteration use node.send(array) for single message output.

Influxdb-batch node

inside this node just setup server configuration where we want to send this data.

Edit influx batch node

Delete Cancel Done

Properties

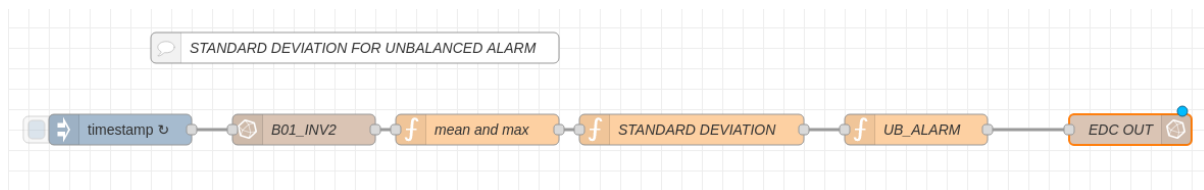
Name EDC OUT

Server [v1.x] iplon-pvindia.com:30086/EDC

☐ Advanced Query Options

Tip: If no retention policy is specified, **autogen** will be assumed.

NODE-RED FLOW FOR STEP 3



STEP 3: Find Mean,Max,Standard Deviation of PR_EDC values coming from influxdb database and set UB_ALARM condition according to Standard deviation and send it to influxdb database again.

Inject Node:

- The inject plus node allows you to inject messages into a flow, either by clicking the button on the node, or setting a time interval between injects.
- Select the newly added Inject node to see information about its properties and a description of what it does in the [Information sidebar pane](#).
- Drag and drop the node from nodes to workspace.

Edit inject node

Delete

Cancel

Done

Properties

Name

Name

≡

msg. payload

=

▼

timestamp

×

≡

msg. topic

=

▼

a_z

×

+ add

inject now

☒ Inject once after

0.1

seconds, then

Repeat

interval

▼

every

5

↑

↓

minutes

▼

☐ Enabled

Influxdb in node:

Click on the pencil icon to add details of the new server

Add Server configuration as follows

Edit influxdb in node > **Edit influxdb node**

Delete Cancel Update

Properties

Name

Version

Host Port

Database

Username

Password

☐ Enable secure (SSL/TLS) connection

Give query in the query box

Query used in this flow is

```
SELECT last("value"),d FROM "energydc_test" WHERE ("d" = 'B01_INV2'  
AND "f" =~ /. *PR_EDC*./) AND time >= now() - 5m GROUP BY time(5m),  
"f" fill(none)
```

Edit influxdb in node

Delete Cancel Done

Properties

Name

Server

☐ Raw Output ☐ Advanced Query Options

Query

```
1 SELECT last("value"),d FROM "energydc_t
```

So that we will get output of array contain details of field PR_EDC from a block all devices in last 5min

Function node to find mean and maximum of PR_EDC values

Here we used logic for iterating array and functions to find mean and maximum from coming PR_EDC values

let plantObj

```
plantObj = []
```

```
var grades = msg.payload
var total = 0;
var max = grades[0].last
for(var i = 0; i < grades.length; i++)
{ var time=grades[i].time
{
var RESULT;
var PR = grades[i].last
total += PR;
if (PR>max)
{max=PR}
```

```
var avg = total / grades.length;
```

```
RESULT=grades[i].f.split("PR_EDC");
var SD="SD"+RESULT[1]
```

```
var OBJ={time,avg,max,PR,
SD,
d:grades[i].d
}
```

```
plantObj.push(OBJ)
}
}
msg.payload =plantObj
```

Function node to calculate Standard Deviation of PR_EDC values using mean and max:

Here we are iterating the input array and calculating standard deviation and calculating standard deviation the equation as follows and setting unbalanced

alarm according to the condition if standard deviation(SD)<1 it is balanced so we are setting integer 0 for balance and 1 for unbalanced

let plantObj

```
plantObj = []
var array= msg.payload;
var sum=0
for(var i = 0; i < array.length; i++)
{
var AVG= array[(array.length)-1].avg
var MAX=array[(array.length)-1].max
var max_70 =0.7*MAX;
var PR=array[i].PR
var Std_deviation =Math.pow((AVG-PR),2);
sum +=Std_deviation
var SD_MEAN =sum/array.length
var SD=Math.sqrt(SD_MEAN)
var REMARKS;
if(SD<1)
{REMARKS=0} // BALENCED
else{REMARKS=1} //UNBALENCED
var sd_field =array[i].SD
var obj={
time:array[i].time,
AVG,
MAX,
PR,
max_70,
Std_deviation,
sum,
sd_field,
SD_MEAN,
SD,
d:array[i].d,
REMARKS
}
plantObj.push(obj)
}
msg.payload=plantObj;
return msg;
```

Function for struture output in influxdb line-protocol

if we need to inject fields,tags and timestamp structure to influxdb we need to structure it in that way for that we use a logic like this

```
for (var i =0; i < msg.payload.length ;i++)
{
var obj={ };
obj.payload=

[
{
measurement: "energydc_test",
fields: {
value:msg.payload[i].REMARKS
},
tags:{
f:"UB_ALARM",
d:msg.payload[i].d
},
timestamp:msg.payload[i].time
}
];

node.send(obj);
}
return null;
```

setting output object in field,tag,timestamp structure then sending the output as single message object after each iteration use node.send(array) for single message output.


Influxdb-batch node

inside this node just setup server configuration where we want to send this data.


Edit influx batch node

DeleteCancelDone

Properties


 Name

EDC OUT

 Server

[v1.x] iplon-pvindia.com:30086/EDC

▼



☐ Advanced Query Options

Tip: If no retention policy is specified, **autogen** will be assumed.