

**Diagnostic Box configuration**

**V.Vani**

**Malavika K.V**

**V1.0 22.06.2022**

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **TOPIC** | **Pg.No** |
| **1** | **IP scan conflict work flow on Node-red** | **3** |
| **2** | **Modbus read work flow on Node-red** | **6** |

## 1. Node-red Flow for IP\_Scan:

## Fig – 1.1

## **Arp** node is used to scan all the devices in the network and the output of node is shown in below figure.

## 

## Fig – 1.2

## From the output, all the live devices MAC addresses are taken and stored in MAC\_LIVE array

## 

## 

## Fig – 1.3

## A CSV file is taken, where Mac addresses of the devices present in the network are already stored as reference and taken as MAC\_STORED array

## 

## Fig – 1.4

## Above flow will run continuously with some particular time interval and check for the count of live devices.

## If any external device gets connected to the network live count goes greater than stored

## Then it will compare both **MAC\_STORED** and **MAC\_LIVE** and separate that particular MAC address of the device connected newly and store in a CSV file as shown in below Fig.

## 

## Fig – 1.5

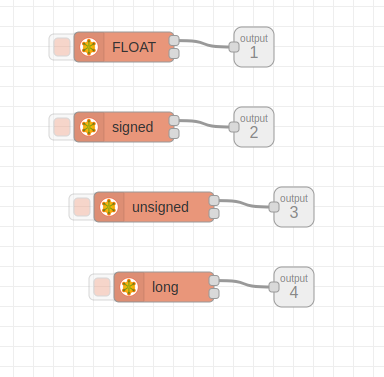
## If any device gets disconnected from the network then the count of live devices decreases and the disconnected device MAC address is saved in a CSV file as shown in below Fig.

## 

## Fig – 1.6

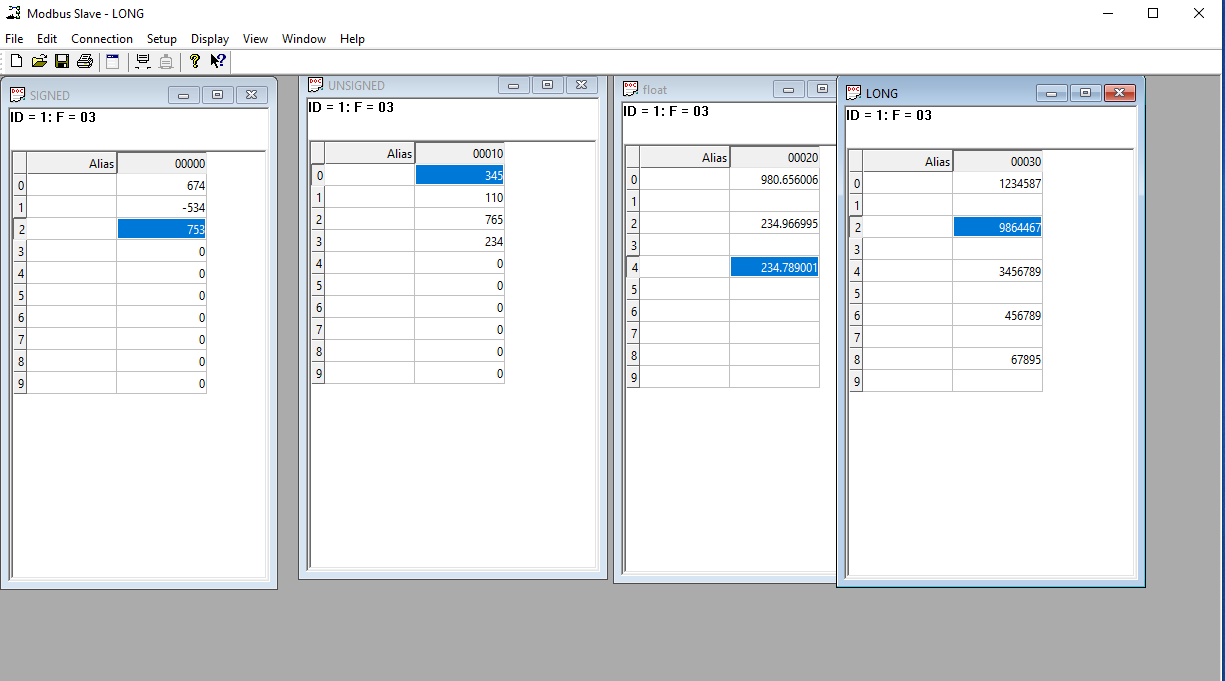
## 2. Modbus read work flow on Node-red

## 



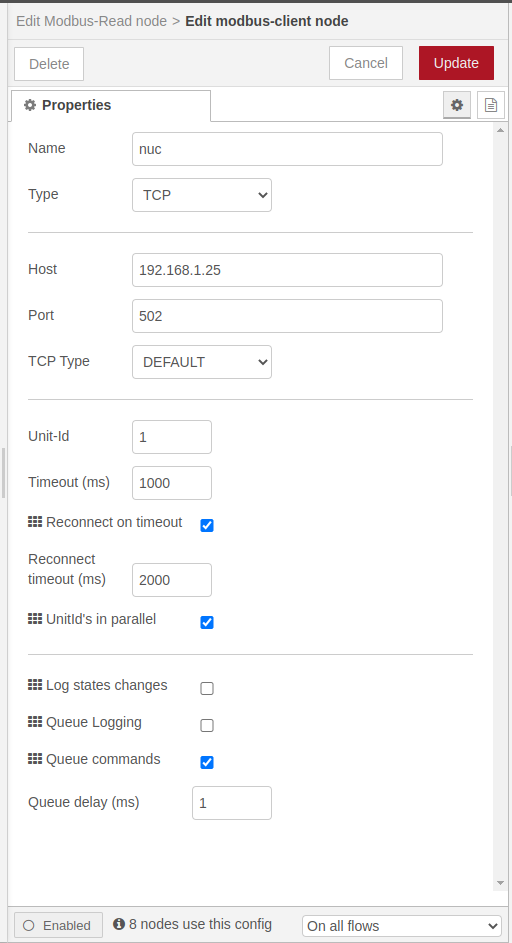
Here 4 modbus read nodes were used to read data from server for using this node first we have to install node-red-contrib-modbus node package from manage pallete.

Here i tested this flow by setting up a modbus slave in different device there feed the values in every registers as shown below.



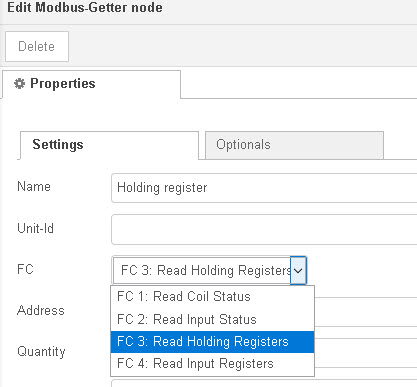
Inside all the modbus read nodes we have to give configuration like host, port ,type,

unit-id as new configuration



in those 4 modbus read nodes one is for read float values and one for signed and one for unsigned values and another for long values inside all this nodes we have give the register address,quantity(how much quantity of values we need from the particular register)and also function code.

The nodes support function codes 1,2,3 and 4 as shown below:



Both nodes have two outputs which are essentially the same. I tend to use output 2.If you pass the output to a debug node and display the complete msg object you will see that you get an output like the one shown below.



The node assumes that the register contains a 16 bit integer and if that is the case then you can use the value field to get the data. If the data represents a float or 32 bit integer you will need to use the buffer.The syntax is msg.values and msg.payload.buffer.

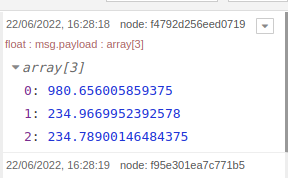
After this to integrate all the 4 modbus read as one input node i created a subflow by selecting those 4 modbus read node and > manage pallete > subflows > create subflow.and named it as IPLON MODBUS so the outputs coming from the subflow node will be as corresponding to thier allignment.

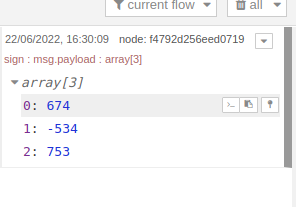
One function node is used to convert the 16bit value in float format which uses a JavaScript code which check every msg.payload <<16 and add that with next output and combine those messages in one msg.payload output.

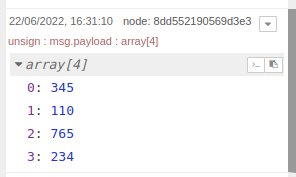
And also using float node (node-red-contrib-float) node package to show the input values in float format.



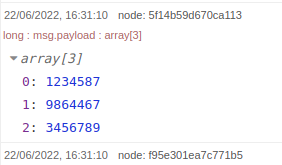
No configuration is needed inside it.



Then for converting 16 bit sign values also we use another function node having JavaScript code which will check every msg.payload which is <<16 and set every inputs as var msg0= (msg.payload[0] << 16) >> 16; and combine those messages in one msg.payload output.

Unsigned values can read and show as it as so that it doesn’t need any function node for conversion.

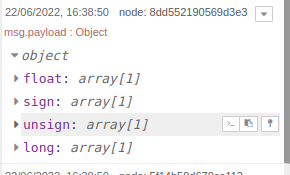
Then for converting long values also we use another function node having JavaScript code with eqations var msg0=(msg.payload[1]<<16)+msg.payload[0]; and combine all this equation as a single output.



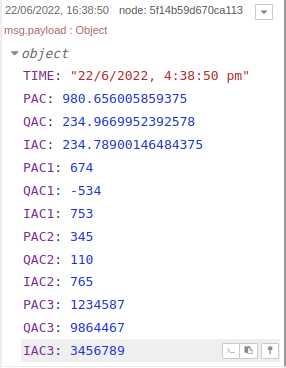
To set up output in a structured format we use function node which take input and give headers to each outputs give a structre to output.Here for all 4 outputs i used function node which has same format only header naming will differ.



To merge all the 4 function output as a single output we have to combine those with msg.topic every inputs should have different topic names and merge those with a function node.



After this give header to all output and give strutured output with the use of function node.

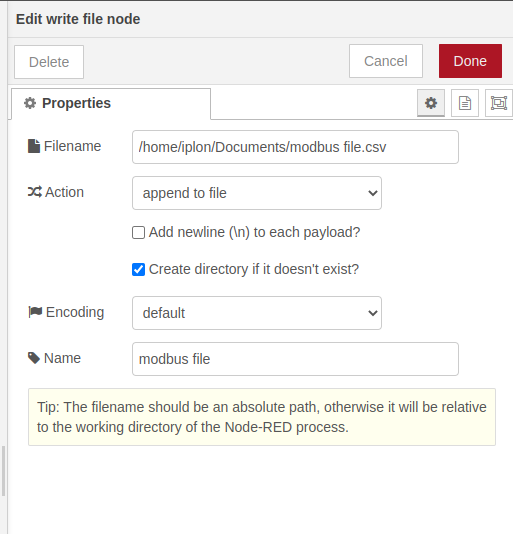


From this output is going to to flow one to create csv file and another one for influx out.

Write a to a csv file using a csv node to pass output as message per row and setting header format.

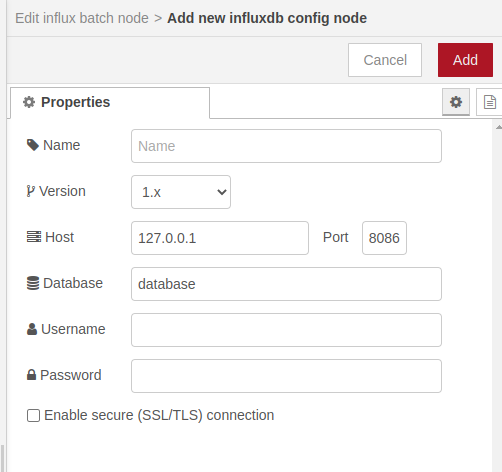


Then a write file node to feed the path where to save the file.



One more function node s using to give measurement, field,tag,timestamp before we injecting this data to influx DB

influxDB batch node (node-red-contrib-influxdb) used here to out this data to influx database for this we have to give some configuration details to this node.



Use debug nodes to take out output from each nodes.

