

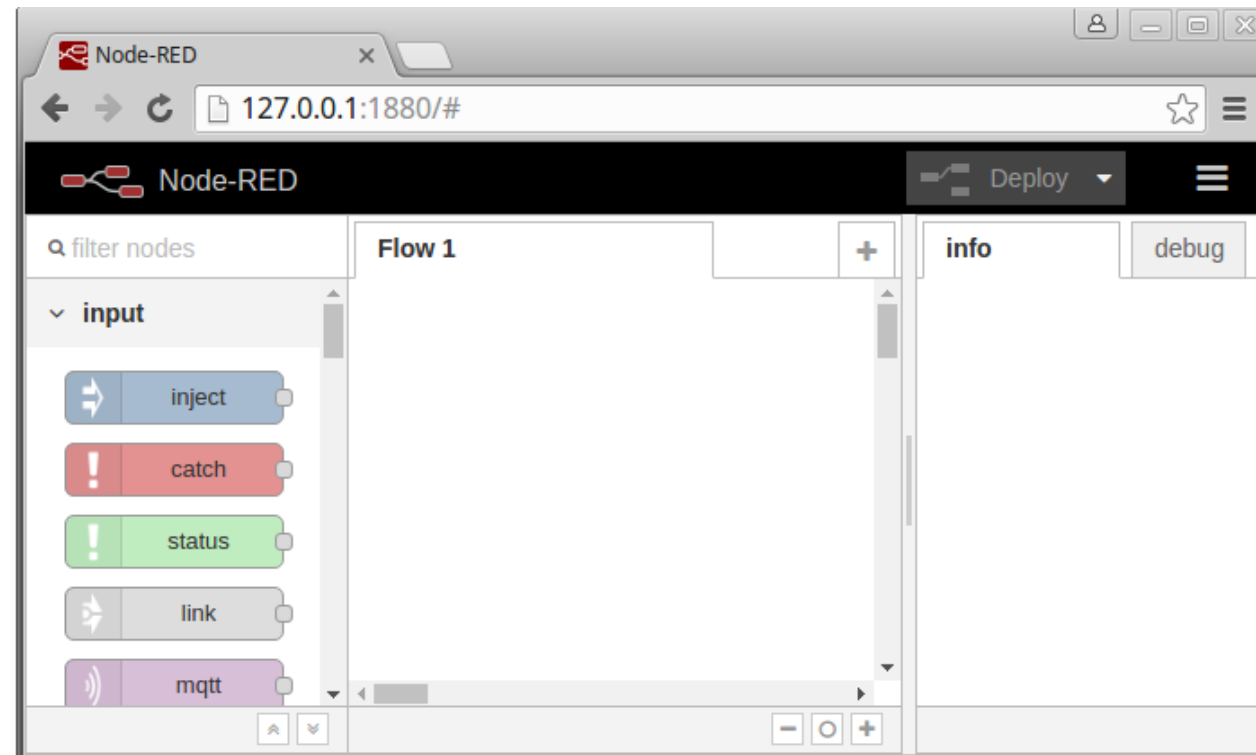
Module 4

IoT Solutions

IoT solutions through Node-red

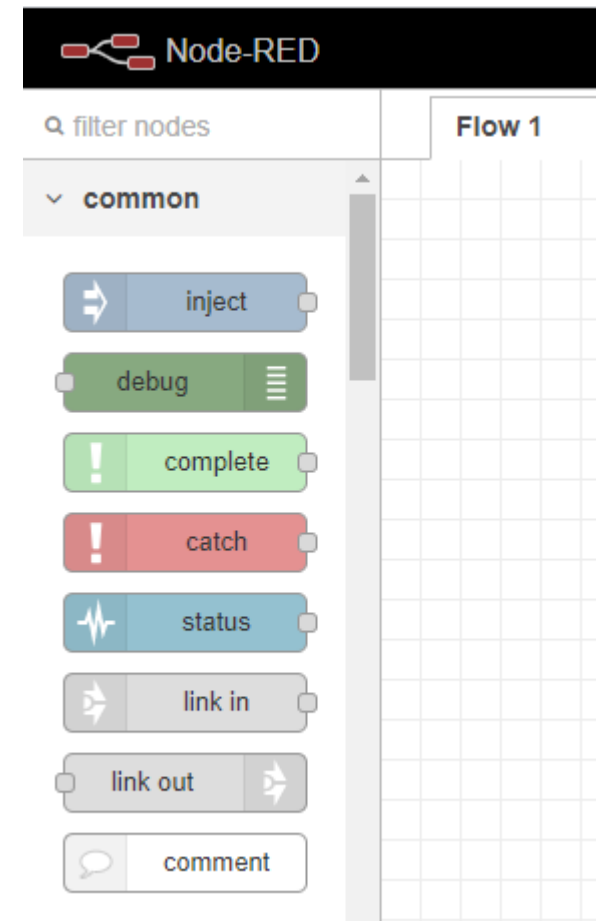
Add an Inject node

- First to start the flow:
 - **Inject Node** the main purposes of the Inject node—to inject a message into the flow.
- For first user of Node-RED there should be one empty flow named **“Flow 1”**.



Node Palette

- From the **node palette** on the left side of the **Node-RED editor**
- Select an Inject node and drag it onto the flow.



Double Click Inject Node

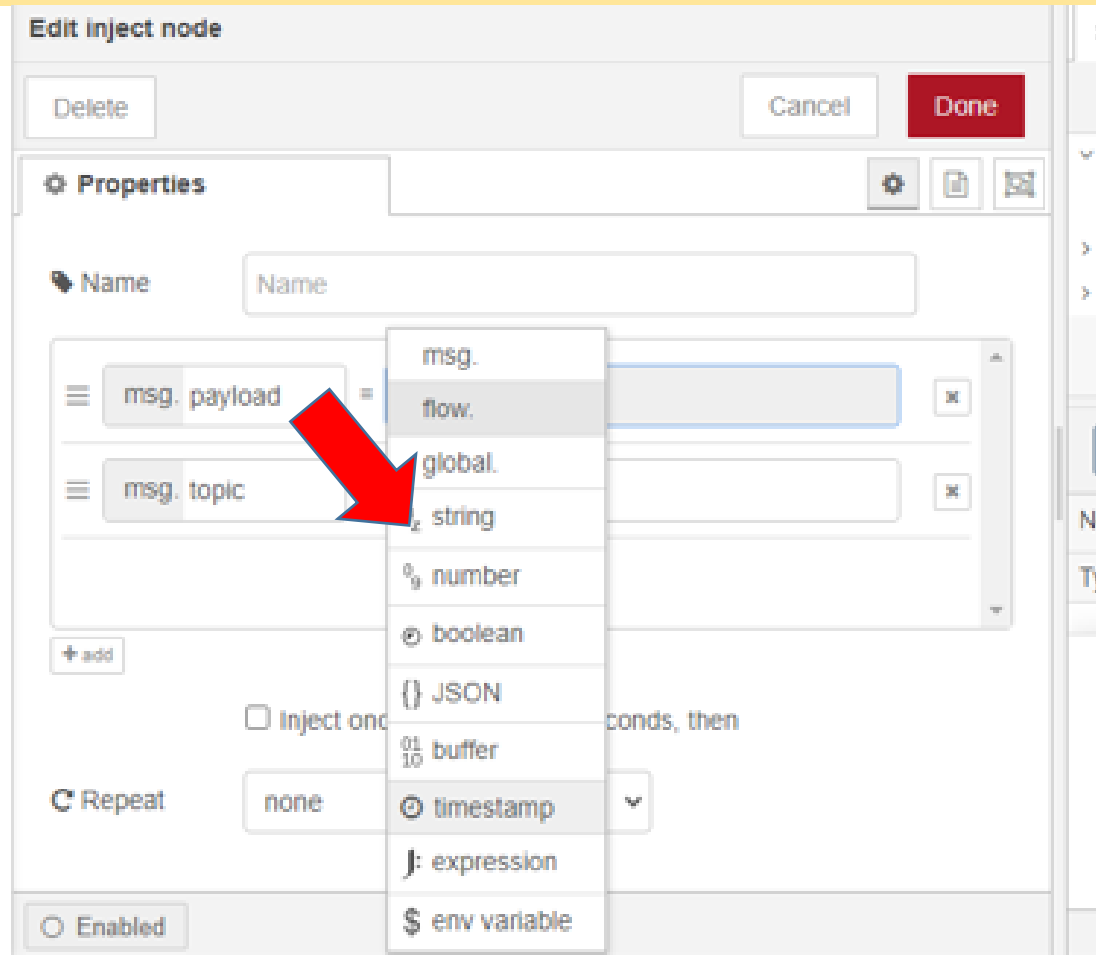
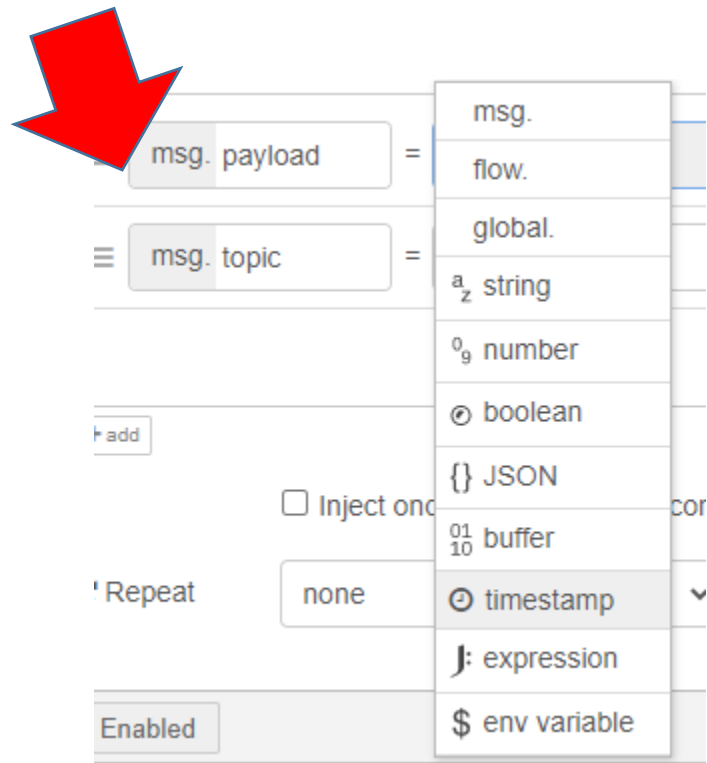
- Edit Inject Node
- Properties

The screenshot shows the 'Edit Inject node' dialog box. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below this is a 'Properties' section with a gear icon, a copy icon, and a refresh icon. The 'Name' field contains the text 'Name'. Below the name field is a list of two properties:

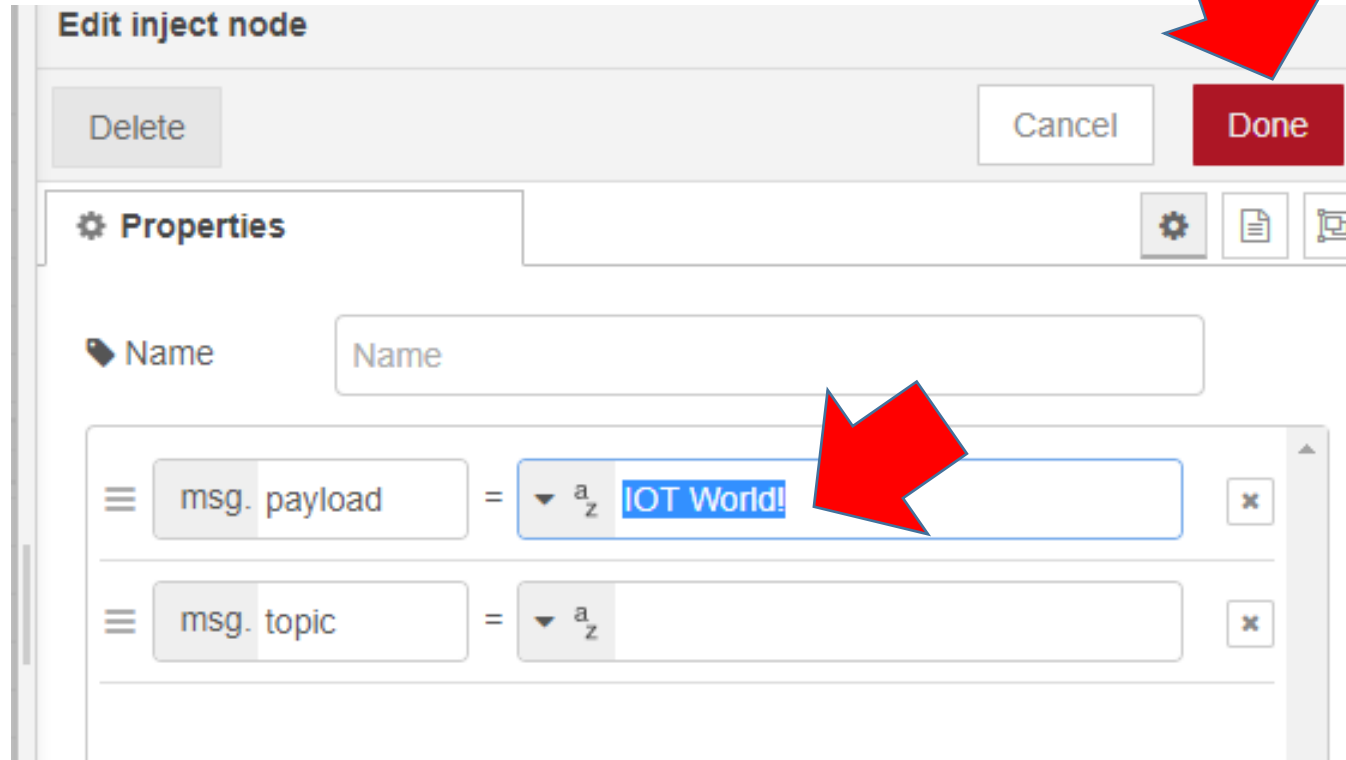
msg. payload	=	timestamp	[X]
msg. topic	=	a_z	[X]

Below the list is a '+ add' button. At the bottom, there is a checkbox labeled 'Inject once after 0.1 seconds, then' which is currently unchecked. Below that is a 'Repeat' dropdown menu set to 'none'. At the very bottom, there is an 'Enabled' checkbox which is currently unchecked.

Select msg payload -> change to string



Enter String : IOT World and click “Done”



The screenshot shows the 'Edit inject node' dialog box. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. The 'Done' button is highlighted with a red arrow. Below the buttons is a 'Properties' section with a gear icon and three sub-panels: 'Name', 'msg. payload', and 'msg. topic'. The 'Name' field is empty. The 'msg. payload' field is selected, showing a dropdown menu with 'a_z' and the text 'IOT World!' entered in the text field. A red arrow points to the text field. The 'msg. topic' field is empty.

Summary :

Double-click the node to open the “Edit inject node” view.

For the Payload field, select string and enter IOT , world! in the text field.

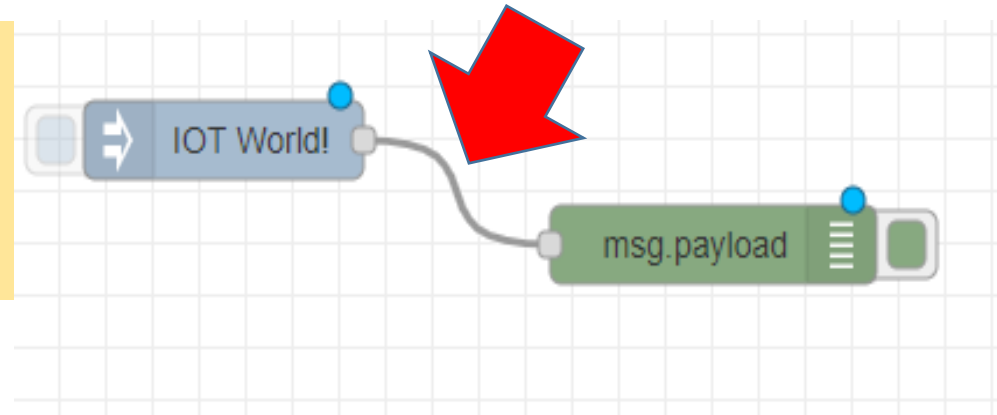
Click Done.

Add a Debug node

- Need a destination for the Inject node's message.
- Use the Debug node to **print out** message to the debug console window.
- From the node palette, select a Debug node and drag it onto the flow, and then place it to the right-hand side of the Inject node.



Wire

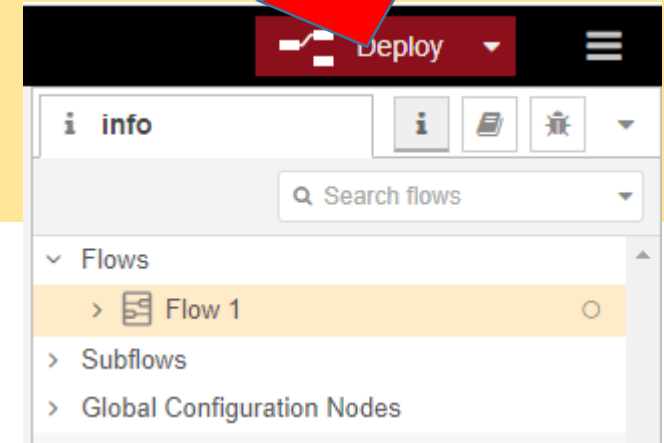


- Wire the **nodes** together.
- Place the mouse cursor over the Inject node's output port (a small gray square on the right-hand side of the node), then left-click and **drag a wire over to the input port of the Debug** node.
- A gray wire should now be connecting the output of the Inject node to the input of the Debug node.
- The Debug node will automatically **print the msg.payload** property to the console window

- Dashboard
 - Group → Tab (layout)

Deploy

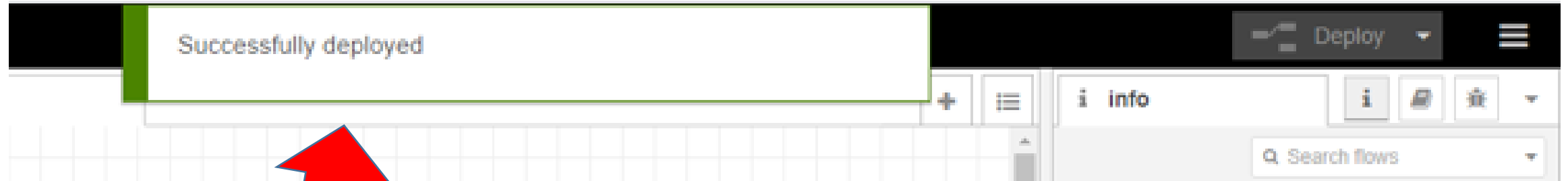
- Now that our flow is complete
- Need to deploy it to the server and run it.
- Click the Deploy button.



49.8d9fb8

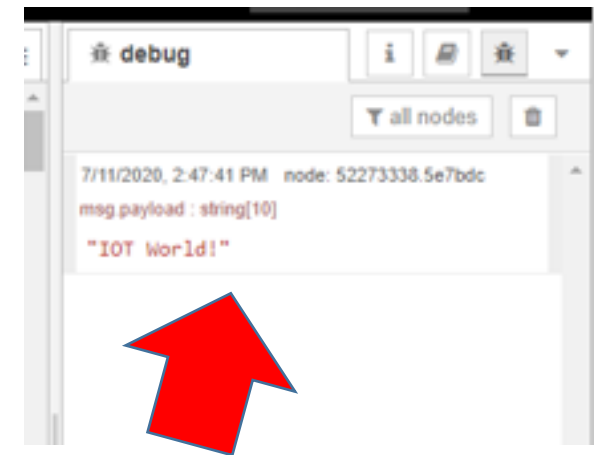
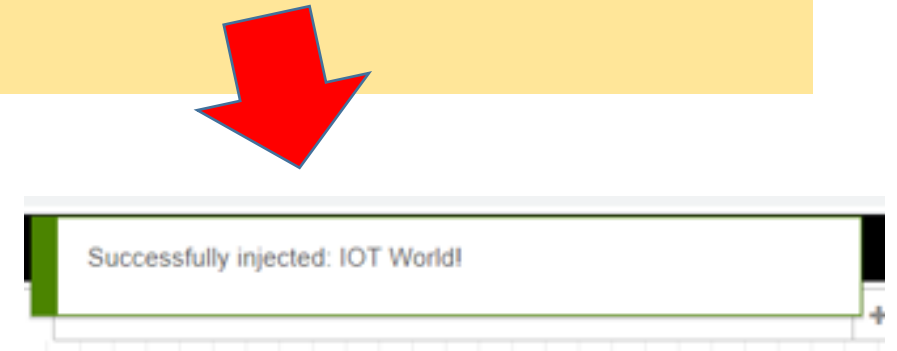


Successfully deployed



Run

- Click the Debug tab in the right-hand side of the editor window.
- Click the Inject node's button **the blue square** coming out from the left-hand side of the Inject node.
- Clicking the button will inject a message into the flow.
- A **"IOT, world!" message** should appear in the debug window.
- Click the Inject node again to send another message.



Successfully injected: IOT World!

filter nodes

Flow 1

common

- inject
- debug
- complete
- catch
- status
- link in
- link out
- comment

function

function



debug

all nodes

7/11/2020, 2:47:41 PM node: 52273338.5e7bdc
msg.payload : string[10]
"IOT World!"

Successfully injected: IOT World!

filter nodes

Flow 1

common

- inject
- debug
- complete
- catch
- status
- link in
- link out
- comment

function

function



debug

all nodes

```
7/11/2020, 2:47:41 PM node: 52273338.5e7bdc  
msg.payload : string[10]  
"IOT World!"  
  
7/11/2020, 2:48:10 PM node: 52273338.5e7bdc  
msg.payload : string[10]  
"IOT World!"
```

Function Node

Add the current time to the message, and execute it every two seconds.

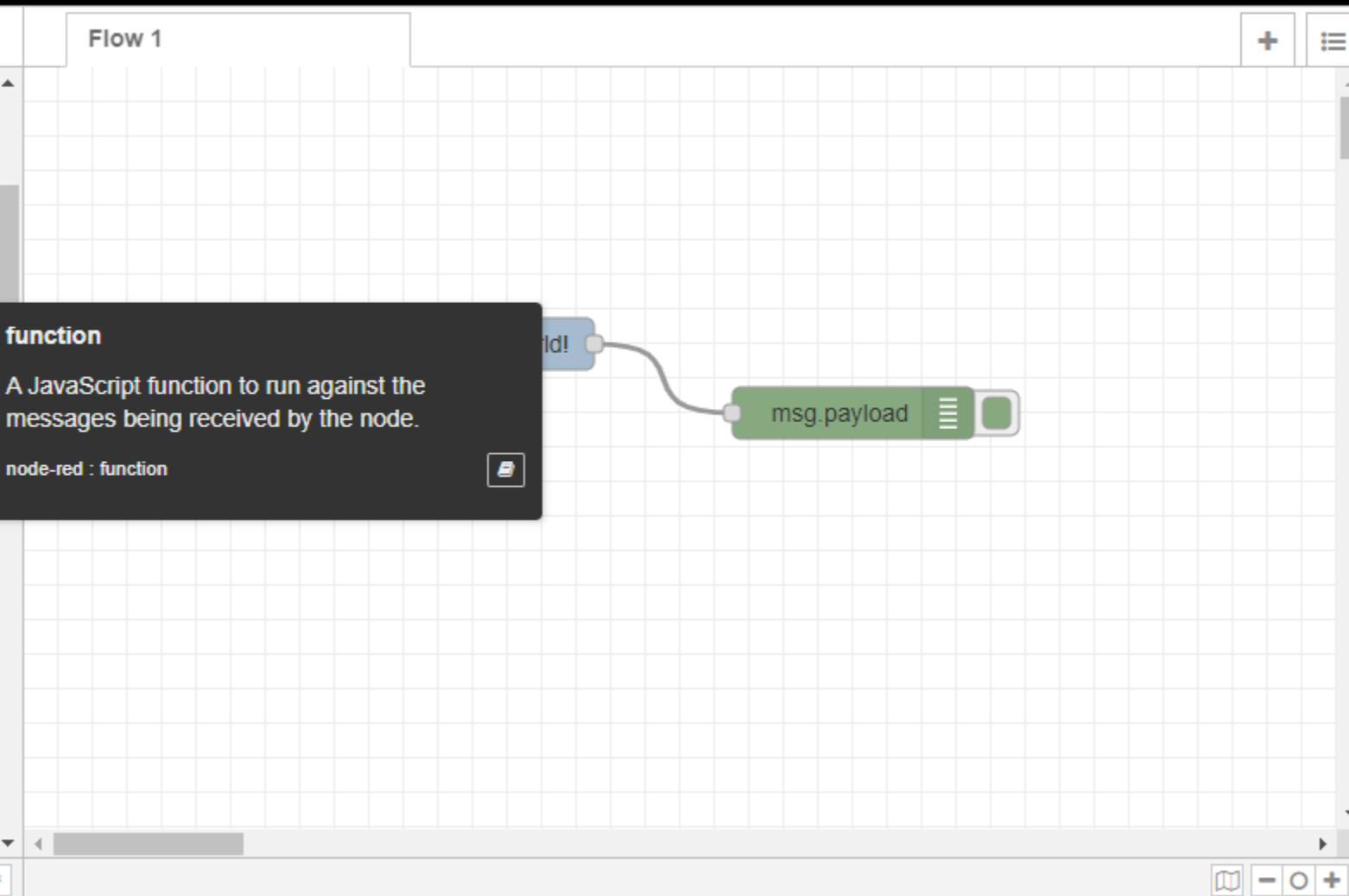
Add a Function node

- Add the current time to the message.
- Use the Function block - allows to enter JavaScript code to manipulate the msg object.
- Select a Function node and drag it out onto the flow.



filter nodes

- link in
- link out
- comment
- function
- switch
- change
- range
- template
- delay
- trigger



function
A JavaScript function to run against the messages being received by the node.
node-red : function

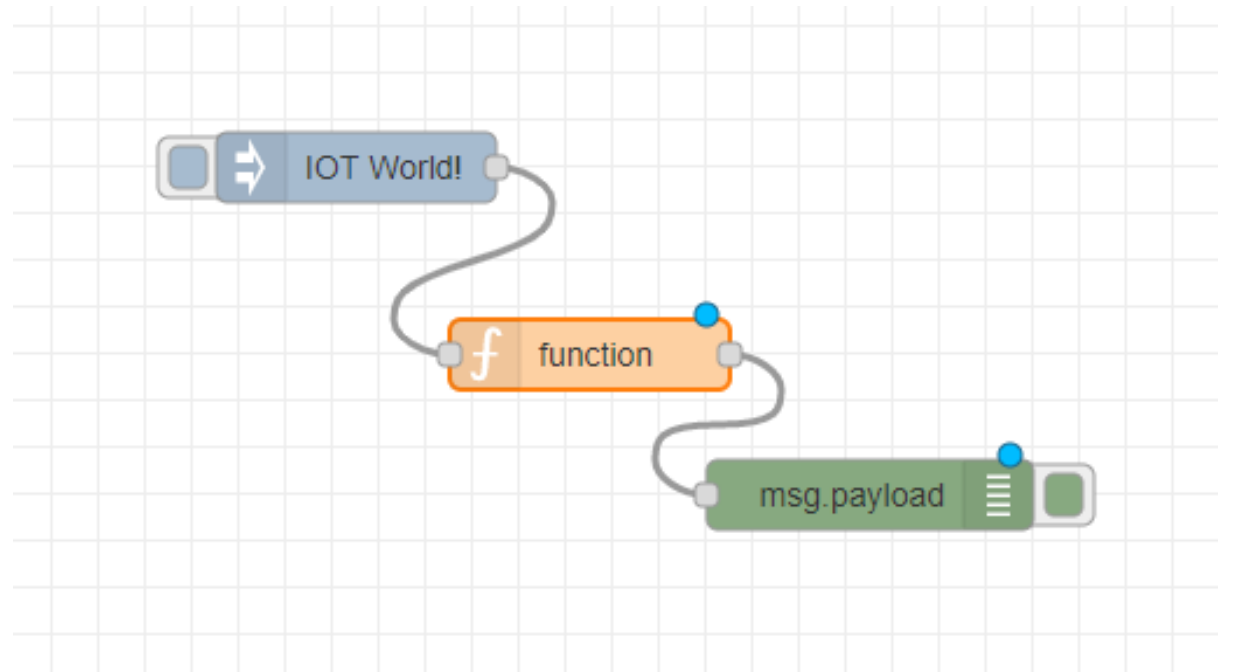
debug

all nodes

```
7/11/2020, 2:47:41 PM node: 52273338.5e7bdc  
msg.payload : string[10]  
"IOT World!"  
7/11/2020, 2:48:10 PM node: 52273338.5e7bdc  
msg.payload : string[10]  
"IOT World!"
```

Placing

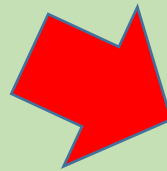
- Carefully place it over the existing wire between the existing Inject and Debug nodes.
- Node-RED will insert the new node between the two existing nodes, and **rewire** the nodes together.



Add Code and Run

- **Double-click** on the Function node to open the “**Edit function node**” view.
- Copy and paste the JavaScript code into the Function field:

```
var dateNow = new Date();  
var timeAsString = dateNow.toLocaleTimeString();  
  
msg.payload = msg.payload +  
    ' The Current time is ' +  
    timeAsString + '.';  
  
return msg;
```

The screenshot shows the 'Edit function node' dialog box. At the top, there are 'Delete', 'Cancel', and 'Done' buttons. Below is a 'Properties' section with a 'Name' field. There are three tabs: 'Setup', 'Function', and 'Close', with 'Function' selected. The 'Function' tab contains a code editor with two lines of JavaScript code: '1' and '2 return msg;'. At the bottom, there is an 'Outputs' section with a dropdown menu set to '1'.

Click Done

Edit function node

Delete Cancel Done

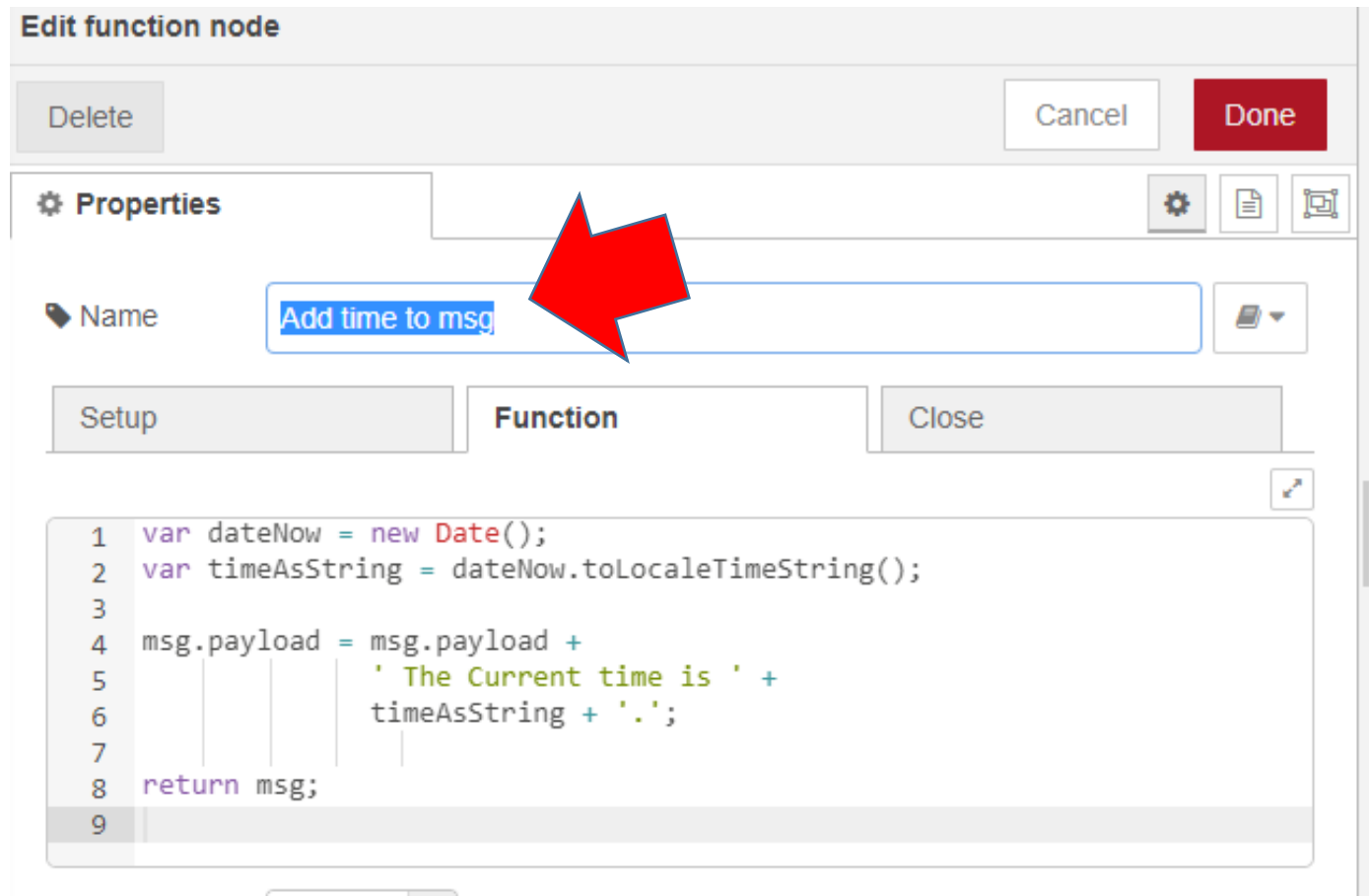
⚙ Properties

Name Name

Setup Function Close

```
1 var dateNow = new Date();
2 var timeAsString = dateNow.toLocaleTimeString();
3
4 msg.payload = msg.payload +
5   ' The Current time is ' +
6   timeAsString + '.';
7
8 return msg;
9
```

Add name : “Add time to msg”



Edit function node

Delete Cancel Done

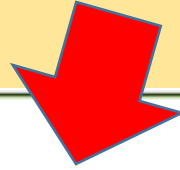
Properties

Name Add time to msg

Setup **Function** Close

```
1 var dateNow = new Date();
2 var timeAsString = dateNow.toLocaleTimeString();
3
4 msg.payload = msg.payload +
5   ' The Current time is ' +
6   timeAsString + '.';
7
8 return msg;
9
```

Click Deploy



The screenshot displays the Node-RED interface with a flow named 'Flow 1'. The flow consists of three nodes connected in sequence: a blue 'IOT World!' node, an orange 'Add time to msg' function node, and a green 'msg payload' node. A notification bar at the top center indicates 'Successfully deployed'. The right sidebar shows a debug console with two log entries for 'IOT World!'.

```
graph LR; IOTWorld[IOT World!] --> AddTime[Add time to msg]; AddTime --> MsgPayload[msg payload];
```

debug console logs:

```
7/11/2020, 2:47:41 PM node: 52273338.5e7bdc  
msg payload : string[10]  
"IOT World!"  
7/11/2020, 2:48:10 PM node: 52273338.5e7bdc  
msg payload : string[10]  
"IOT World!"
```

Click inject node (Blue Button)

The screenshot displays the Node-RED web interface. At the top, a notification box with a green border contains the text "Successfully injected: IOT World!". A large red arrow points from the top of the image to this notification box. Below the notification, the main workspace shows a flow named "Flow 1" on a grid. The flow consists of three nodes connected in sequence: a blue "inject" node with the text "IOT World!", an orange "function" node with the text "Add time to msg", and a green "msg.payload" node. The left sidebar shows a "function" category with various nodes like "function", "switch", "change", "range", "template", "delay", and "trigger". The right sidebar shows a "debug" console with three log entries:

```
7/11/2020, 2:47:41 PM node: 52273338.5e7bdc  
msg.payload : string[10]  
"IOT World!"  
7/11/2020, 2:48:10 PM node: 52273338.5e7bdc  
msg.payload : string[10]  
"IOT World!"  
7/11/2020, 2:56:51 PM node: 52273338.5e7bdc  
msg.payload : string[42]  
"IOT World! The Current time is  
2:56:51 PM."
```

Debug Window – Output message

```
7/11/2020, 2:56:51 PM node: 52273338.5e7bdc
```

```
msg.payload : string[42]
```

```
"IOT World! The Current time is  
2:56:51 PM."
```

```
7/11/2020, 2:57:01 PM node: 52273338.5e7bdc
```

```
msg.payload : string[42]
```

```
"IOT World! The Current time is  
2:57:01 PM."
```

```
7/11/2020, 2:57:03 PM node: 52273338.5e7bdc
```

```
msg.payload : string[42]
```

```
"IOT World! The Current time is  
2:57:03 PM."
```


Add Two-Second Interval

- Adjust the flow to automatically inject a new message every two seconds.
- The existing Inject node need to adjust its settings.
- Double-click on the existing Inject node to open the “**Edit inject node**” view.

Edit inject node

Delete Cancel Done

⚙️ Properties 📄 🖨️

📌 Name Name

☰ msg.payload = a_z IOT World! ✕

☰ msg.topic = a_z ✕

+ add

Inject once after 0.1 seconds, then

🔄 Repeat interval ▾

every 2 | seconds ▾

Repeat

- For the Repeat field, select **interval** and enter **2 seconds** for the period.
- Click Done.
- Click Deploy.

Edit inject node

Delete Cancel Done

⚙️ Properties

📁 Name Name

msg. payload = a_z IOT World!

msg. topic = a_z

Repeat

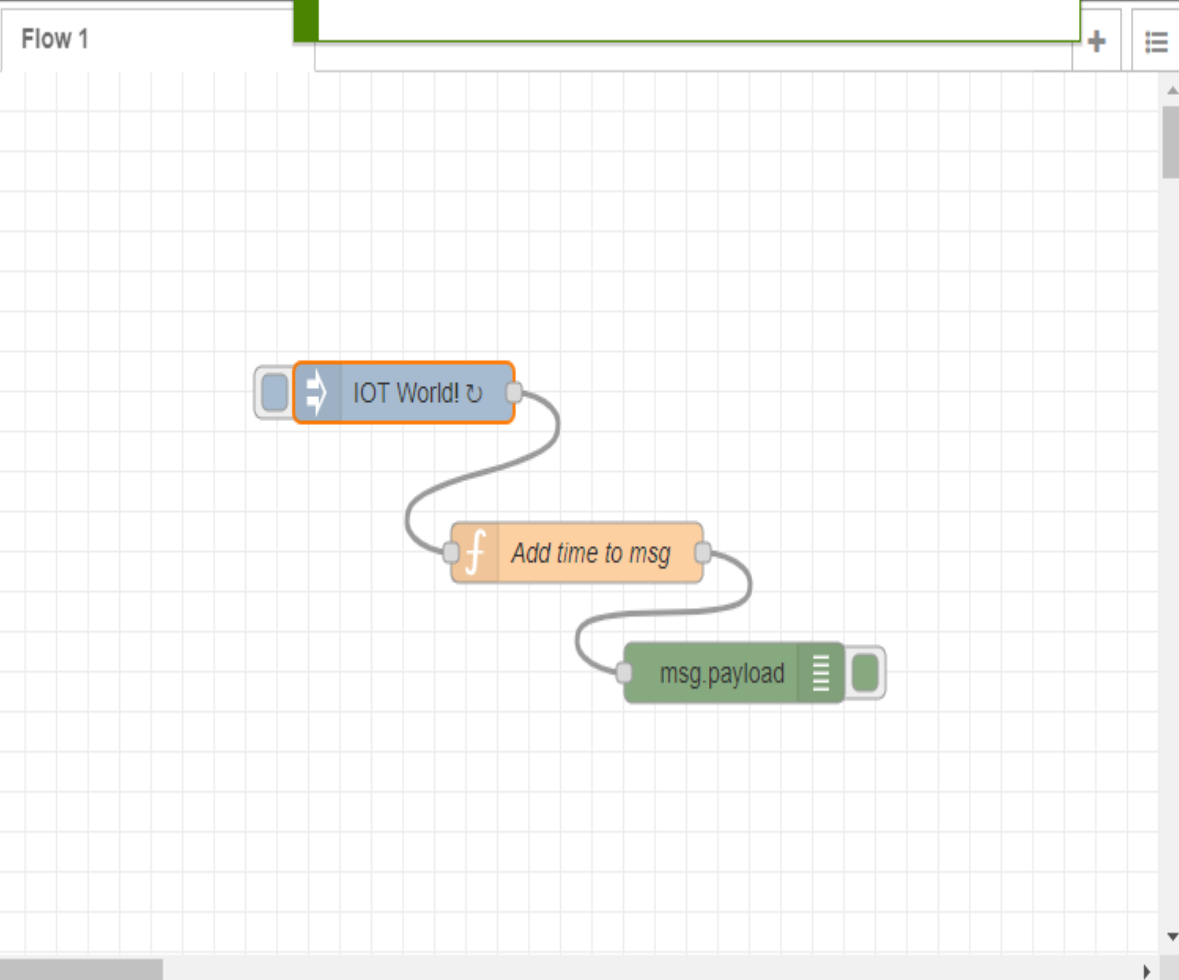
Inject once after 0.1 seconds, then

interval

every 2 seconds

Node-RED Successfully deployed Deploy

- filter nodes
- link in
 - link out
 - comment
 - function
 - function
 - switch
 - change
 - range
 - template
 - delay
 - trigger



debug

all nodes

```
7/11/2020, 2:48:10 PM node: 52273338.5e7bdc
msg.payload : string[10]
"IOT World!"

7/11/2020, 2:56:51 PM node: 52273338.5e7bdc
msg.payload : string[42]
"IOT World! The Current time is 2:56:51 PM."

7/11/2020, 2:57:01 PM node: 52273338.5e7bdc
msg.payload : string[42]
"IOT World! The Current time is 2:57:01 PM."

7/11/2020, 2:57:03 PM node: 52273338.5e7bdc
msg.payload : string[42]
"IOT World! The Current time is 2:57:03 PM."

7/11/2020, 2:59:48 PM node: 52273338.5e7bdc
msg.payload : string[42]
"IOT World! The Current time is 2:59:48 PM."
```

In the Debug tab, should now see your output every two seconds.

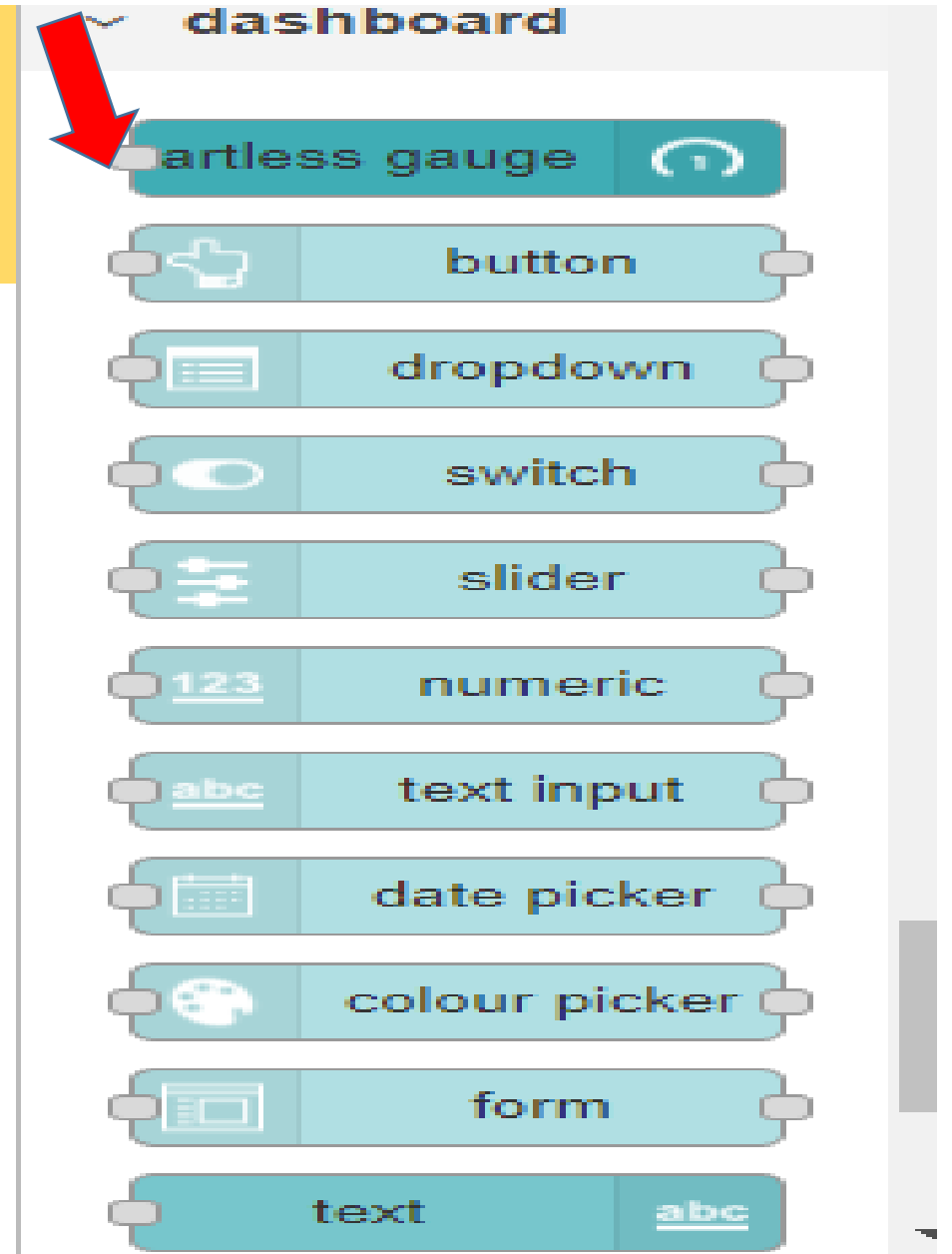
The screenshot displays a Node-RED workflow on a grid background. The flow consists of three nodes connected in sequence:

- Inject Node:** A blue node labeled "IOT World! ☺" with a right-pointing arrow icon.
- Function Node:** An orange node labeled "Add time to msg" with a function symbol 'f'.
- Debug Node:** A green node labeled "msg.payload" with a list icon.

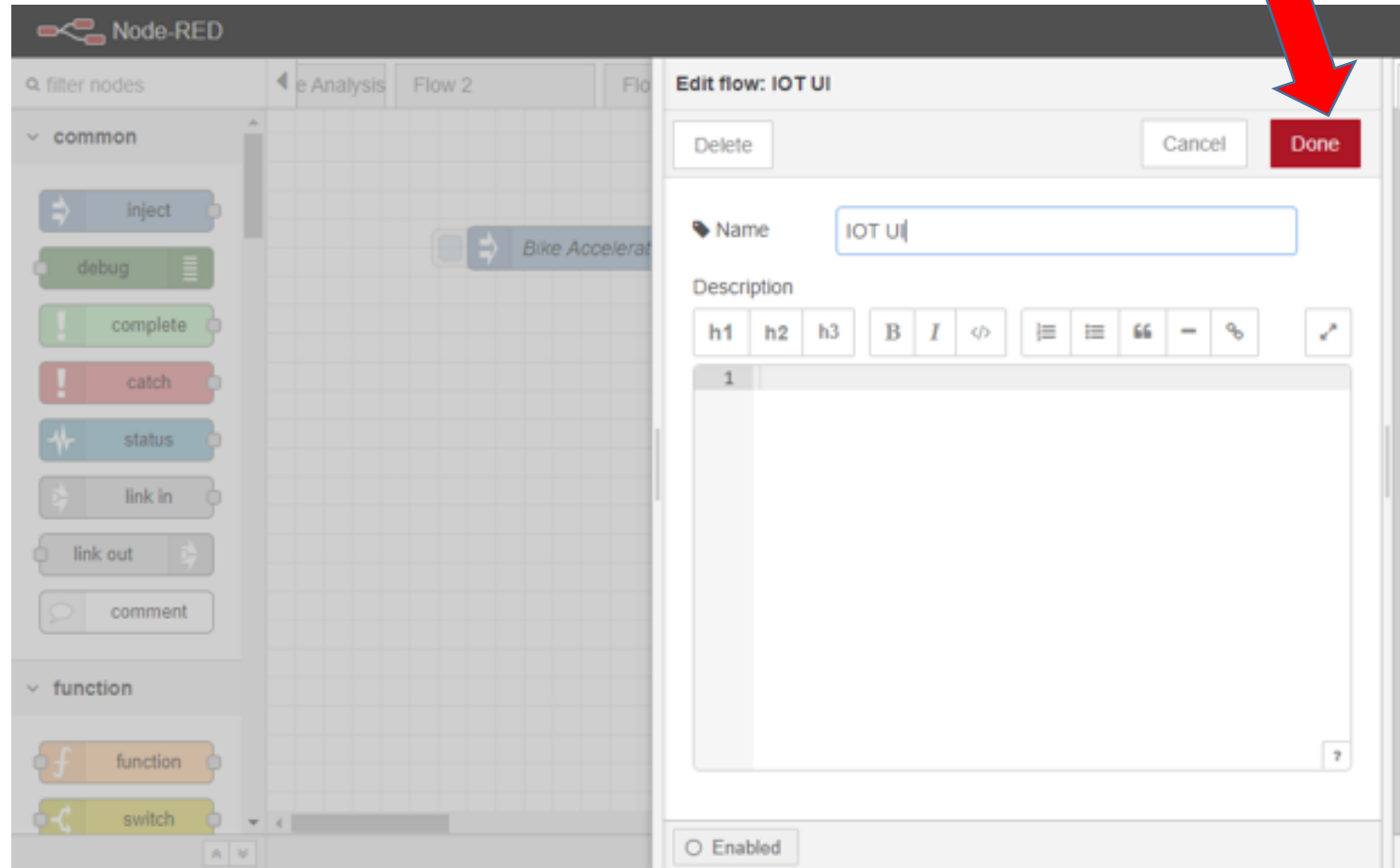
The debug console on the right side of the interface shows the output of the flow. Each entry includes a timestamp and the node ID, followed by the message payload:

```
msg.payload : string[42]  
"IOT World! The Current time is  
3:00:34 PM."  
7/11/2020, 3:00:36 PM node: 52273338.5e7bdc  
msg.payload : string[42]  
"IOT World! The Current time is  
3:00:36 PM."  
7/11/2020, 3:00:38 PM node: 52273338.5e7bdc  
msg.payload : string[42]  
"IOT World! The Current time is  
3:00:38 PM."  
7/11/2020, 3:00:40 PM node: 52273338.5e7bdc  
msg.payload : string[42]  
"IOT World! The Current time is  
3:00:40 PM."  
7/11/2020, 3:00:42 PM node: 52273338.5e7bdc  
msg.payload : string[42]  
"IOT World! The Current time is  
3:00:42 PM."
```

Widgets

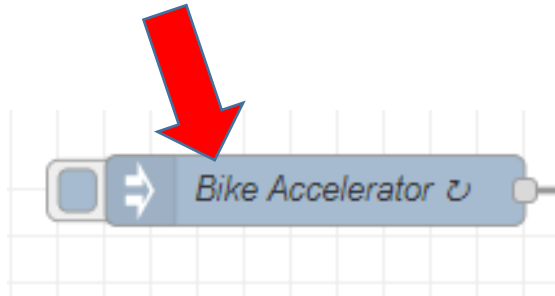


Create a Flow and edit the flow name



The screenshot displays the Node-RED interface. On the left, a sidebar shows various nodes categorized under 'common' (inject, debug, complete, catch, status, link in, link out, comment) and 'function' (function, switch). The main workspace contains a single node labeled 'Bike Accelerat'. On the right, the 'Edit flow: IOT UI' dialog box is open. It features a 'Delete' button, a 'Cancel' button, and a prominent red 'Done' button. A red arrow points to the 'Done' button. The dialog also includes a 'Name' field containing 'IOT UI', a 'Description' field with a rich text editor toolbar (h1, h2, h3, B, I, code, list, quote, link, unlink, help), and an 'Enabled' checkbox at the bottom.

Inject Node



Edit inject node

Delete Cancel Done

Properties

Name: Bike Accelerator

msg. payload = 0₉ 80

msg. topic = a_z

+ add

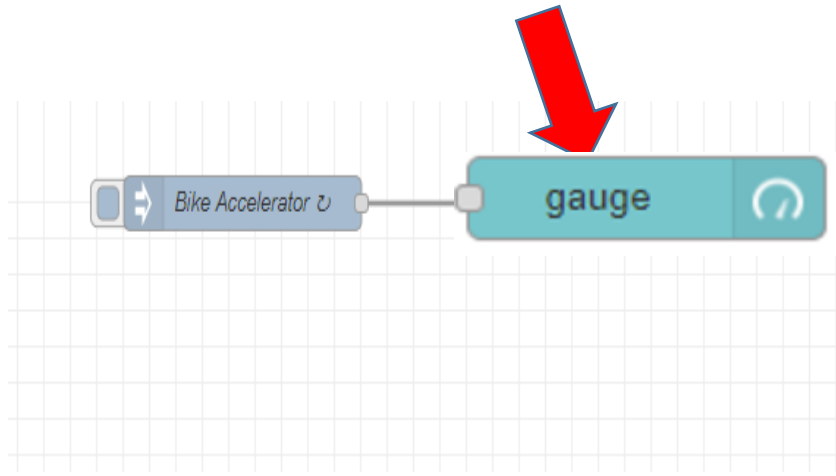
Inject once after 0.1 seconds, then

Repeat: interval

every 2 seconds

Enabled

Insert and Edit Gauge Node



Edit gauge node

Delete Cancel Done

Properties

SIZE auto

Type Gauge

Label gauge

Value format {{value}}

Units units

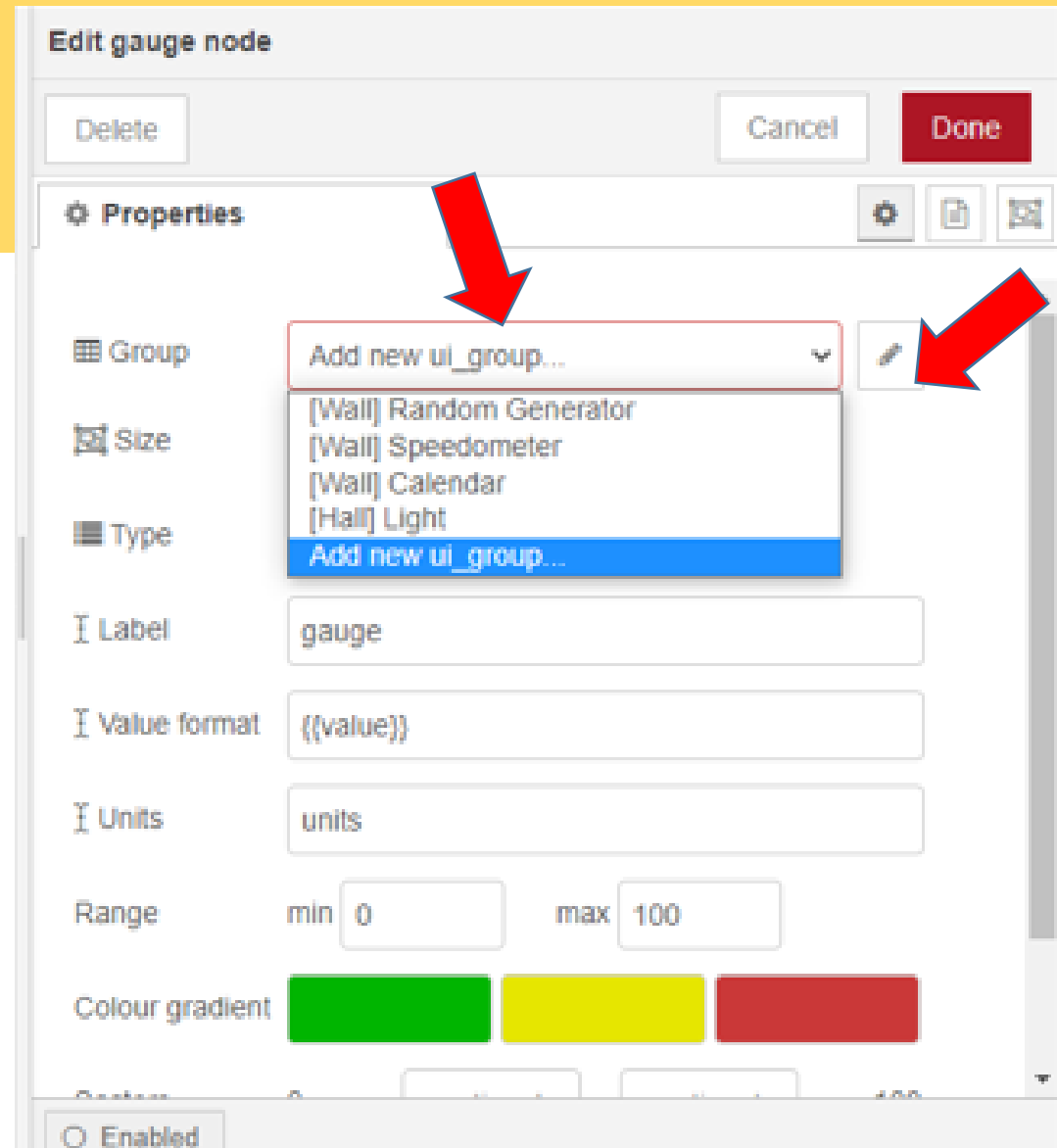
Range min 0 max 180

Colour gradient

Sectors 0 ... optional ... optional ... 180

Name Speedometer

Layout :Group 1




Tab Name

Edit gauge node > Edit dashboard group node

Delete Cancel Update

Properties

Name IOT Panel 1

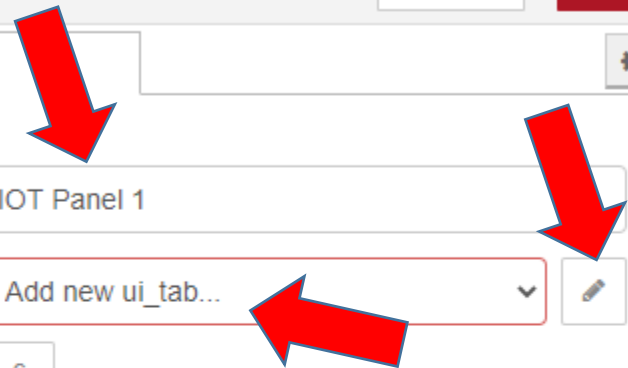
Tab Add new ui_tab... 

Width 6

Display group name

Allow group to be collapsed

Enabled **1 node uses this config** On all flows



Edit gauge node > Edit dashboard group node > Edit dashboard tab node

Delete Cancel Update

Properties

Name IOT Devices

Icon dashboard

State Enabled

Nav. Menu Visible

The **Icon** field can be either a [Material Design icon](#) (e.g. 'check', 'close') or a [Font Awesome icon](#) (e.g. 'fa-fire'), or a [Weather icon](#) (e.g. 'wi-wu-sunny').

You can use the full set of google material icons if you add 'mi-' to the icon name. e.g. 'mi-videogame_asset'.

Enabled **2 nodes use this config** On all flows

Edit gauge node > **Edit dashboard group node**

Delete Cancel **Update**

Properties

Name IOT Panel 1

Tab IOT Devices

Width 6

Display group name

Allow group to be collapsed

Edit gauge node

Delete Cancel **Done**

Properties

Group [IOT Devices] IOT Panel 1

Size auto


Type Gauge

Label Speedometer

Value format {{value}}

Units units

Range min 0 max 200

Colour gradient 

Sectors 0 ... optional ... optional ... 200

Flot Edit gauge node

Delete Cancel Done

⚙ Properties

📁 Group [IOT Devices] IOT Panel 1

📏 Size auto

📄 Type Level


🏷 Label

📏 Units

Range min 0 max 200

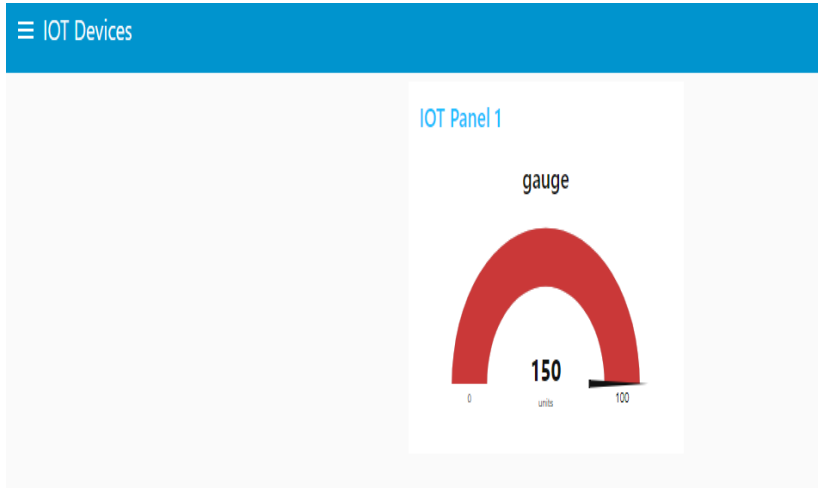
👉 Name IOT UI

Enabled

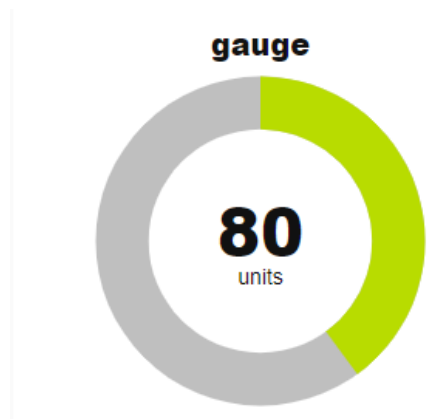


The image shows a software interface for editing a gauge node. The 'Type' dropdown menu is open, showing options: Level, Gauge, Donut, Compass, and Level. A red arrow points to the 'Gauge' option, which is highlighted in blue. The interface includes buttons for 'Delete', 'Cancel', and 'Done', and a 'Properties' section with various configuration fields like Group, Size, Type, Label, Units, Range, and Name.

Gauge

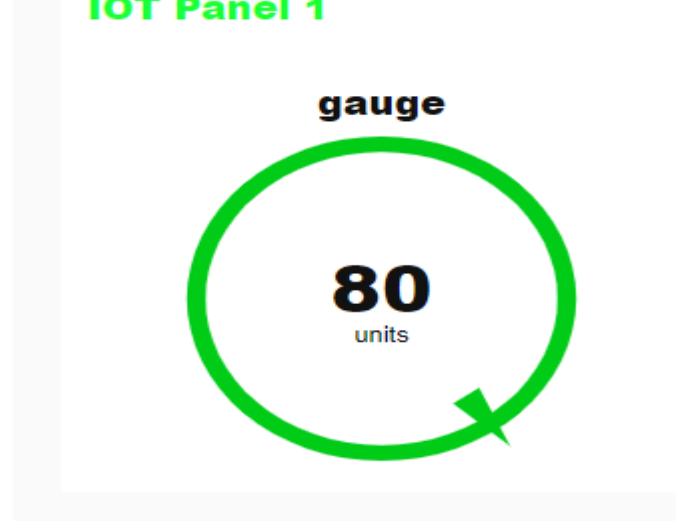


Donut



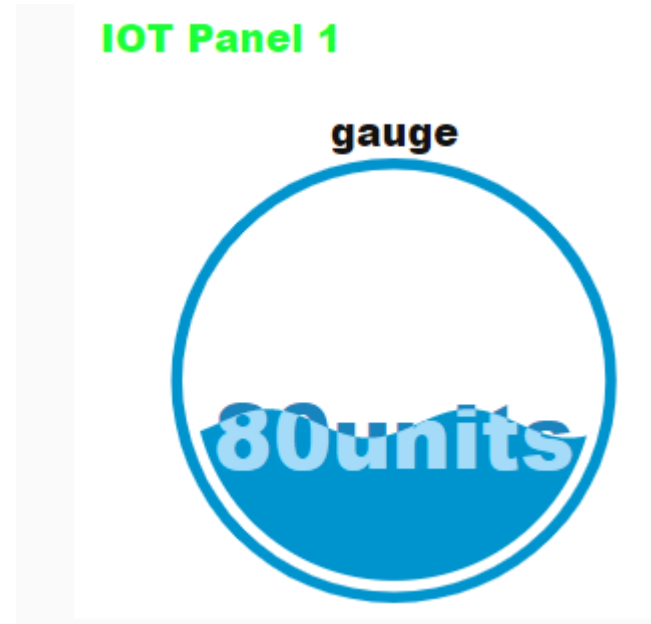
Compass

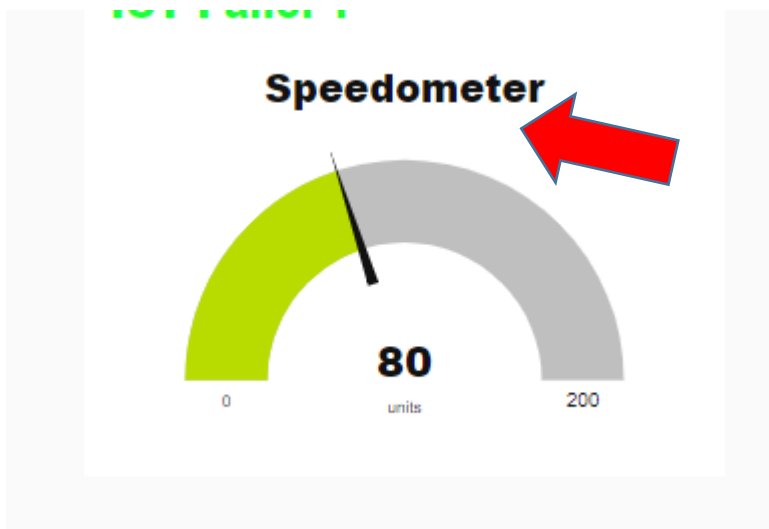
IOT Panel 1



Level

IOT Panel 1





Edit gauge node

Delete Cancel Done

Properties

Group [IOT Devices] IOT Panel 1

Size auto

Type Level

Label Speedometer

Units units

Range min 0 max 200




Name Speedometer


Enabled

The "Edit gauge node" dialog box contains several fields for configuring the gauge. The "Label" field is set to "Speedometer", the "Units" field is set to "units", and the "Name" field is set to "Speedometer". The "Range" field shows a minimum of 0 and a maximum of 200. The "Done" button is highlighted in red, and a red arrow points to it. Another red arrow points to the "Label" field, and a third red arrow points to the "Name" field.

Edit gauge node

Delete Cancel Done

Properties   


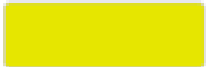

 **Type** Gauge

Label gauge

Value format {{value}}

Units units

Range min 0 max 100

Colour gradient   

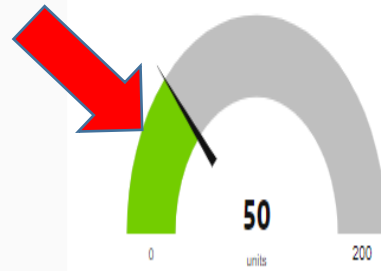
Sectors 0 ... optional ... optional ... 100

Name IOT U|

Enabled

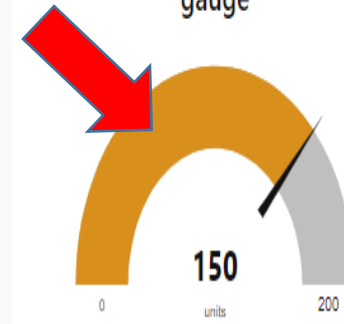
IOT Panel 1

gauge



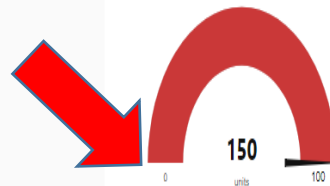
IOT Panel 1

gauge



IOT Panel 1

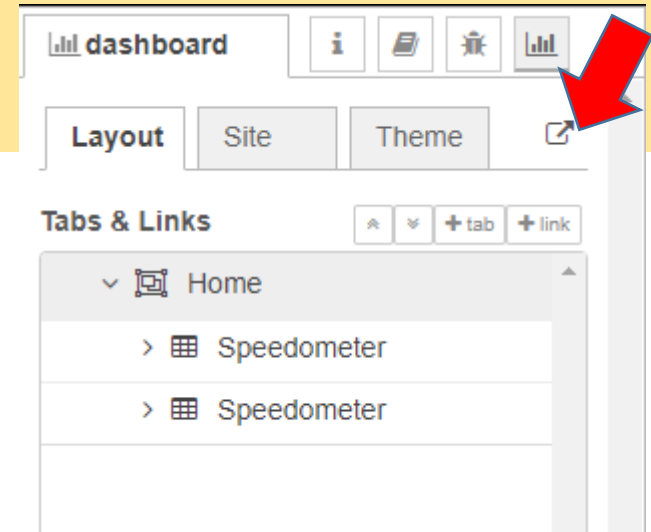
gauge



Wire, Deploy and Execute



<http://127.0.0.1:1880/ui>



IOT World

Wall

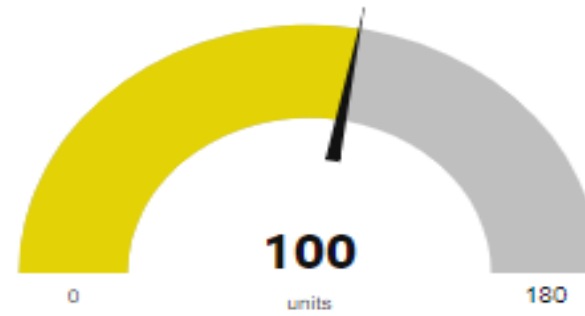
IOT Banking

IOT Devices



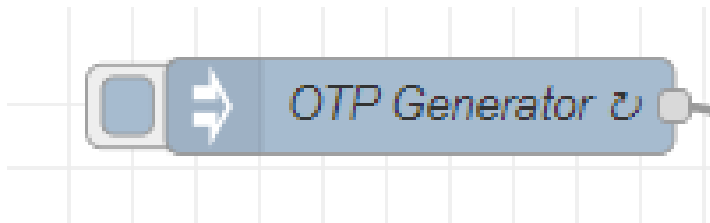
Speedometer

gauge



Layout :Group 2

- Insert Inject Node and Edit it



Edit inject node

Delete Cancel Done

⚙ Properties

Name OTP Generator

msg. payload = a_z OTP Generator

msg. topic = timestamp

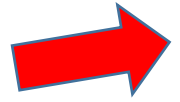

+ add

Inject once after 0.1 seconds, then

Repeat interval

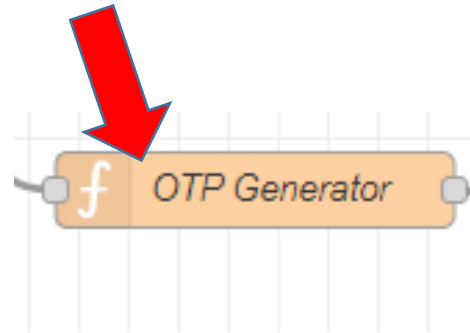
every 3 seconds

Enabled

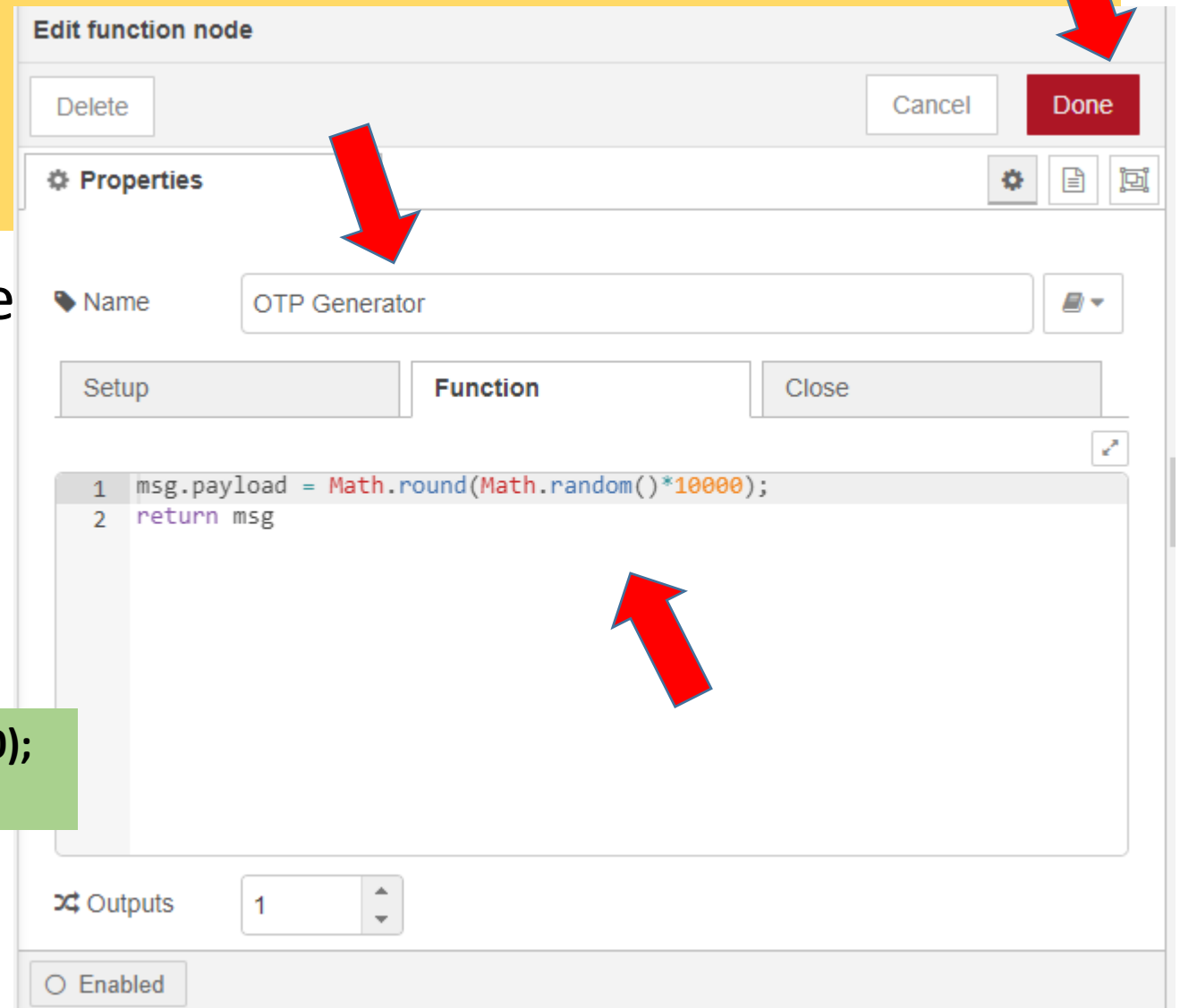


Function Node

- Insert and edit Function Node



```
msg.payload = Math.round(Math.random()*10000);  
return msg
```



Delete Cancel Done

⚙ Properties

Name OTP Generator

Setup Function Close

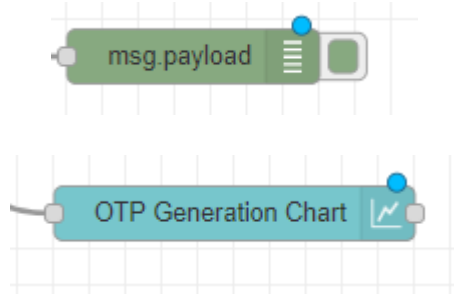
```
1 msg.payload = Math.round(Math.random()*10000);  
2 return msg
```

Outputs 1

Enabled

Debug and Chart node

- Insert Debug node
- Insert Chart Node and Edit it





The screenshot shows the 'Edit chart node' dialog box. At the top, there are buttons for 'Delete', 'Cancel', and 'Done'. Below is a 'Properties' section with various settings. Two red arrows point to the 'Group' dropdown menu and the 'Done' button. The 'Group' dropdown is currently set to 'Add new ui_group...'. Other settings include 'Size' (auto), 'Label' (OTP Generation Chart), 'Type' (Line chart), 'X-axis' (last 5 minute: OR 1000 points), 'X-axis Label' (HH:mm:ss), 'Y-axis' (min and max fields), and 'Legend' (Show, Interpolate linear).



Group and Tab Name

Edit chart node > Edit dashboard group node

Delete Cancel Update

⚙️ Properties  

📌 Name Bank

📅 Tab IOT Banking  

↔️ Width 6

Display group name

Allow group to be collapsed



Enabled  4 nodes use this config On all flows 

Chart Label

Edit chart node

Delete Cancel Done

Properties

Group [IOT Banking] Bank

Size auto

Label OTP Generation Chart

Type Line chart enlarge points

X-axis last 5 minute: OR 1000 points

X-axis Label HH:mm:ss as UTC

Y-axis min max

Legend Show Interpolate linear

Series Colours

Chart Type

Edit chart node

Delete Cancel Done

Properties

Group [IOT Devices] IOT Panel 2

Size auto

Label OTP Generation Chart

Type

- Pie chart
- Line chart
- Bar chart
- Bar chart (H)
- Pie chart**
- Polar area chart
- Radar chart

Legend

Series Colours

Blank label display this text before valid data arrives

Enabled

Chart Name

Edit chart node

Delete Cancel Done

Properties

Size auto

Label OTP Generation Chart

Type Pie chart

Legend Show Cutout 0 %

Series Colours

Blank label display this text before valid data arrives

Name

Enabled




Chart Type

≡ IOT Banking

Bank

Gauge **8421**

slider 

OTP Generation Chart



OTP VALUE
8421



Bank

Gauge **11**

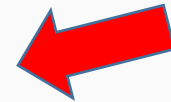
slider 

OTP Generation Chart

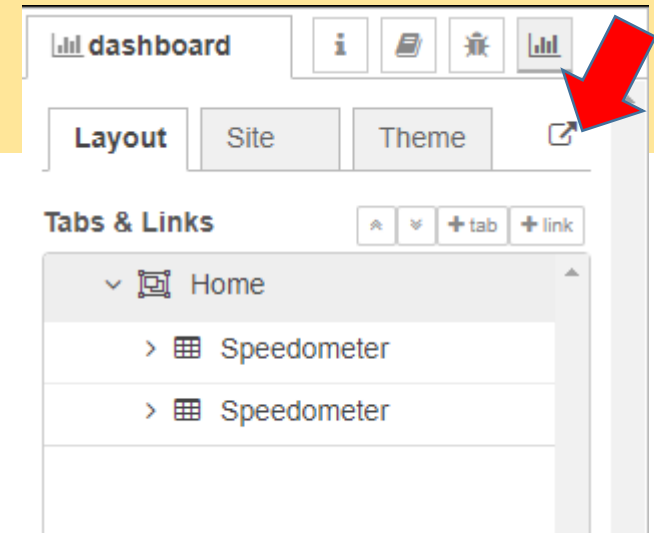
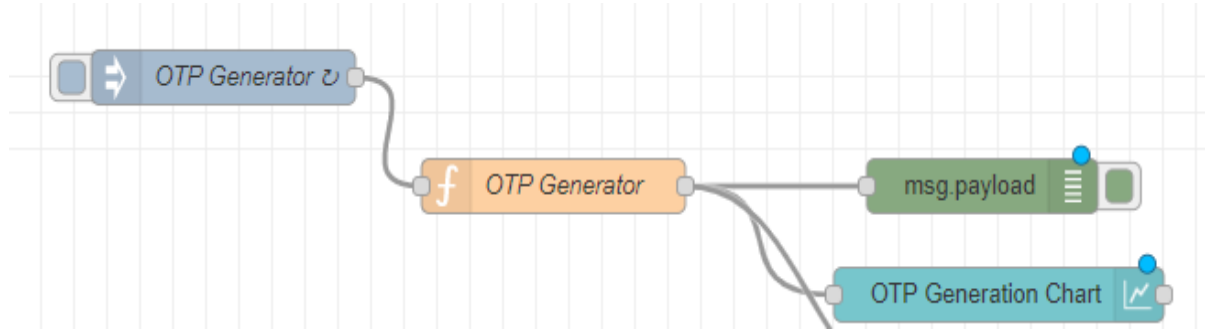
Line1
Line2



OTP VALUE
11



Wire, Deploy and Execute



<http://127.0.0.1:1880/ui>



Layout Group

IOT World

Wall

IOT Banking

IOT Devices



IOT Banking

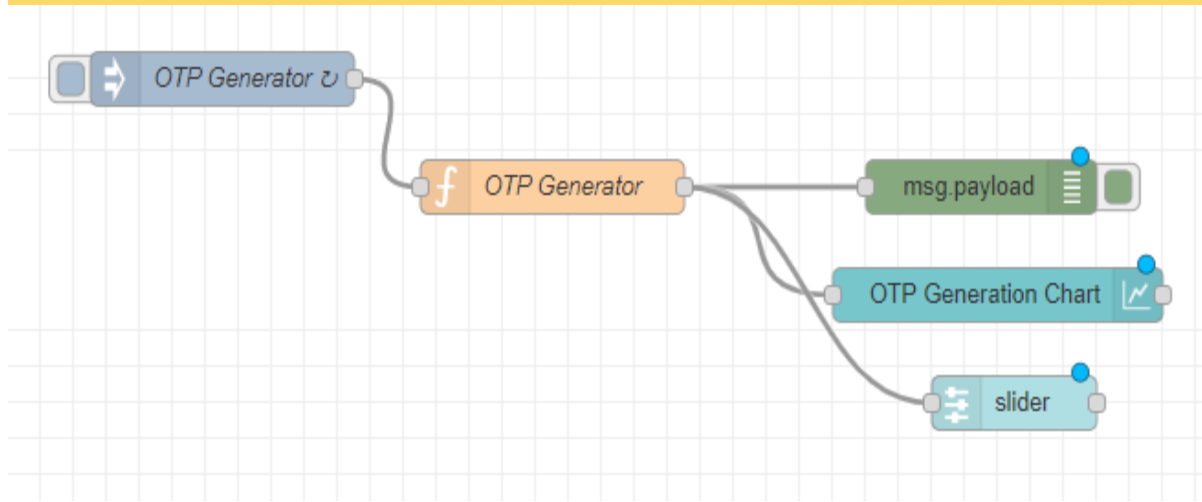
Bank

OTP Generation Chart

Line1
Line2



Slider Node



Edit slider node

Delete

Cancel

Done

Properties

Group

[IOT Devices] IOT Panel 2

Size

auto

Label

slider

Tooltip

optional tooltip

Range

min 0

max 10

step 1

Output

continuously while sliding

→ If `msg` arrives on input, set slider to new payload value:

When changed, send:

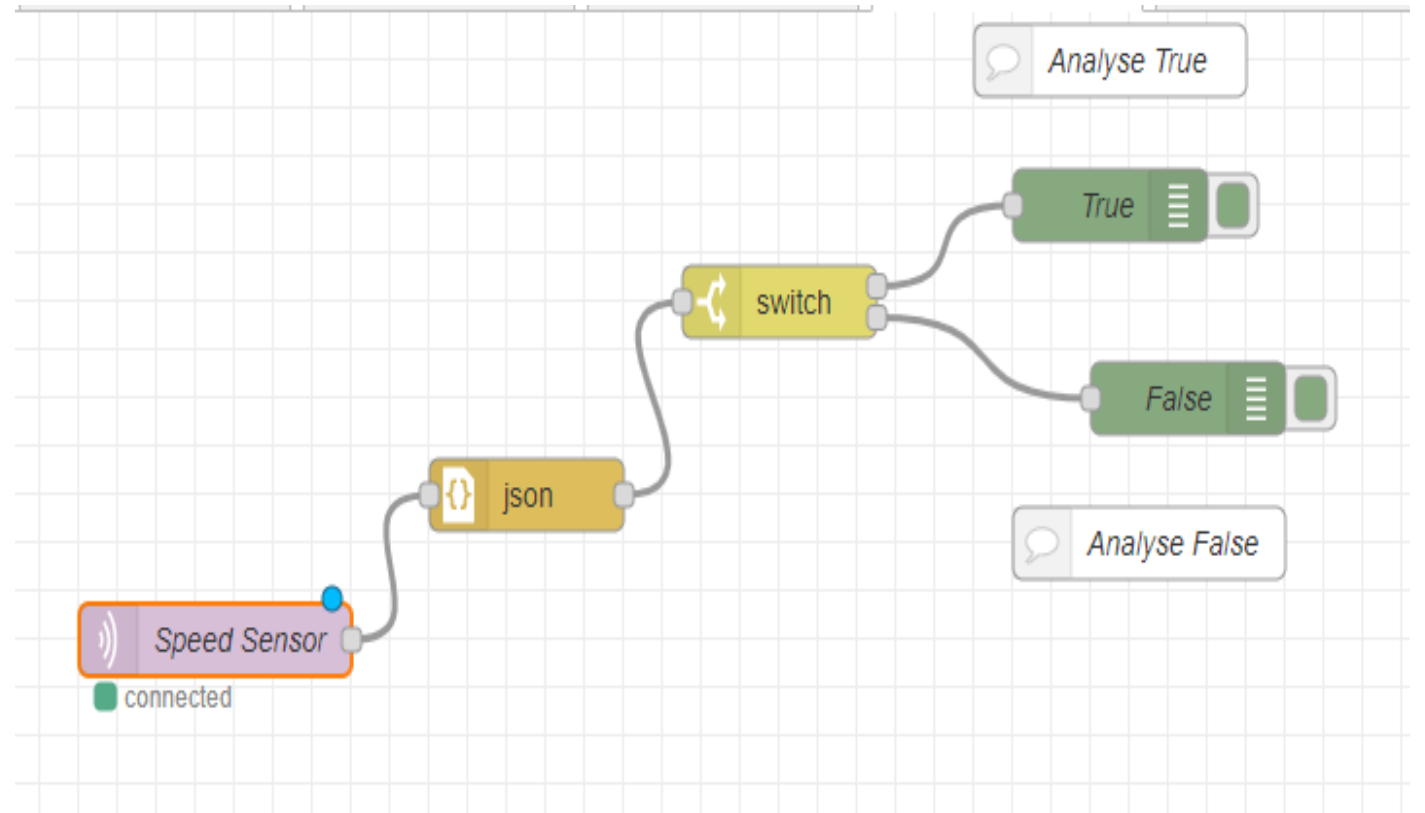
Payload

Current value

Enabled

Flow : Switch Node and Comment Node

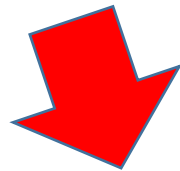
- Insert **MQTT** in Node
- Insert **Json** Node
- Insert **Switch** Node
- Insert **Debug** Node
- Insert **Comment** Node



MQTT and Json

- MQTT in Node configuration (Refer pervious Session)
 - **Server : broker.mqttdashboard.com**
 - **Client ID AND Topic : Speed Sensor**
- Json

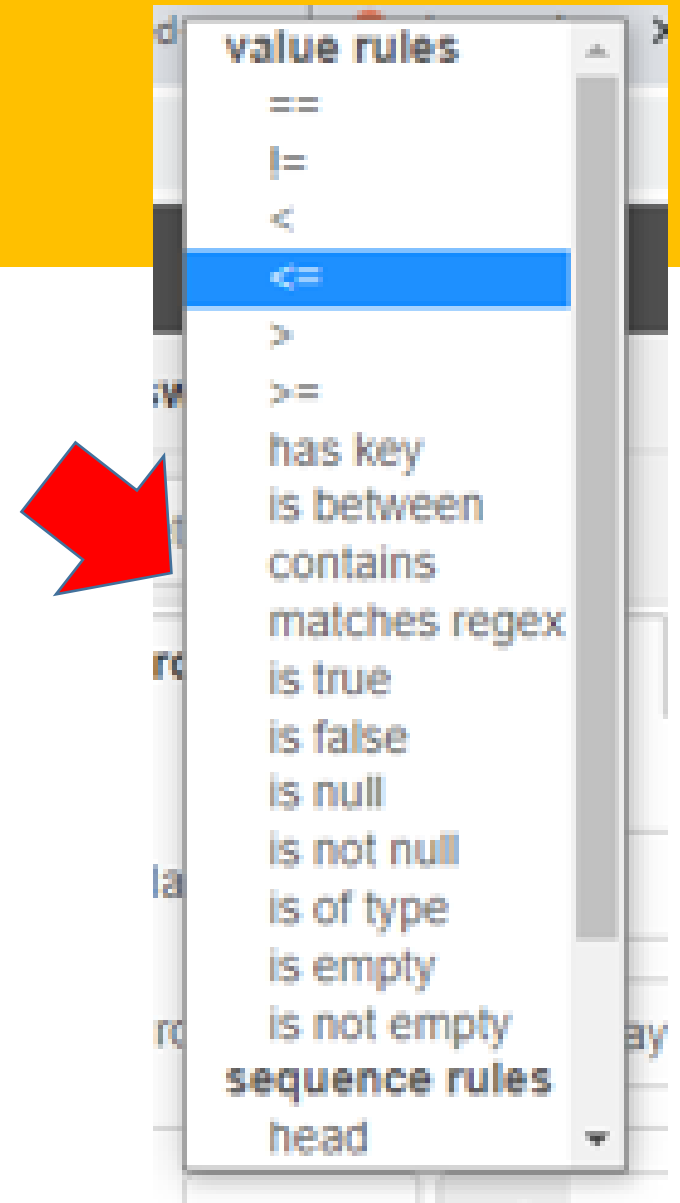
Reference Link for MQTT



- <https://www.youtube.com/watch?v=LCYIFoyBn2I&t=419s>

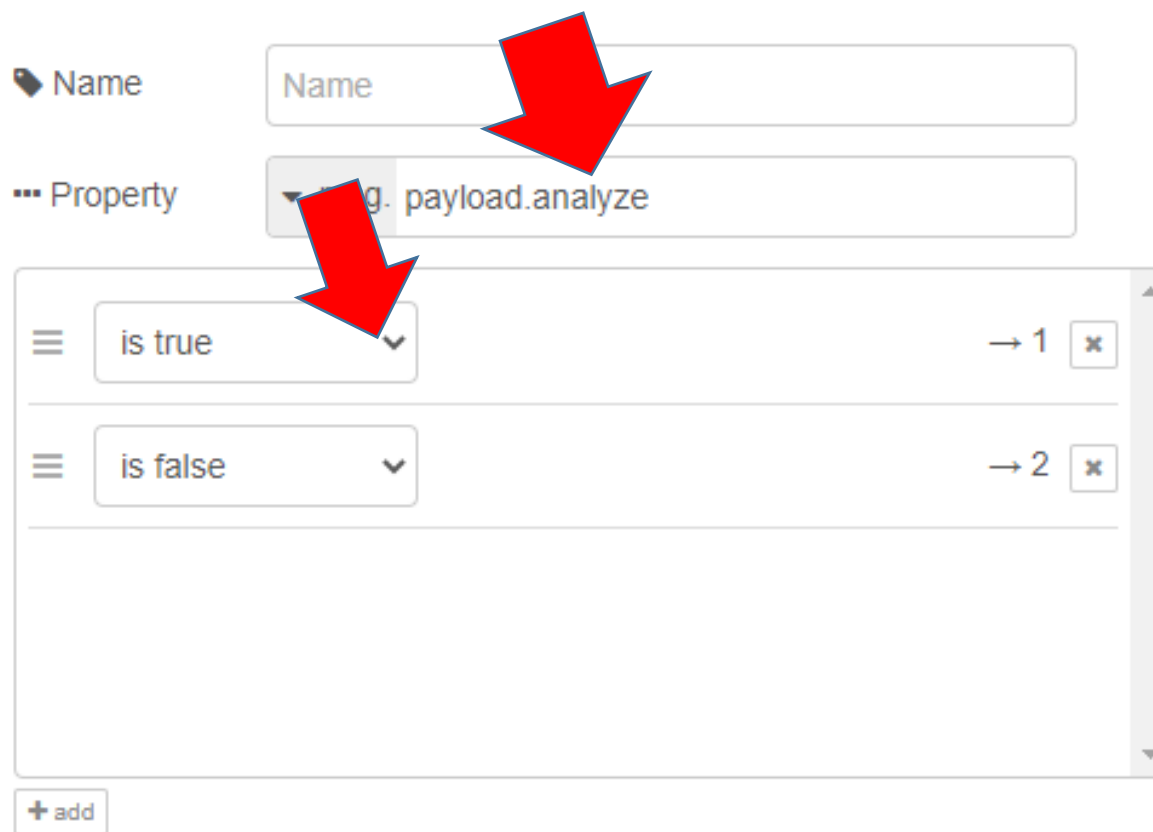
Switch Node

- Switch Node helps to **'switch'** or **route messages** depending on the incoming message properties.
- Example: Check the `msg.payload.analyze` property and, depending on its value (true/false), **decide to route a message** to one of the switch node's outputs.



Switch Node : Edit Properties

- **Property :payload.analyze**



The screenshot shows the configuration interface for a Switch Node. It features a 'Name' field containing the text 'Name' and a 'Property' dropdown menu with the selected value 'g. payload.analyze'. Below these fields is a list of conditions. The first condition is 'is true' with a right-pointing arrow and the number '1', and a close button 'x'. The second condition is 'is false' with a right-pointing arrow and the number '2', and a close button 'x'. At the bottom left of the list is a '+ add' button. Two large red arrows are overlaid on the image: one points to the 'Name' field, and the other points to the 'Property' dropdown menu.

Name

Property

g. payload.analyze

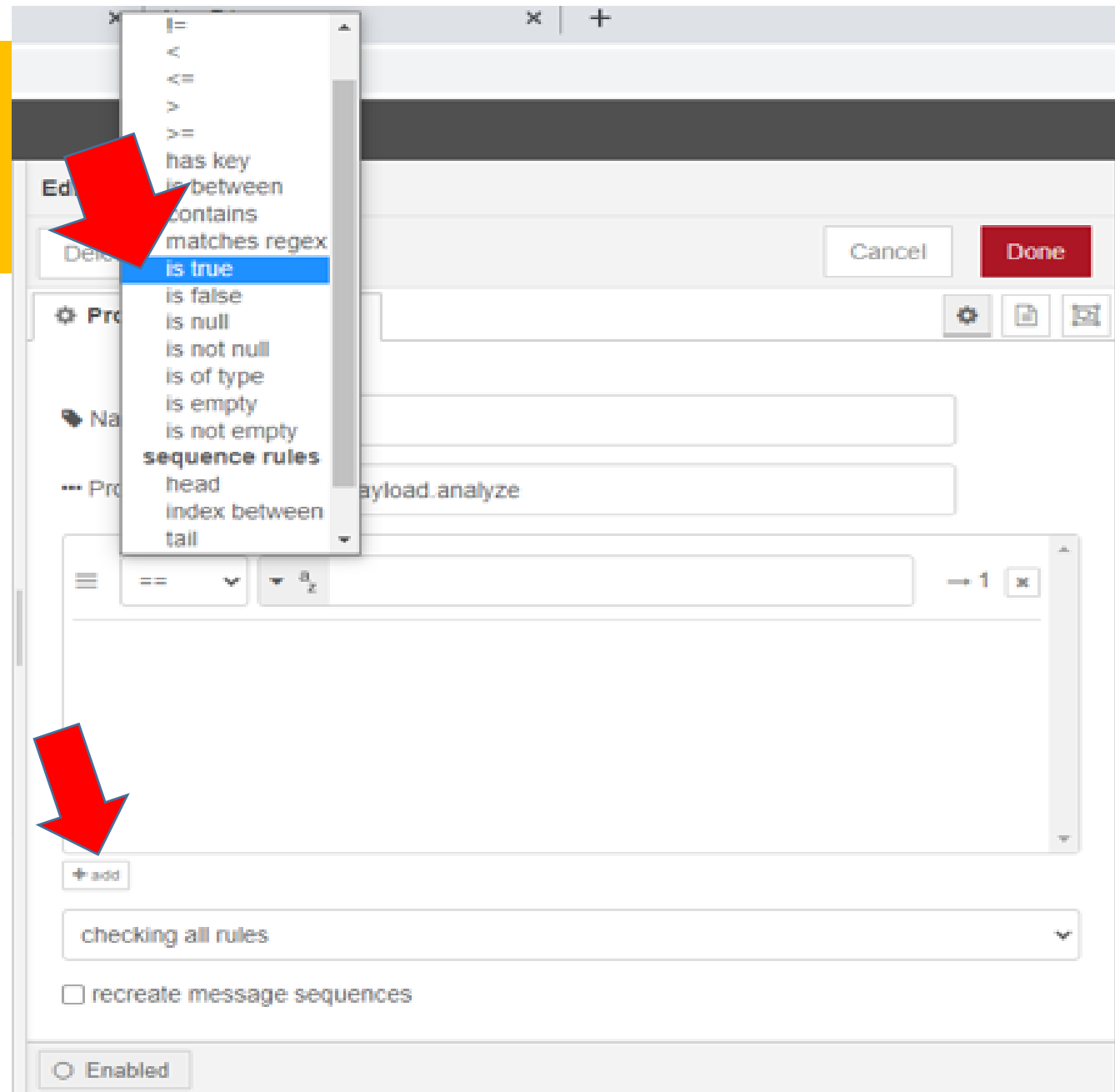
is true → 1 x

is false → 2 x

+ add

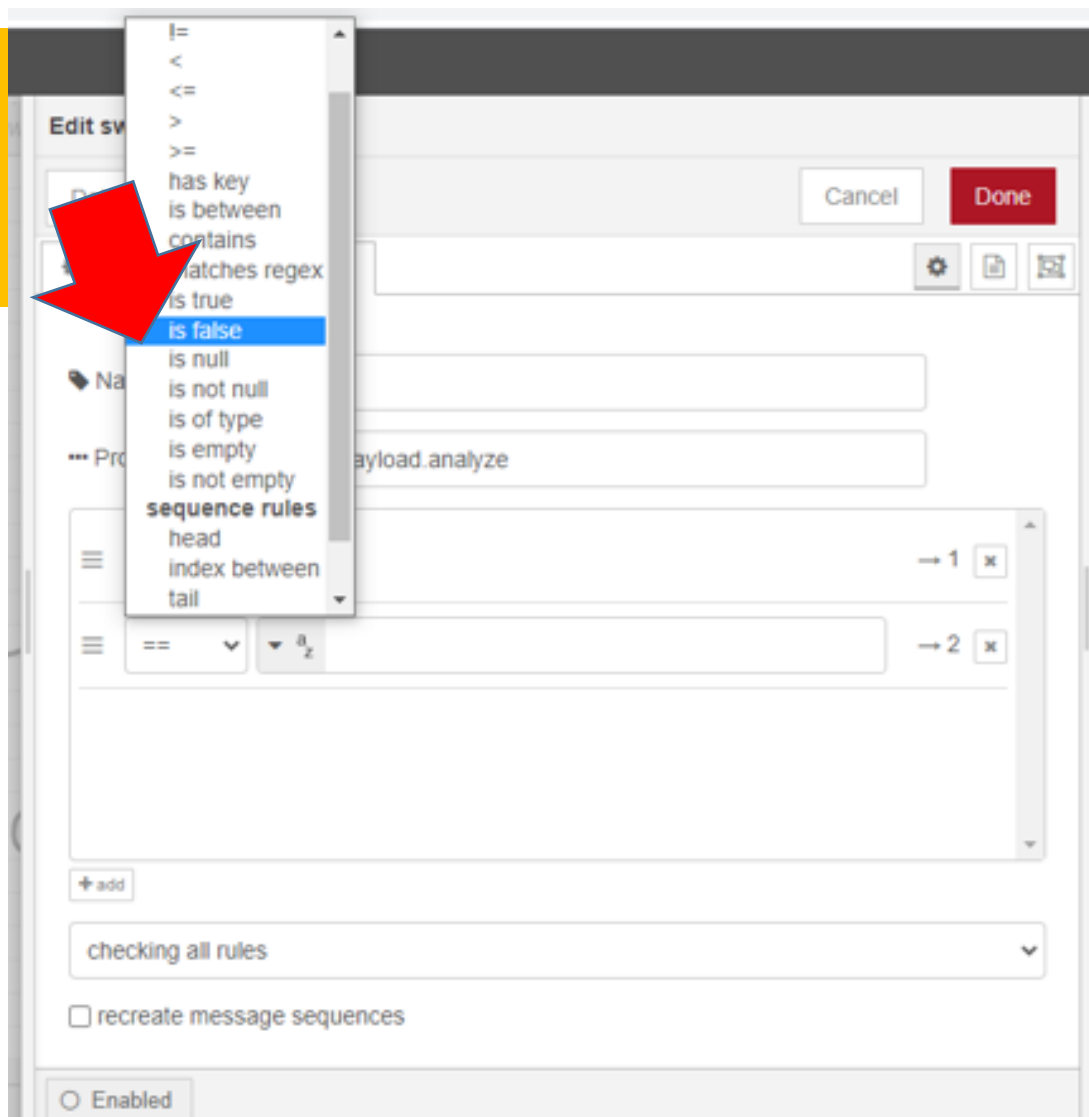
Switch Node

- Property
 - **Is True**
- Click **Add**



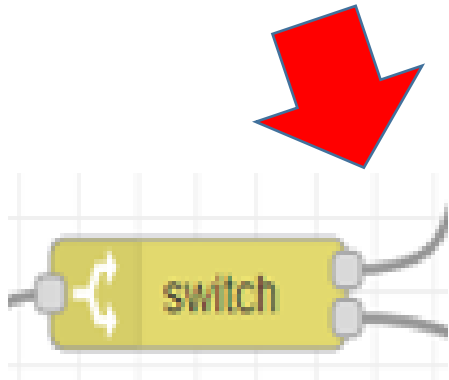
Switch Node

- Property
 - Is False



Switch Node

- Click Done



Edit switch node

Delete Cancel Done

⚙ Properties 📄 🖨

👉 Name

⋮ Property

☰	<input type="text" value="is true"/>	→ 1	✕
☰	<input type="text" value="is false"/>	→ 2	✕

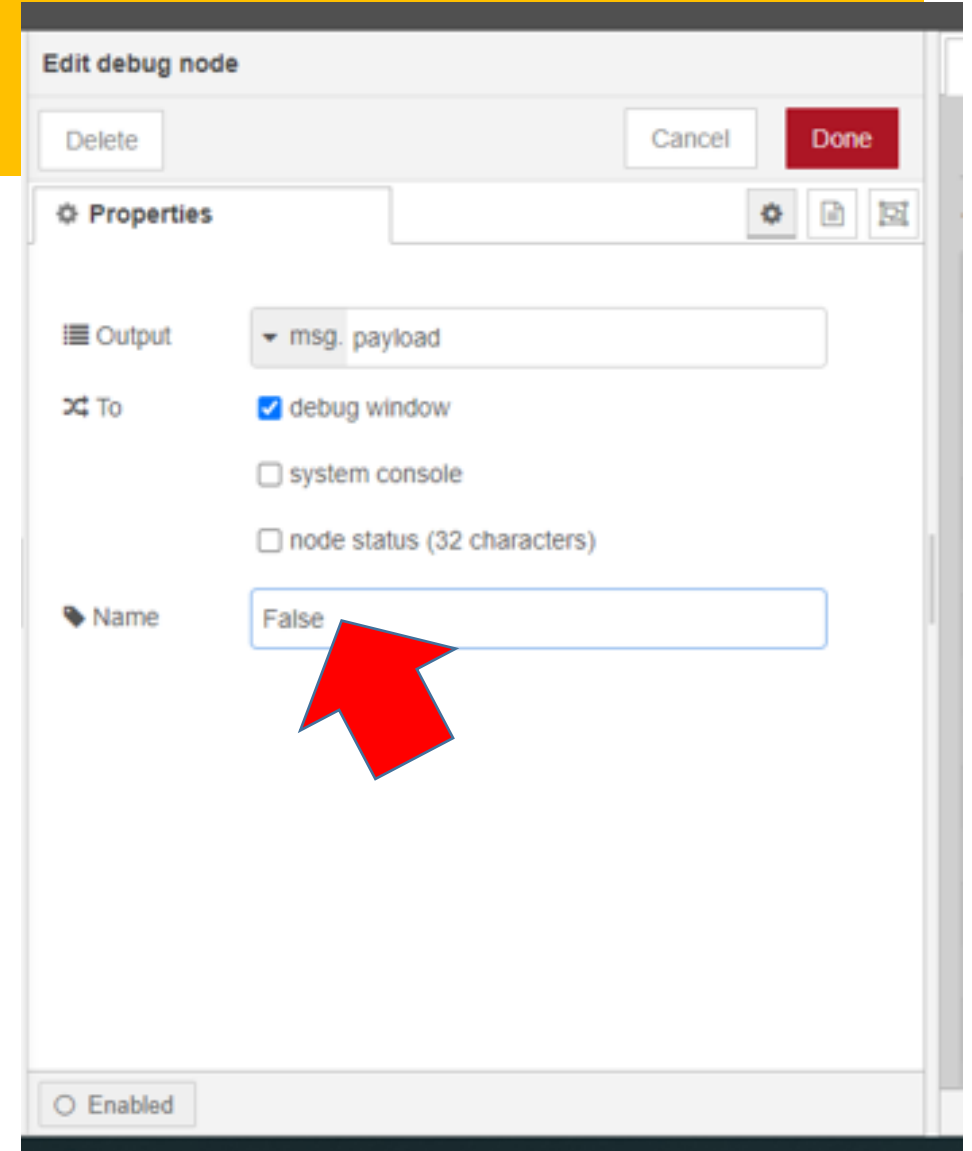
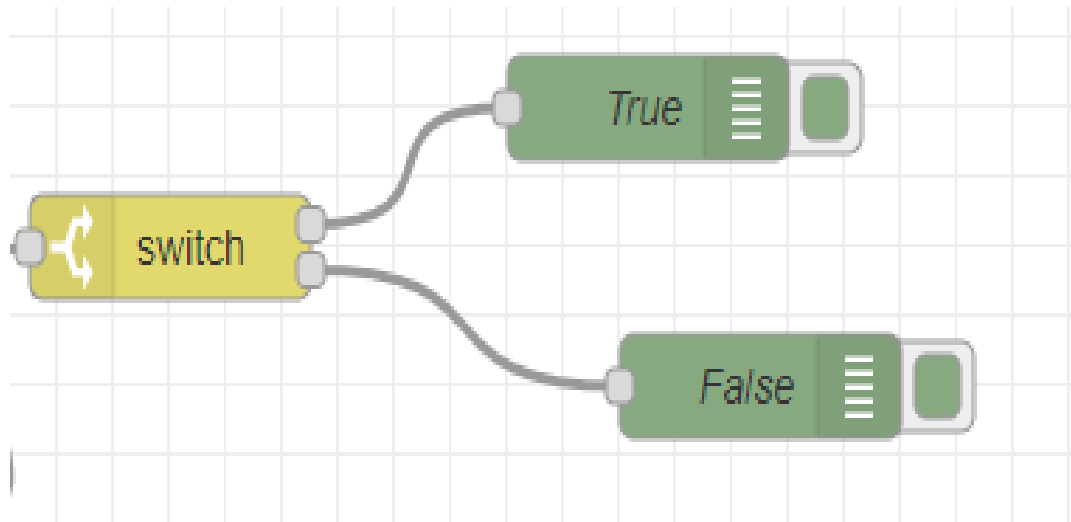
+ add

recreate message sequences

Enabled

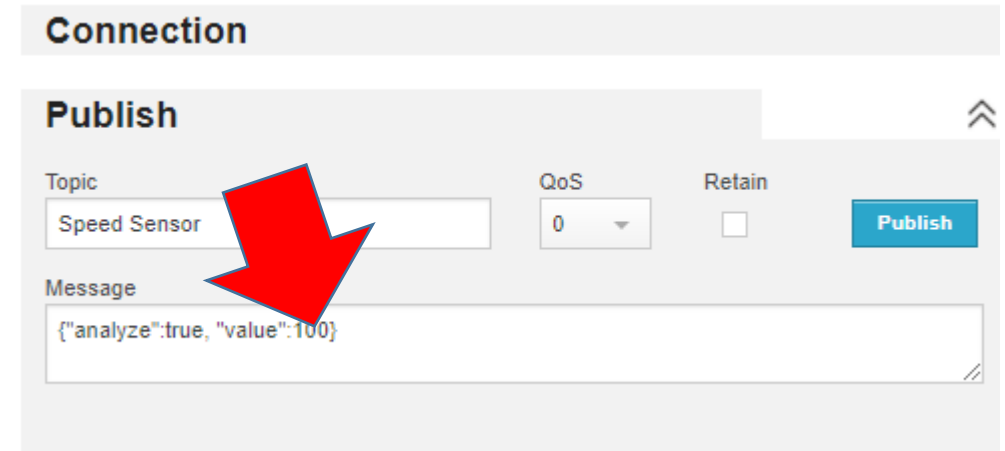
Debug Node Properties

- Name : False (Analyze)



HiveMQ

- Establish Connection
- Send Json messages through Publish
- **`{"analyze":false, "value":10}`**

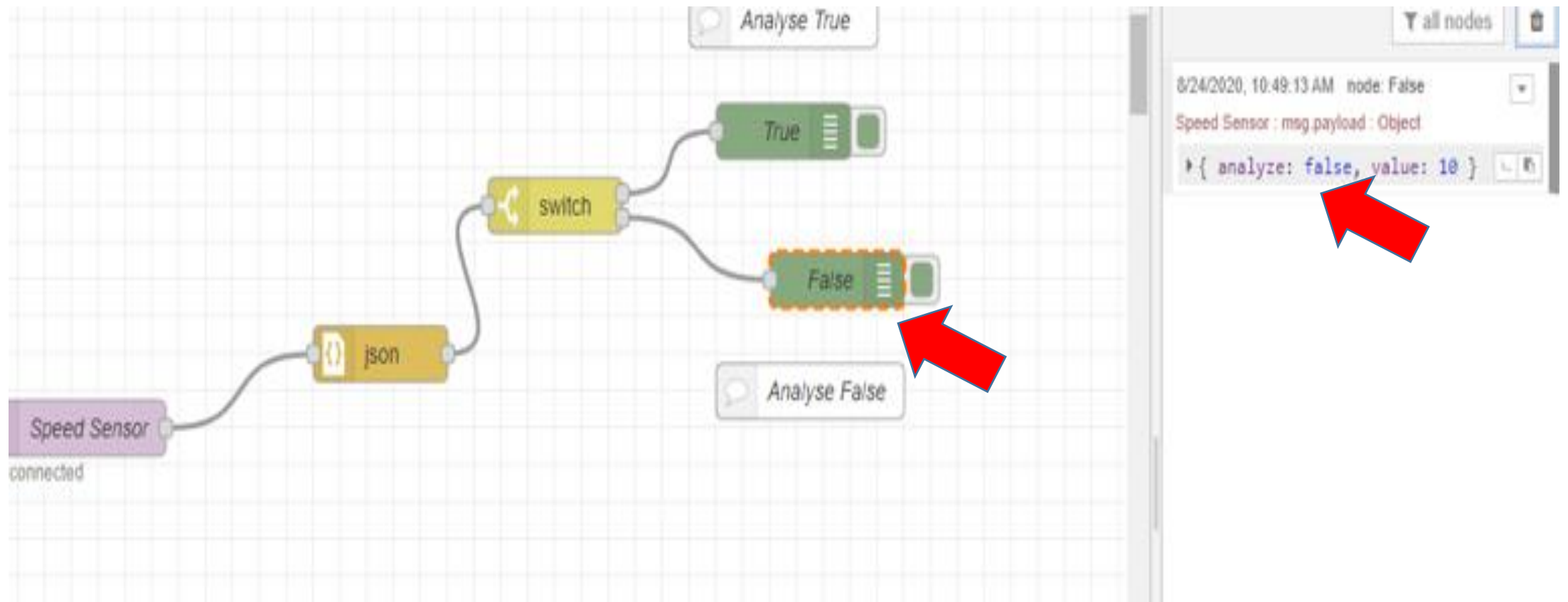


The screenshot shows the 'Publish' tab in the HiveMQ web interface. The 'Topic' field contains 'Speed Sensor'. The 'QoS' dropdown is set to '0'. The 'Retain' checkbox is unchecked. A blue 'Publish' button is visible. The 'Message' text area contains the JSON string `{"analyze":true, "value":100}`. A large red arrow points from the 'Message' field back to the red JSON string in the list above.

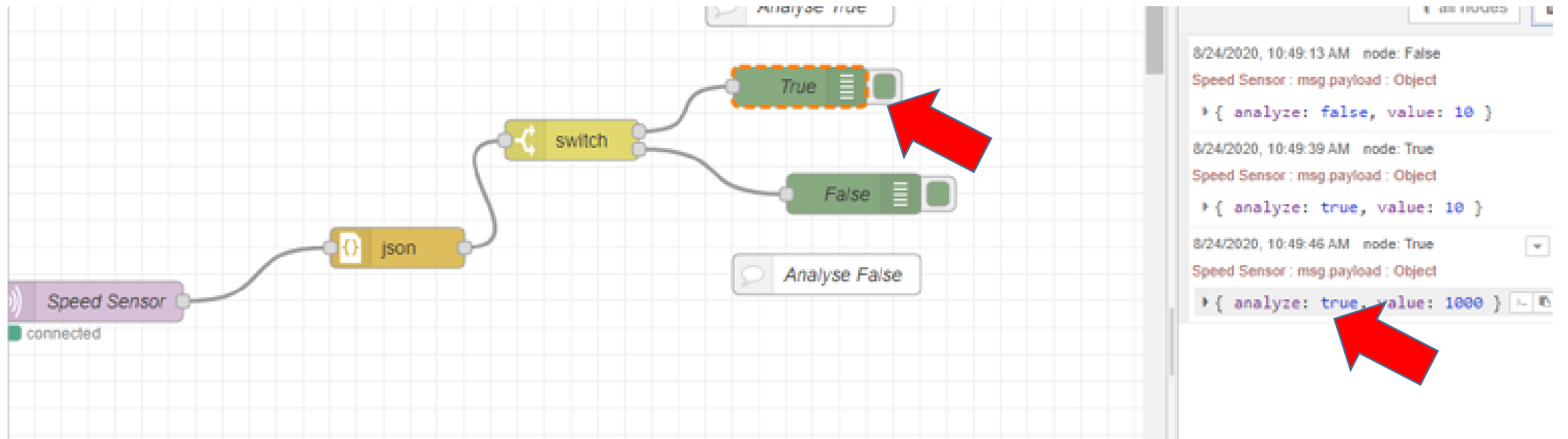
Reference Link for MQTT 

- <https://www.youtube.com/watch?v=LCYIFoyBn2I&t=419s>

Output in Debug Monitor: False Path

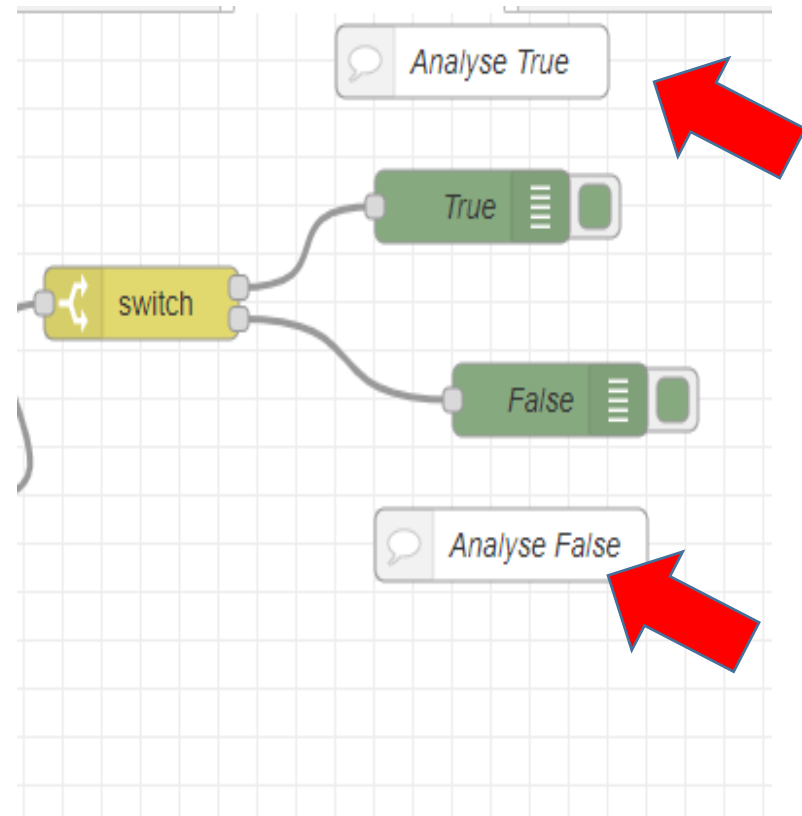


Output in Debug Monitor: True Path

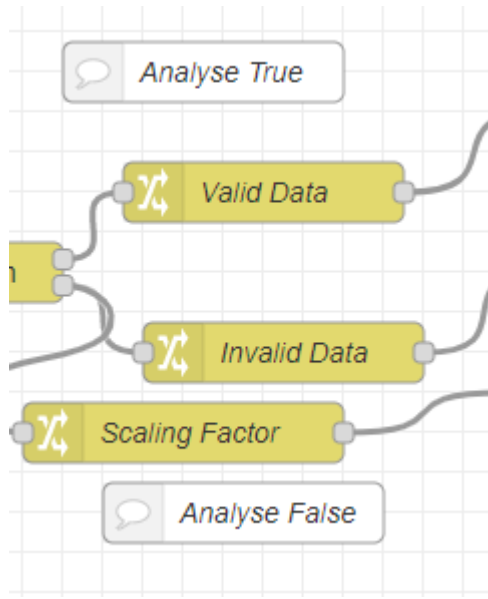


Comment Node

- Comment Nodes are useful during **designing** complex flows.



Properties of Comment Node

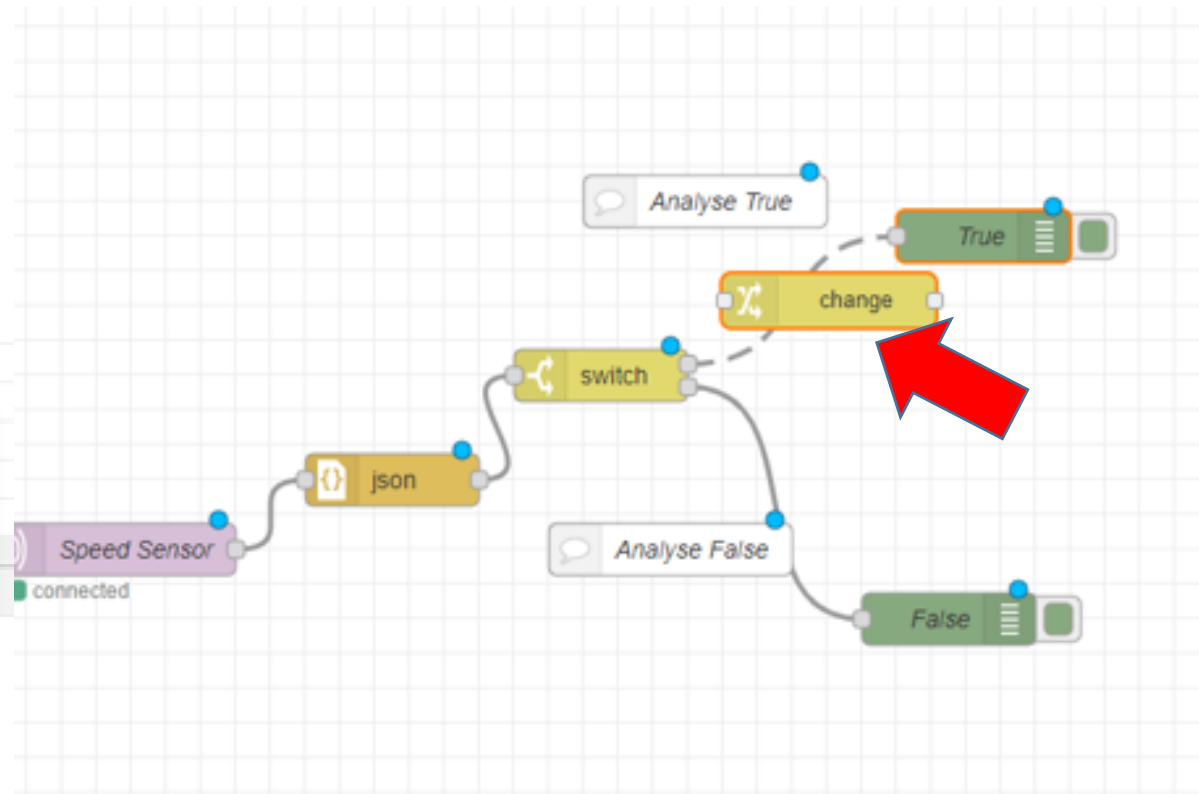


The screenshot shows the "Edit comment node" dialog box. At the top, there are "Delete", "Cancel", and "Done" buttons. Below is a "Properties" section with a gear icon and a help icon. The "Name" field contains the text "Analyse True". Below the name field is a rich text editor toolbar with buttons for "h1", "h2", "h3", "h4", "h5", "h6", "B" (bold), "I" (italic), "code", "list", "ul", "ol", "li", "link", "unlink", and "undo". A red arrow points to the "h1" button. Below the toolbar is a text area containing the number "1".

Change Node

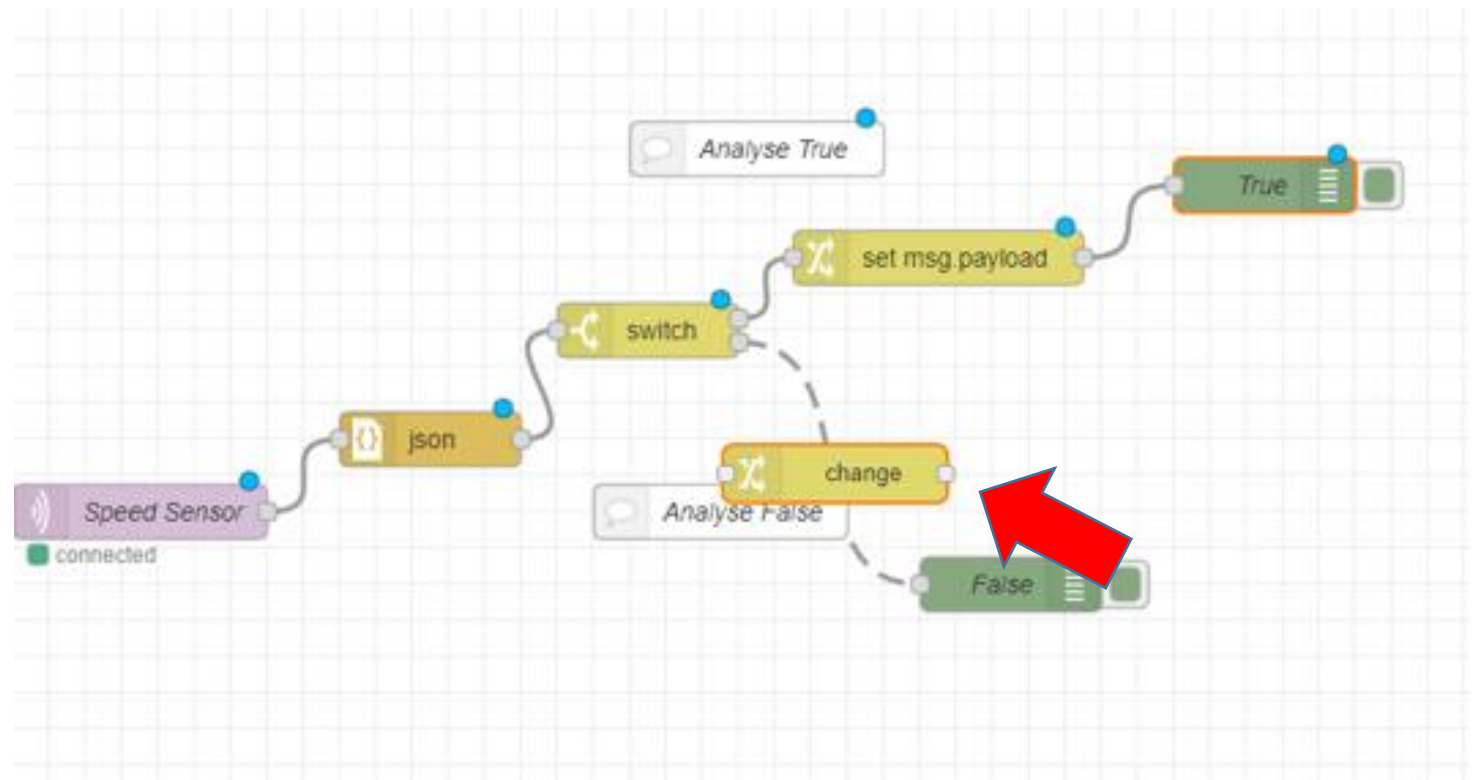
- Change node will allow to **change a message payload or add new properties**
- It helps to **perform and send analysis** based on the received data
- Change node will affect the properties in a message:
 - **either by changing existing ones**
 - **deleting them or**
 - **adding new properties.**

Insert Change Node

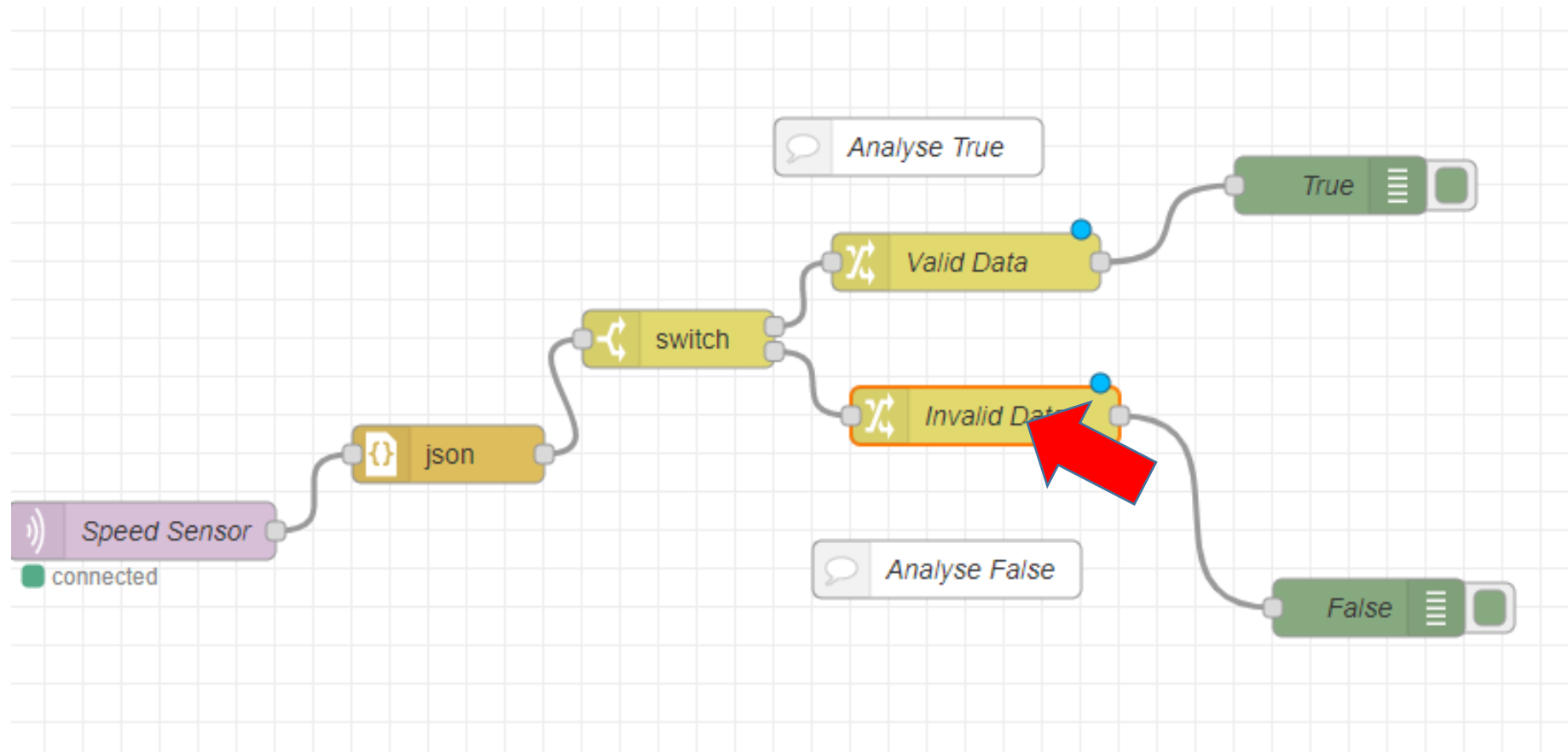


Insert Change Node

- Insert Change Node in False path Analyze



Insert Change Node



Insert Change Node

- Set Property
- Done

Dialog box titled "Edit change node" with buttons: Delete, Cancel, Done.

Section: Properties

Name: [Name]

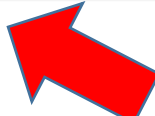
Section: Rules

Rule: Set msg. payload to Speed Data received is being analyze

Buttons: +add, Enabled

Deploy and Publish

```
8/24/2020, 11:12:47 AM node: True  
Speed Sensor : msg.payload : string[37]  
"Speed Data received is being  
analyzed"
```



Connection

Publish

Topic

Speed Sensor

QoS

0

Retain

Publish




Message

```
{"analyze":true, "value":100}
```


Publish : Analyze False

```
8/24/2020, 11:14:16 AM node: False  
Speed Sensor : msg.payload : string[35]  
"Invalid: Speed Data is not analyzed"
```



Connection

Publish

Topic

Speed Sensor


QoS

0

Retain

Message

{"analyze":false, "value":5}



msg.payload.note

- payload.note
- Done
- **Deploy and Publish**

```
Speed Sensor : msg.payload : Object
  ▶ { analyze: false, value: 1000,
    note: "Invalid: Speed Data is not
    ana..." }
```

Edit change node

Delete Cancel Done

⚙ Properties

Name Invalid Data

Rules

Set msg.payload.note

to Invalid: Speed Data is not analyzed

+ add