

# **Production Supporting Systems in Factories**

**ระบบสนับสนุนการผลิตในโรงงานอุตสาหกรรม**

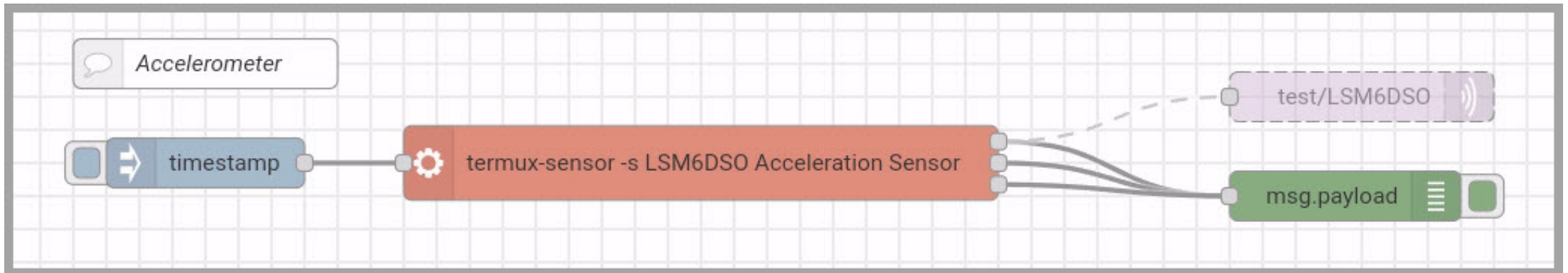
# Smart Sensors

# Smart sensors

- A smart sensor is a device that
  - takes input from the physical environment
  - perform predefined functions upon detection of specific input
  - then process data before passing it on.
- Your mobile phone running `Node-Red` can be programmed to be very **smart** sensors.

# **Module 4-1: Mobile sensors**

- Use Node-Red from a mobile phone.
- Flow
  - inject , exec , debug , mqtt out



- **exec node**
  - Command: `termux-sensor -s LSM6DS0 Acceleration Sensor`
    - *Your sensor name will be different.*
  - Output: `while the command is running ....`

The screenshot shows a dialog box titled "Edit exec node". At the top, there are three buttons: "Delete", "Cancel", and "Done". Below the buttons is a tabbed interface with the "Properties" tab selected. The "Properties" tab contains several configuration fields:

- Command:** A text field containing the command `termux-sensor -s LSM6DS0 Acceleration Sensor`.
- Append:** A checkbox labeled "Append" is unchecked. To its right is a text field containing `msg. payload`.
- extra input parameters:** A text field containing the text `extra input parameters`.
- Output:** A dropdown menu with the selected option being `while the command is running - spawn mode`.
- Timeout:** A text field containing `optional` followed by the label `seconds`.
- Hide console:** A checkbox labeled "Hide console" is unchecked.
- Name:** A text field containing the text `Name`.

- mqtt out node
  - Server : Create new server similar to M3-1
  - Topic : test/LSM6DS0
    - *Your topic will be different.*
  - QoS : 1

**Edit mqtt out node**

Delete Cancel Done

**Properties**

Server Prodsup

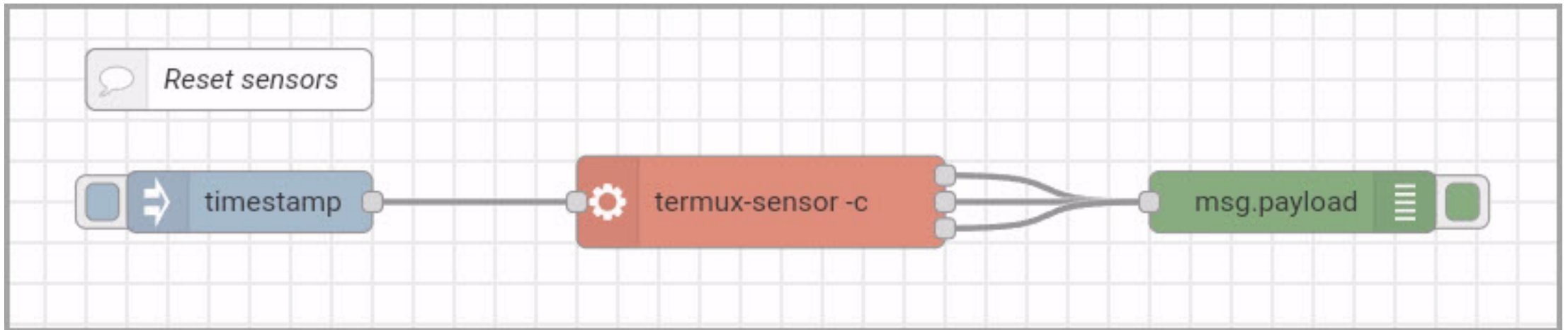
Topic test/LSM6DS0

QoS 1 Retain true

Name Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

- Resetting sensors





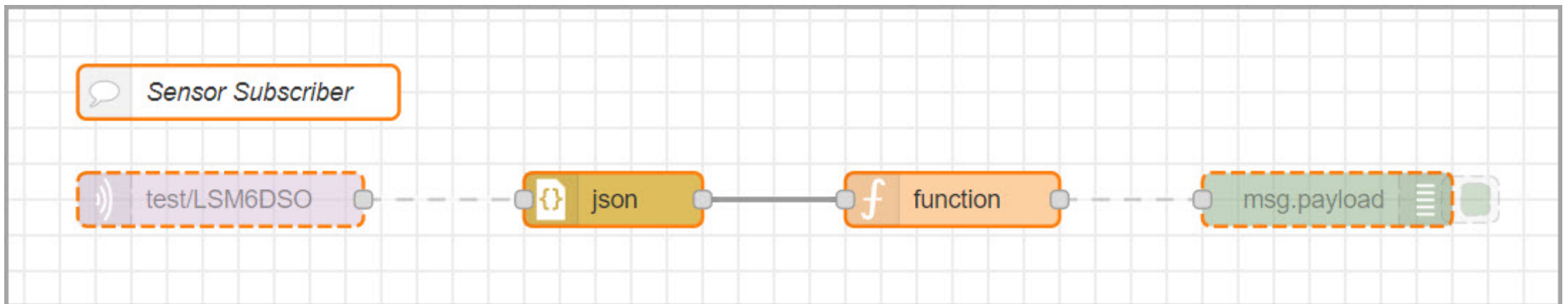
- **exec node**
  - Command: `termux-sensor -c`
  - Output: when the command is complete ...

The screenshot shows a dialog box titled "Edit exec node". At the top, there are three buttons: "Delete", "Cancel", and "Done". Below the buttons is a tabbed interface with a "Properties" tab selected. The "Properties" tab contains several configuration fields:

- Command:** A text field containing the text "termux-sensor -c".
- + Append:** A checkbox that is currently unchecked, followed by a text field containing "msg. payload".
- extra input parameters:** A text field containing the text "extra input parameters".
- Output:** A dropdown menu with the selected option "when the command is complete - exec mode".
- Timeout:** A text field containing "optional" followed by the word "seconds".
- Hide console:** A checkbox that is currently unchecked.
- Name:** A text field containing the text "Name".

# **Module 4-2: Sensor Subscriber**

- Use `Node-Red` from your computer
- Flow
  - `mqtt in`, `json`, `function`, `debug`



- mqtt in node
  - Topic : test/LSM6DSO
  - QoS : 1

**Edit mqtt in node**

Delete Cancel Done

**Properties**

Server ProdSup

Action Subscribe to single topic

Topic test/LSM6DSO

QoS 1

Output auto-detect (string or buffer)

Name Name

- json node
  - No need to adjust anything.

The screenshot shows a dialog box titled "Edit json node". At the top, there are three buttons: "Delete", "Cancel", and "Done". Below these is a tab labeled "Properties" with a gear icon. To the right of the tab are three icons: a gear, a document, and a square with a circle. The main area contains three fields: "Action" with a dropdown menu showing "Convert between JSON String & Object", "Property" with a text input containing "msg. payload", and "Name" with a text input containing "Name". At the bottom, there is a section titled "Object to JSON options" with a checkbox labeled "Format JSON string".

**Edit json node**

Delete Cancel Done

⚙️ **Properties** ⚙️ 📄 🖼️

🎯 Action Convert between JSON String & Object ▼

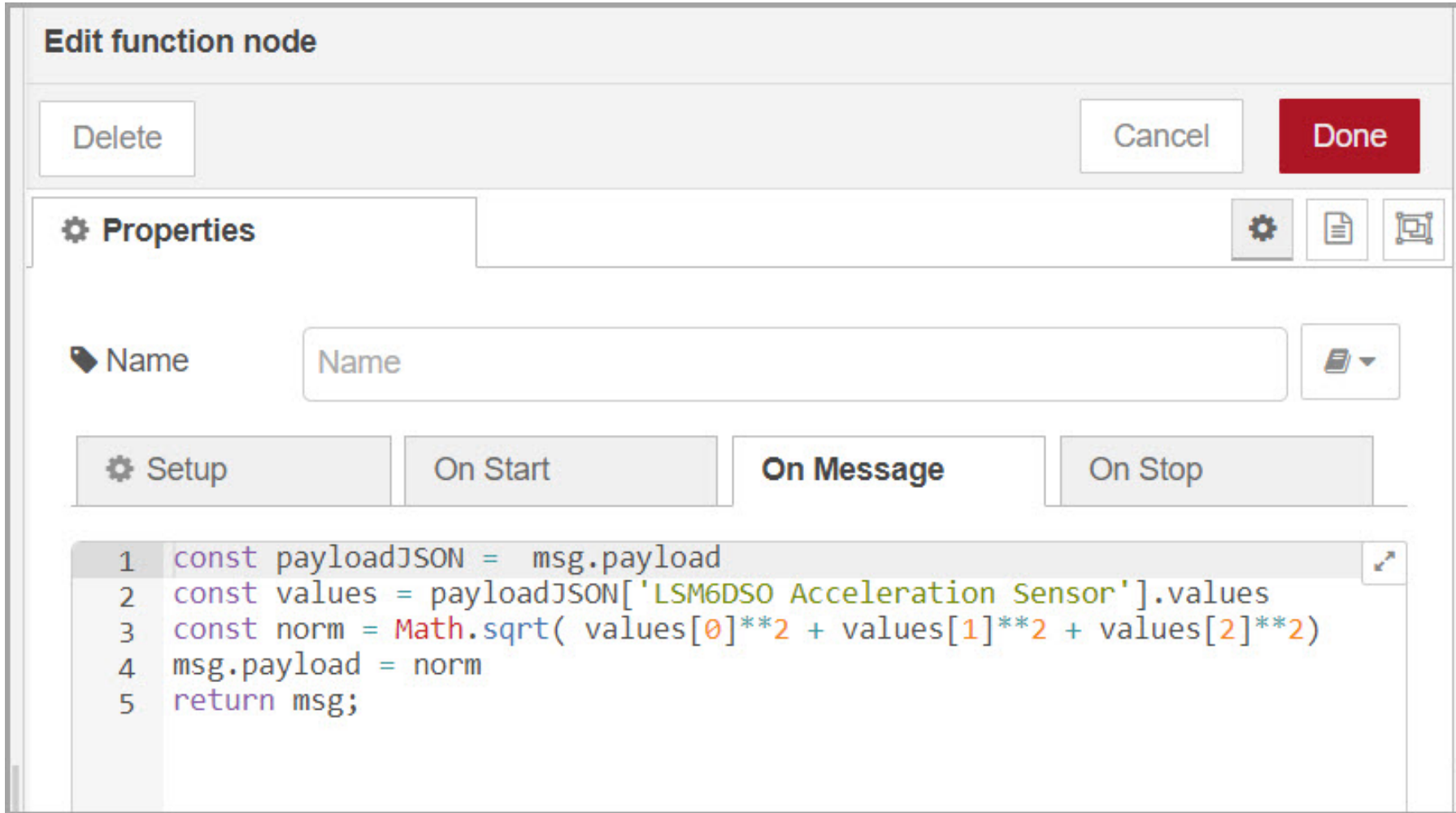
⋮ Property msg. payload

🏷️ Name Name

Object to JSON options

☐ Format JSON string

- `function` node (code on the next page)

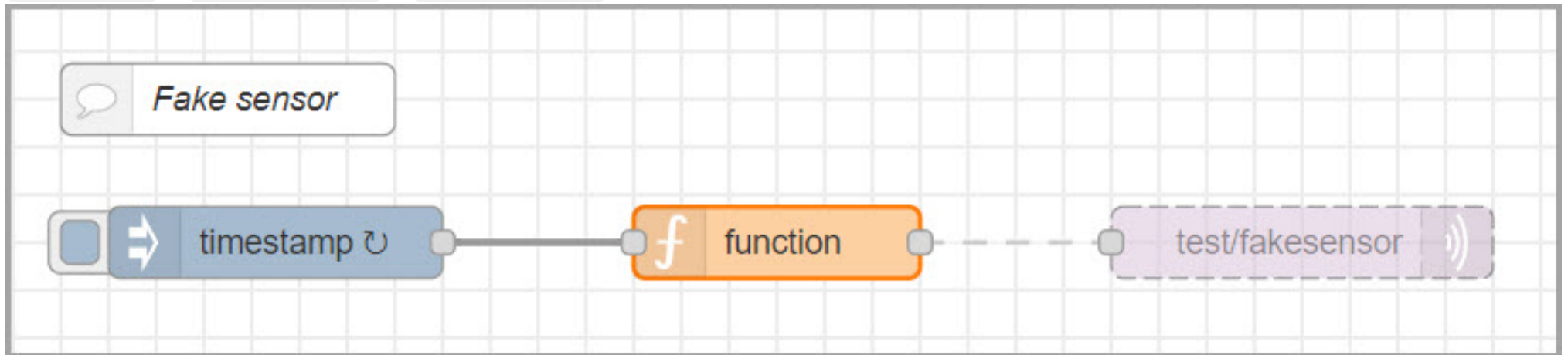


```
const payloadJSON = msg.payload;
const values = payloadJSON['LSM6DSO Acceleration Sensor'].values;
const norm = Math.sqrt(values[0] ** 2 + values[1] ** 2 + values[2] ** 2);
msg.payload = norm;
return msg;
```

# **Module 4-3: Fake sensors**



- If you cannot use a mobile phone to send sensor data, you can create a fake sensor data from `Node-Red` in your computer.
- Flow
  - `inject`, `function`, `mqtt out`



- inject node
  - Repeat : every 1 second

### Edit inject node

Delete Cancel Done

⚙ Properties

🔑 Name

≡ msg. payload = timestamp ×

≡ msg. topic = a<sub>z</sub> ×

+ add inject now

☐ Inject once after 0.1 seconds, then

🔄 Repeat interval ▾

every 1 seconds ▾

- **function node**

**Edit function node**

Delete

Cancel

Done

**Properties**

Name

Setup

On Start

**On Message**

On Stop

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




```
msg.payload = Math.random();  
return msg;
```


- mqtt out node
  - Topic: test/fakesensor
  - QoS: 1

### Edit mqtt out node


DeleteCancelDone


#### Properties




 Server

ProdSup




 Topic


test/fakesensor

 QoS

1

 Retain

true

 Name

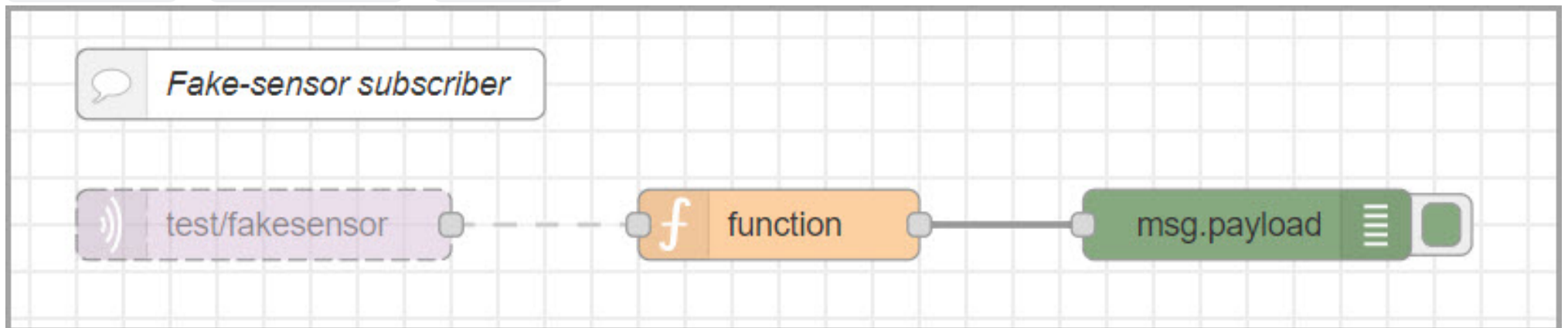
Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

- Lastly, we can listen to the sensors.

- Flow

- mqtt in , function , debug



- mqtt in node
  - Topic : test/fakesensor
  - QoS : 1

### Edit mqtt in node

DeleteCancelDone

Properties

Server

ProdSup

Action

Subscribe to single topic

Topic

test/fakesensor

QoS

1

Output

auto-detect (string or buffer)

Name

Name

- `function` node

Edit function node

Delete

Cancel

Done

⚙️ Properties

⚙️

📄

🔗

🏷️ Name

Name

📄 ▼

⚙️ Setup

On Start

**On Message**

On Stop

1

2

3

```
const payload = msg.payload;
msg.payload = parseFloat(payload);
return msg;
```

↗️



```
const payload = msg.payload;  
msg.payload = parseFloat(payload);  
return msg;
```