

# **Production Supporting Systems in Factories**

**ระบบสนับสนุนการผลิตในโรงงานอุตสาหกรรม**

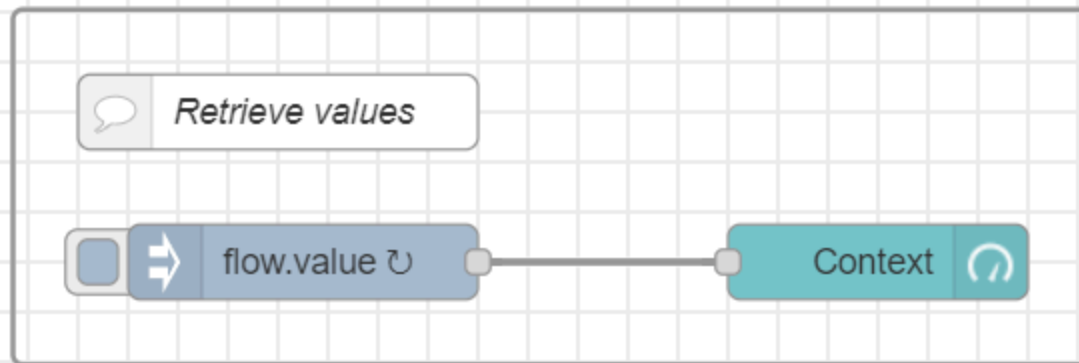
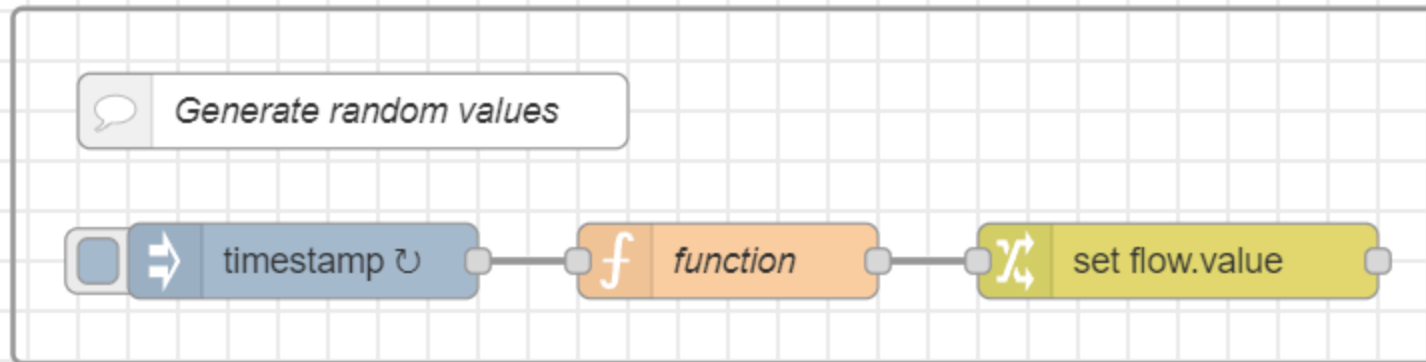
# Context

# Scope

The `scope` of a particular context value.

- `Node` - only visible to the node that set the value
- `Flow` - visible to all nodes on the same flow (or tab in the editor)
- `Global` - visible to all nodes

**Use `flow` context to pass values**



# Upper inject node

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 1 seconds

📄 ☐ Enabled

## function node

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

# change node

### Edit change node

Delete

Cancel

Done

⚙️ Properties

⚙️

📄

🔗

🏷️ Name

Name

☰ Rules

Set

▼

▼ flow. value

☰

to the value

▼ msg. payload

☐ Deep copy value

✕

+ add

📄

☐ Enabled

8



# Lower inject node

**Edit inject node**

Delete Cancel Done

**Properties**

Name

☐ msg. payload =

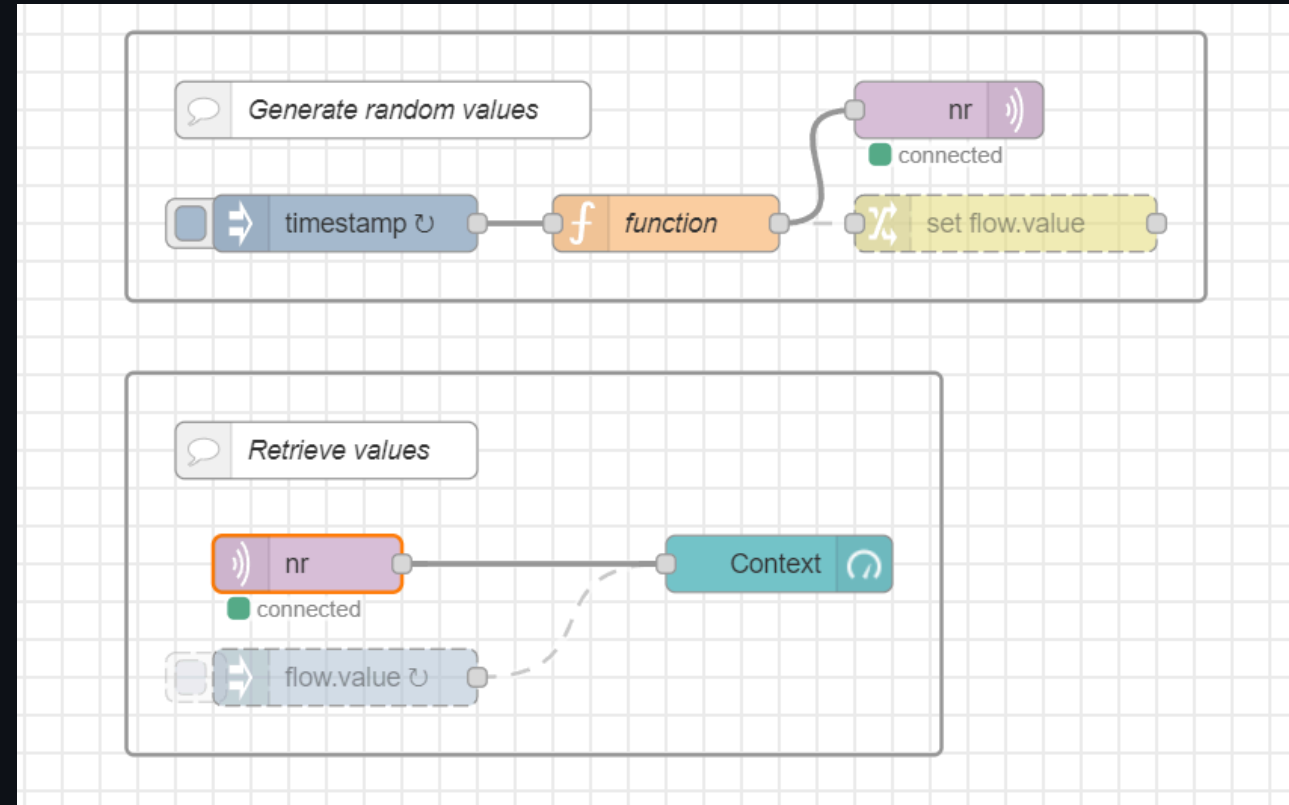
☐ msg. topic =

☐ Inject once after  seconds, then

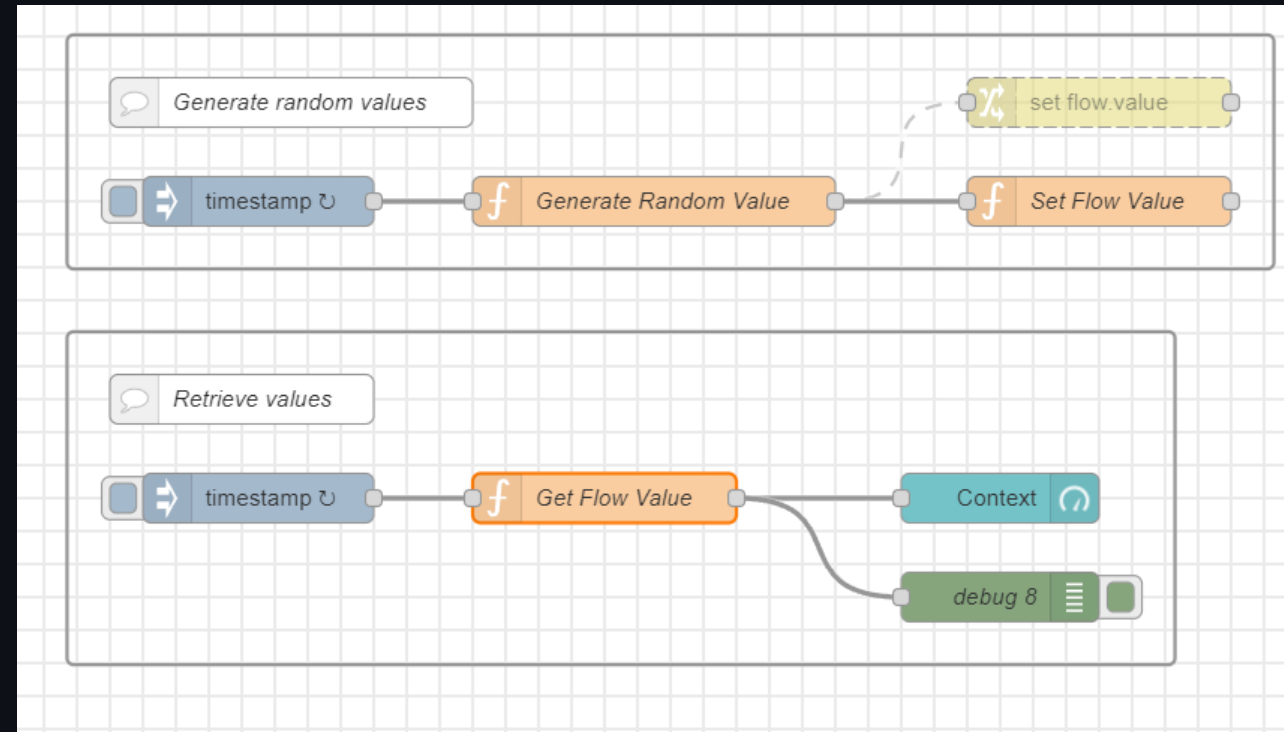
every

## Side note

- You can use MQTT protocol to pass "context" information.
  - This is actually a better way.



# Using code



## Set Flow Value node

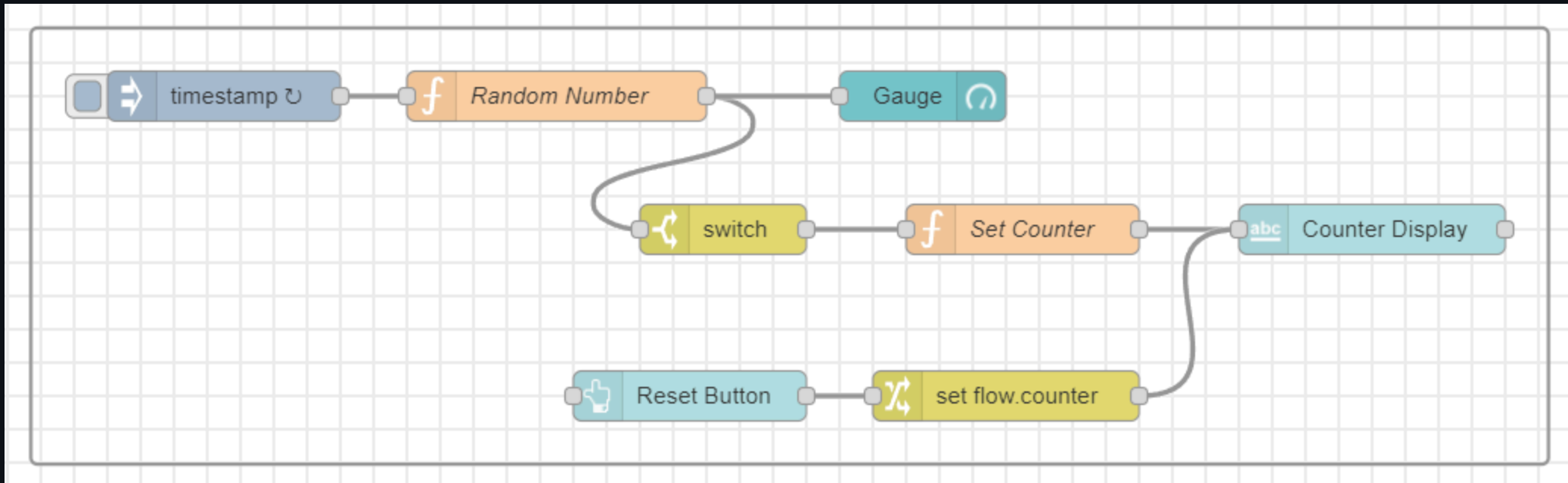
```
flow.set("value", msg.payload);
```

## Get Flow Value node

```
const value = flow.get("value") ?? 0;  
msg.payload = value;  
return msg;
```

# Counter with reset button

# Flow



## Random Number (function) node

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

# Switch node

Edit switch node

Delete

Cancel

Done

⚙ Properties

⚙ 📄 🖨

📌 Name

Name

⋮ Property

▼ msg. payload

☰ >= ▼ ▼ 0.5 0.5 → 1 ✕

+ add

checking all rules ▼

☐ recreate message sequences

📄 ☐ Enabled



## Set Counter (Function) node

```
const curCounter = flow.get("counter") ?? 0;  
flow.set("counter", curCounter + 1);  
msg.payload = curCounter + 1;  
return msg;
```

# button node

Edit button node

Delete Cancel Done

⚙ Properties

Group [Context] Group 1

Size auto

Icon optional icon

Label Reset Button

Tooltip optional tooltip

Color optional text/icon color

Background optional background color

☒ When clicked, send:

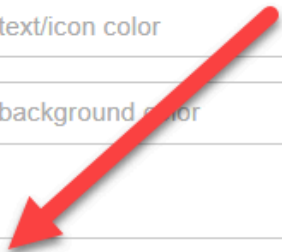
Payload ▾ 0

Topic ▾ msg. topic

➔ If msg arrives on input, emulate a button click: ☐

</> Class Optional CSS class name(s) for widget

Name Name



# change node

Edit change node

Delete Cancel Done

⚙ Properties

Name Name

☰ Rules

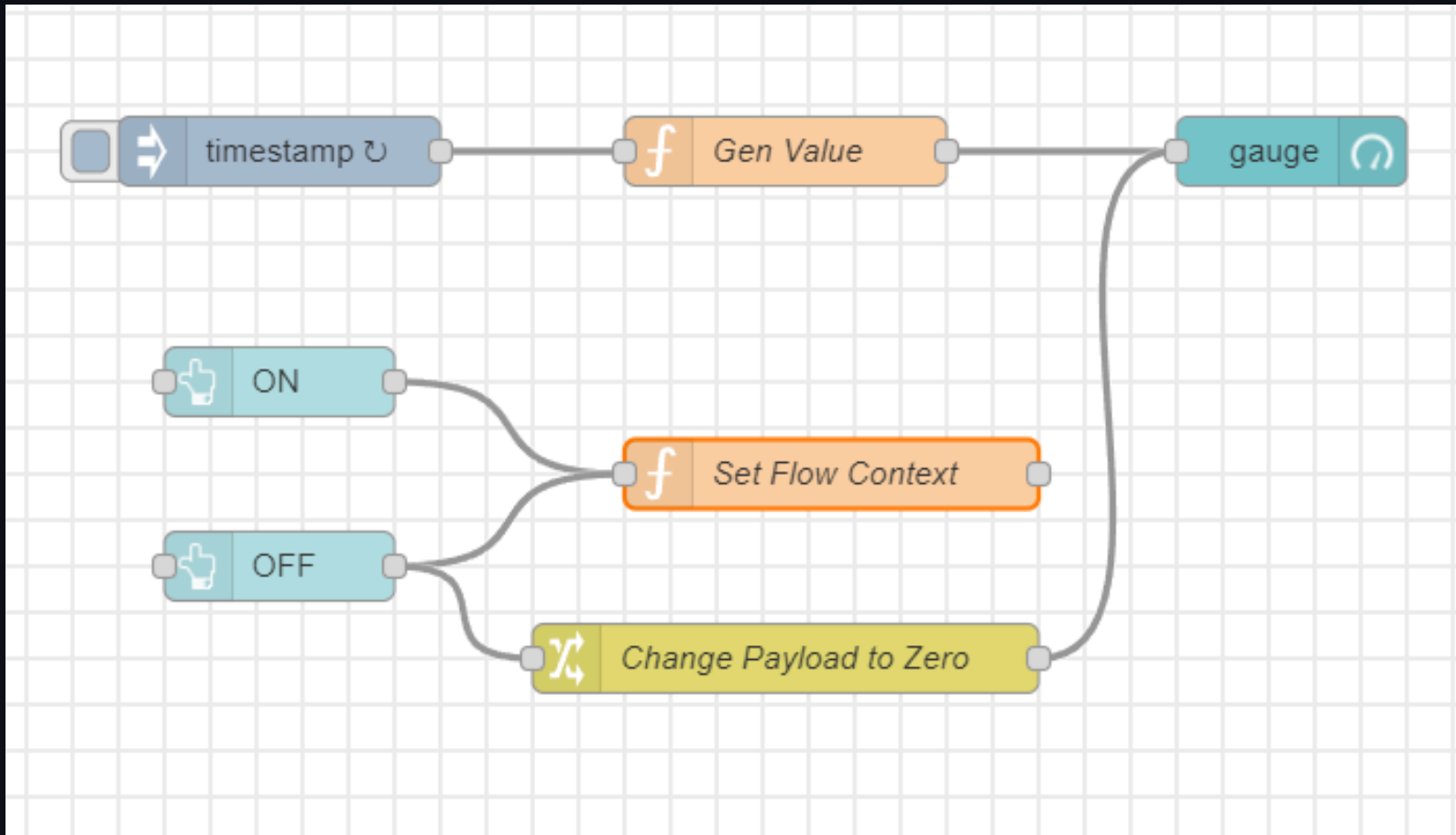
Set ▼ ▼ flow. counter

☰ to the value ▼ msg. payload ✕

☐ Deep copy value

# Turning sensor display on and off

# Flow



# Gen Value node

- Testing

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

- Actual

```
const status = flow.get("status") ?? "OFF";  
if (status === "ON") {  
  msg.payload = Math.random();  
  return msg;  
} else {  
  // Do nothing for now.  
}
```

# ON button

Edit button node

Delete Cancel Done

⚙ Properties

Group [ON OFF] Group 1

Size auto

Icon optional icon

Label ON

Tooltip optional tooltip

Color optional text/icon color

Background green

When clicked, send:

Payload

Topic

→ If msg arrives on input, emulate a button click: ☐

Class

Name

# OFF button

Edit button node

Delete Cancel Done

Properties

Group [ON OFF] Group 1

Size auto

Icon optional icon

Label OFF

Tooltip optional tooltip

Color optional text/icon color

Background red

When clicked, send:

Payload  $a_z$  OFF

Topic msg. topic

→ If msg arrives on input, emulate a button click: ☐

Class Optional CSS class name(s) for widget

Name Name



# Change node

**Edit change node**

Delete Cancel Done

**Properties**

Name Change Payload to Zero

**Rules**

Set ▼ msg. payload

to the value ▼ 0<sub>9</sub> 0