

Node-RED

Nico De Witte

bereikbaar via nico.dewitte@vives.be

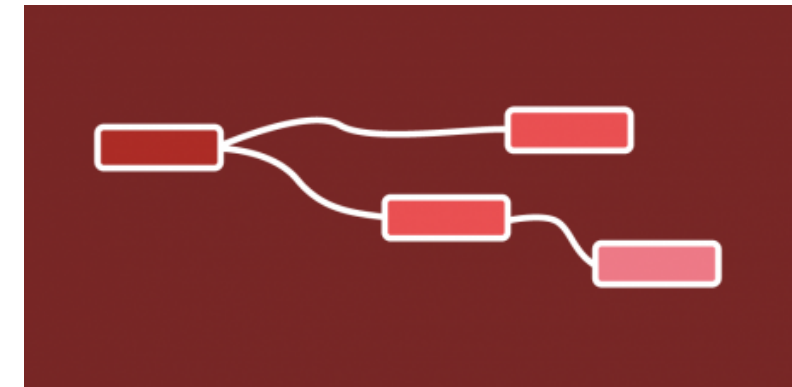
Doel

Binnenhalen van onze data uit de cloud (The Things Network) naar onze eigen services.



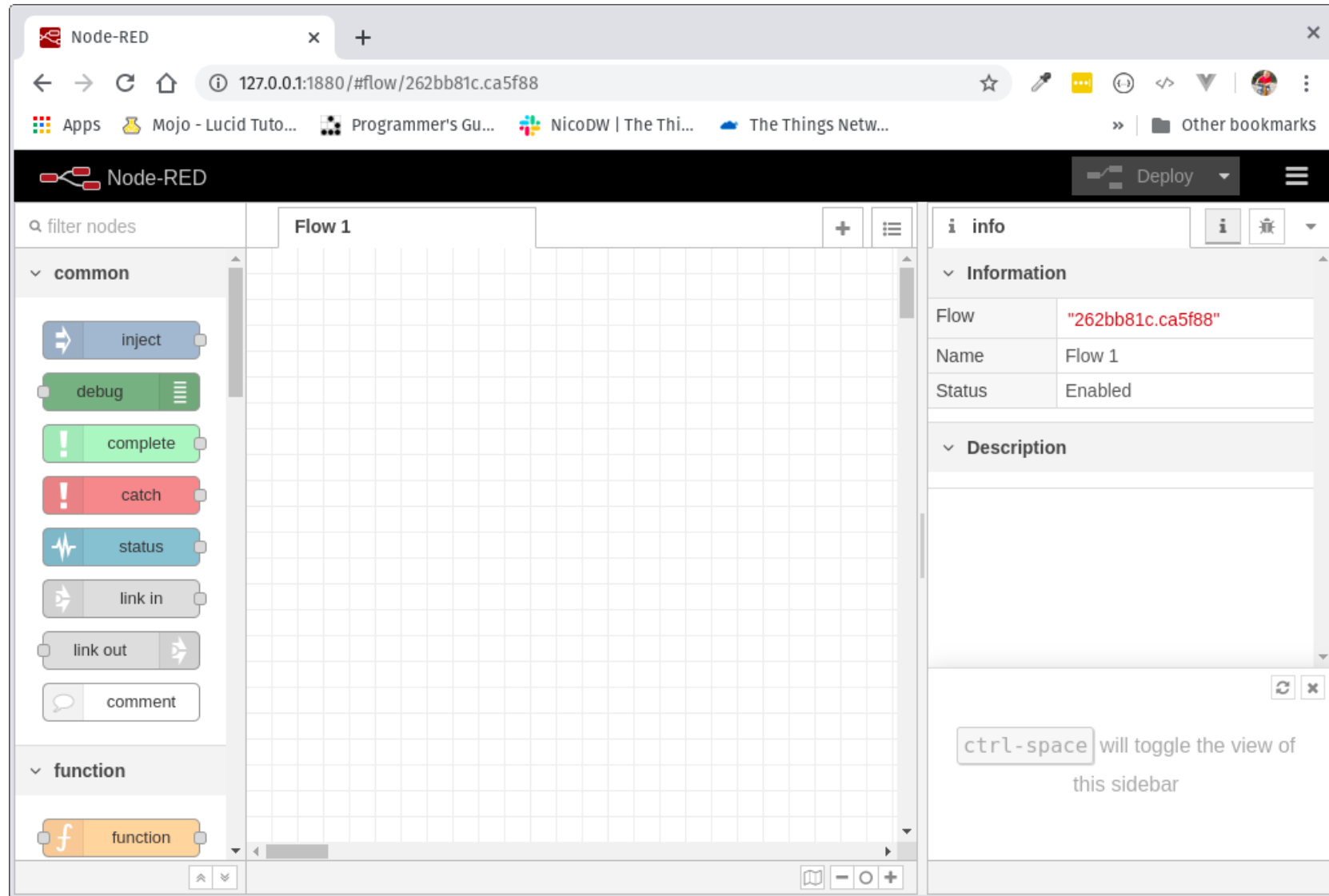
Node-RED

Node-RED is een flow gebaseerde visuele ontwikkelomgeving ontwikkeld door IBM. Het laat toe hardware toestellen, API's en online diensten met elkaar te communiceren in het teken van het Internet of Things.



Node-RED

Het biedt een browsergebaseerde editor die het gemakkelijk maakt om flows met elkaar te verbinden met behulp van het brede scala aan nodes die met een klik kunnen worden ingezet tijdens de uitvoering.



Node-RED

Flows kunnen tevens heel makkelijk worden geëxporteerd of geïmporteerd als JSON.

Node.js gebaseerd

Node.js is een **open-source, platform-onafhankelijke** JavaScript runtime omgeving die JavaScript-code buiten een browser uitvoert.

De lichtgewicht runtime is gebouwd op Node.js en profiteert volledig van het event-based, non-blocking async model. Dit maakt het ideaal om aan de rand van het netwerk te draaien op goedkope hardware zoals de Raspberry Pi en in de cloud.

Met meer dan 225.000 modules in de packet repository van Node, is het eenvoudig om gamma aan nodes uit te breiden en zo nieuwe mogelijkheden toe te voegen.

Installeren van Node.js

Start met installeren van Node.js. Surf hiervoor naar <https://nodejs.org/en/>.

Selecteer de LTS (Long Term Support) versie (12.16).

Volg de wizard om Node.js te installeren.

Testen van Node.js

Open een `powershell` venster en type volgende command:

```
node --version
```

De output zo ongeveer het volgende moeten zijn:

```
v12.16.1
```


Installeren van Node-RED

Om Node-RED te installeren dien je volgende command uit te voeren in `powershell` :

```
npm install -g --unsafe-perm node-red
```

Herstart nu voor alle zekerheid je computer.

Starten van Node-RED

Node-RED dient te worden opgestart. Dit kan je realiseren door onderstaand commando uit te voeren in een `powershell` venster:

```
node - red
```

Dit dien je telkens opnieuw te doen als je Node-RED wil gebruiken.

```
node-red
C:\Users\nicod> node-red
12 Mar 17:00:06 - [info]

Welcome to Node-RED
=====

12 Mar 17:00:06 - [info] Node-RED version: v1.0.4
12 Mar 17:00:06 - [info] Node.js version: v12.16.1
12 Mar 17:00:06 - [info] Windows_NT 10.0.18362 x64 LE
12 Mar 17:00:07 - [info] Loading palette nodes
12 Mar 17:00:08 - [info] Settings file : C:\Users\nicod\.node-red\settings.js
12 Mar 17:00:08 - [info] Context store : 'default' [module=memory]
12 Mar 17:00:08 - [info] User directory : C:\Users\nicod\.node-red
12 Mar 17:00:08 - [warn] Projects disabled : editorTheme.projects.enabled=false
12 Mar 17:00:08 - [info] Flows file : C:\Users\nicod\.node-red\flows_DESKTOP-K8P4FEV.json
12 Mar 17:00:08 - [info] Creating new flow file
12 Mar 17:00:08 - [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.

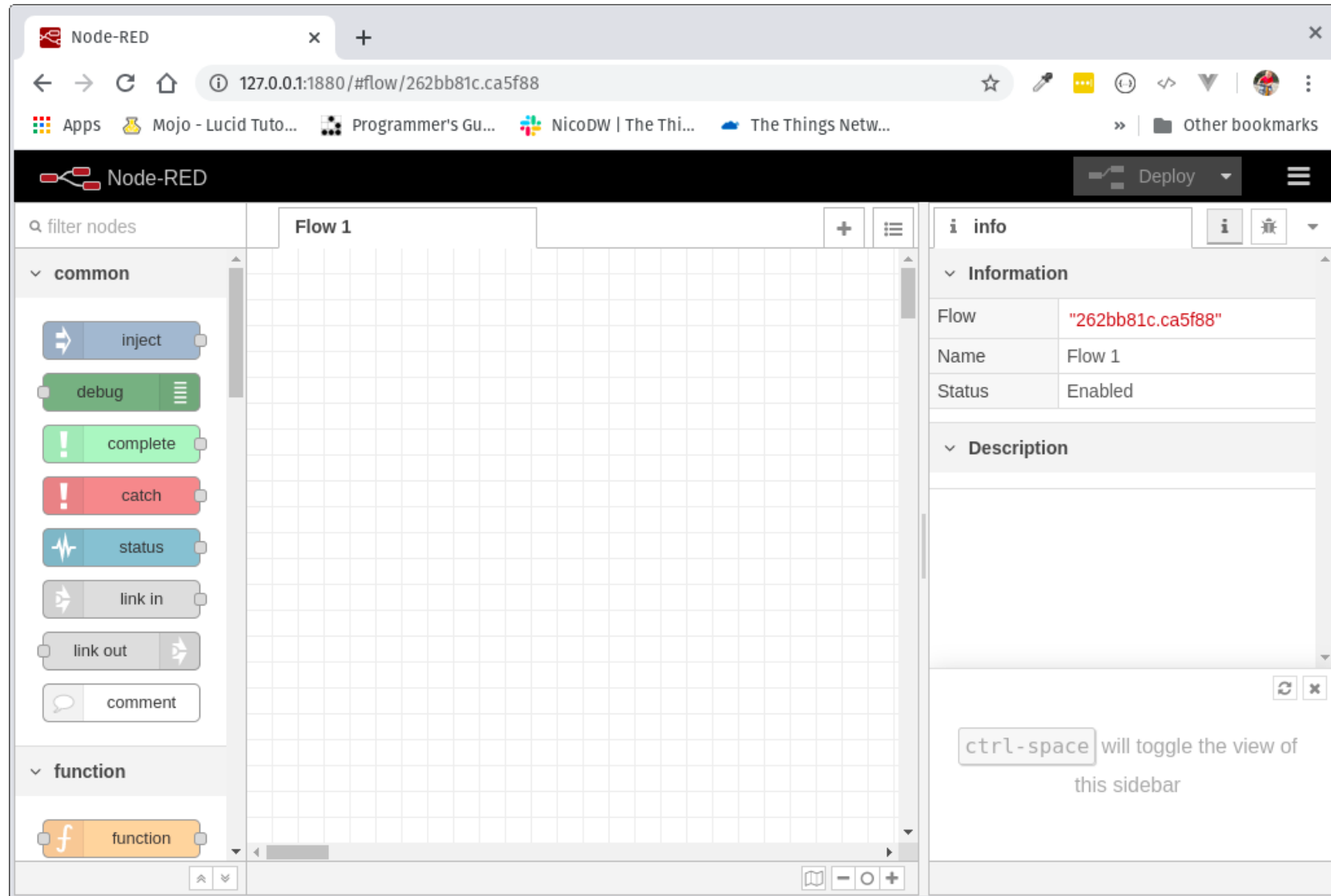
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

12 Mar 17:00:08 - [info] Server now running at http://127.0.0.1:1880/
12 Mar 17:00:08 - [info] Starting flows
12 Mar 17:00:08 - [info] Started flows
```

Werken met Node-RED

Open een browser en surf naar

<http://127.0.0.1:1880>.



???????

Merk op dat een node die data genereert (uitvoer) aan de rechterkant kan gekoppeld worden en een node die data binnen neemt (invoer) aan de linkerkant gekoppeld kan worden.

???????

???????

Data van TTN binnenhalen

Om data van The Things Network binnen te halen dient er een flow gebouwd te worden. De connectie met TTN wordt gemaakt aan de hand van MQTT.

MQTT

MQTT is één van de meest gebruikte protocollen voor Internet of Things apparaten te laten communiceren met elkaar.

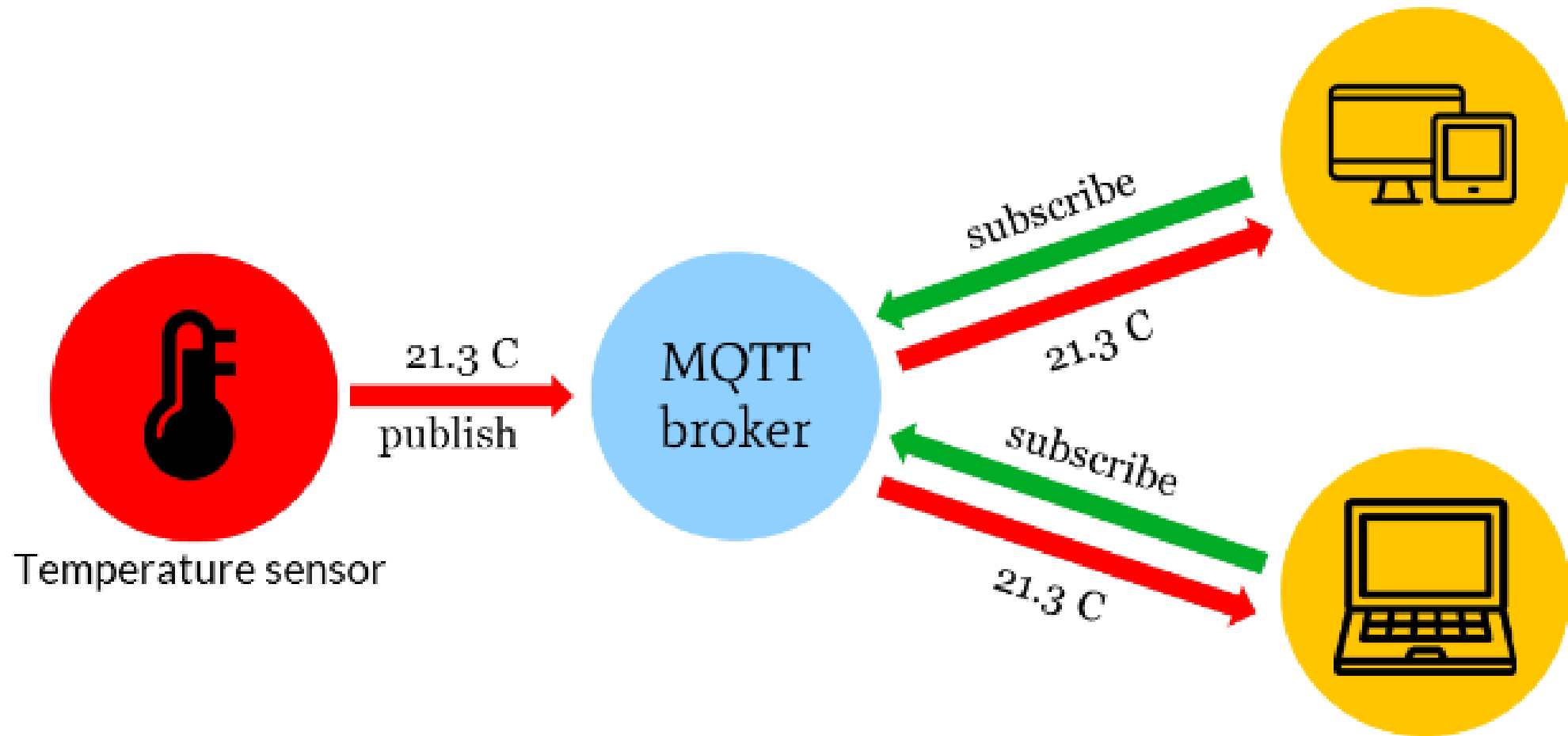
MQTT staat voor Message Queuing Telemetry Transport. Het is een zeer lichtgewicht berichten protocol dat gebruik maakt van publish/subscribe mechanisme om gegevens tussen verschillende clients uit te wisselen.

Het is klein in formaat, laag in vermogenverbruik en gebruikt geminimaliseerde data pakketten, ideaal voor 'machine tot machine' of Internet of Things.

Publiceren en abonneren

Stel dat je een temperatuur sensor hebt. Deze wil zijn waarden doorsturen naar een MQTT broker. Aan de andere kant hebben we toestellen zoals computers en smartphones die deze waarden willen ontvangen om ze weer te geven of te verwerken. Dan zijn er twee dingen die gebeuren.

- De sensor geeft een **topic** op waaronder het zijn gegevens zal publiceren. Bijvoorbeeld `temperatuur`. Dan zal het zijn temperatuurwaarde publiceren.
- Iedereen die de gegevens wil ontvangen kan zich dan gaan abonneren op `temperatuur`. Elke keer dat de sensor nieuwe gegevens publiceert worden alle abonnees automatisch verwittigd met de nieuwe temperatuur waarde.



Schematic data flow from sensor (machine) to devise (machine)

MQTT broker van TTN

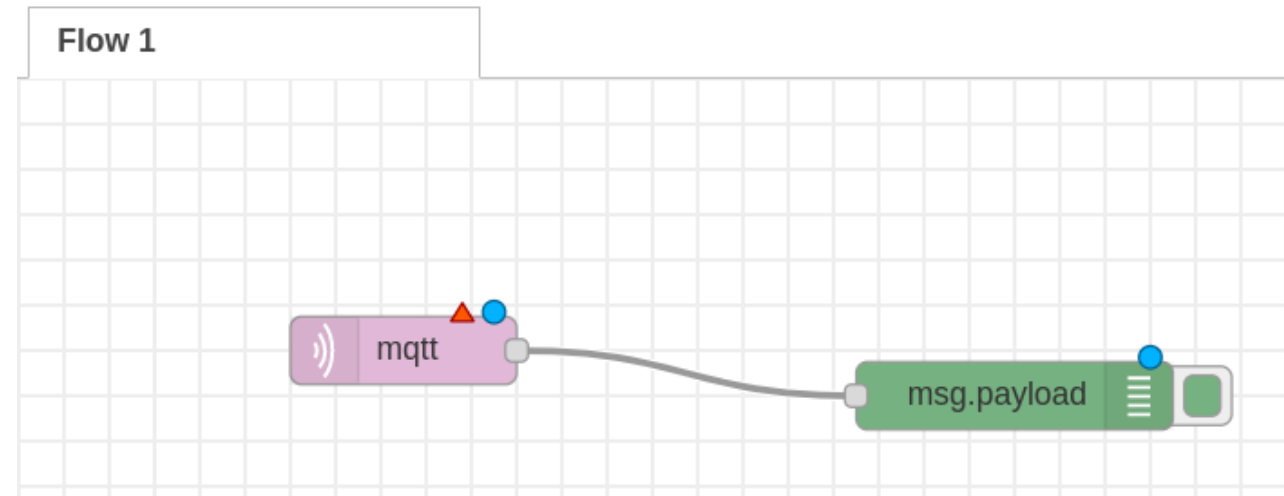
Bij onze opstelling is er echter wel een groot verschil!

De sensor published zijn waarden bij ons niet via MQTT maar wel via LoRaWAN naar de TTN. The Things Network stelt echter onze data ter beschikking via MQTT. Zij voorzien dus met andere woorden de broker waar wij kunnen subscriben op onze data.

MQTT Input

Zoek de node `mqtt input` in de lijst aan de linker kant, en sleep deze naar het centrale deel.

Vervolgens kunnen we ook een `debug` node aan de uitgang van de `mqtt input` node koppelen zodat we de informatie makkelijk kunnen raadplegen die van The Things Network komt.



MQTT Input - Configuratie

Dubbelklik op de mqtt input node om de configuratie te tonen. Hier dienen we de broker van The Things Network in te stellen.

Edit mqtt in node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🔗

🌐 Server

Add new mqtt-broker...

✎

📄 Topic

Topic

⚙ QoS

2

▼

➡ Output

auto-detect (string or buffer)

▼

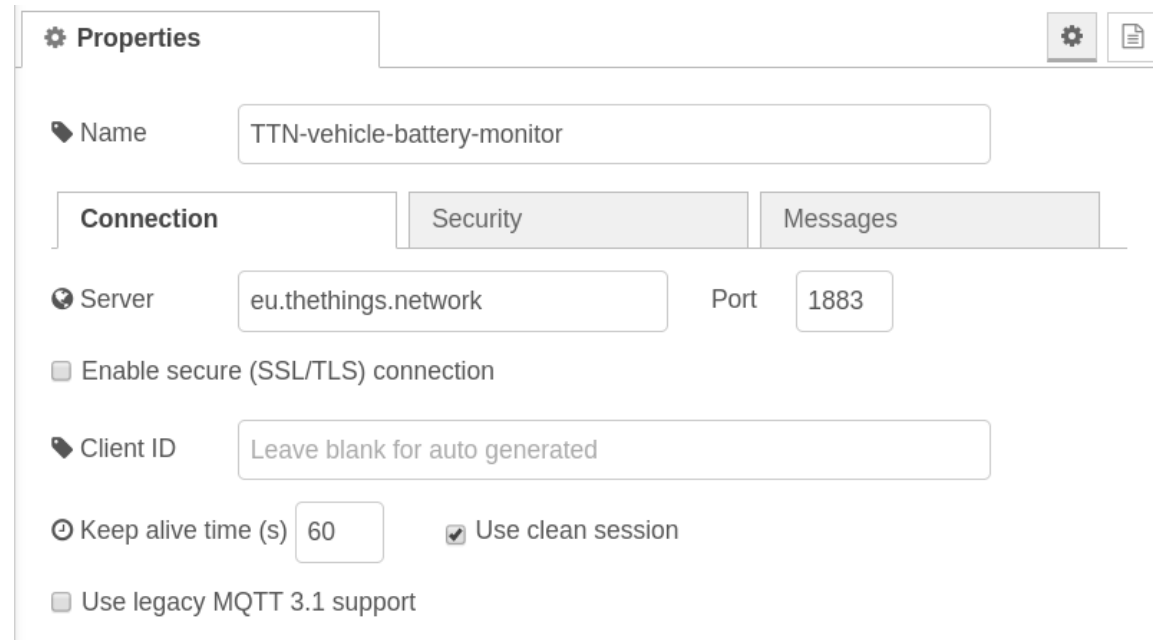
🏷 Name

Name

MQTT Input - Connectie

Klik op de knop met het potlood icoontje, rechts naast het invoerveld `Add new mqtt broker...`. Eerst en vooral kan je de broker configuratie best een duidelijk naam geven zoals `TTN-vehicle-battery-monitor`.

Vervolgens dienen we in het tabblad **Connection** de server in te stellen op `eu.thethings.network`.



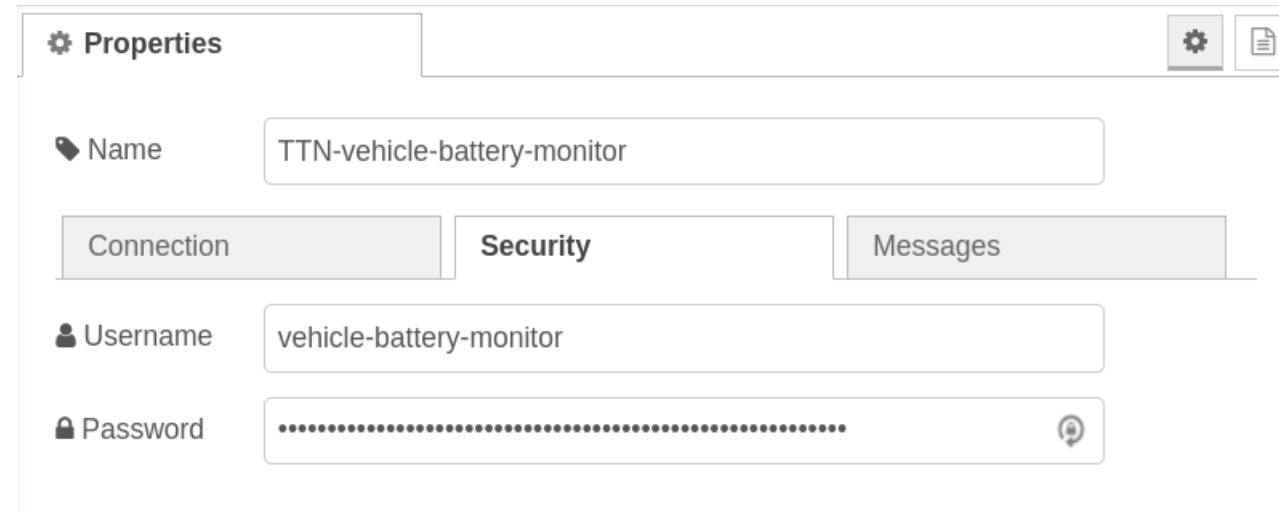
The screenshot shows the 'Properties' dialog box for an MQTT broker configuration. The 'Connection' tab is selected. The 'Name' field is set to 'TTN-vehicle-battery-monitor'. The 'Server' field is set to 'eu.thethings.network' and the 'Port' is set to '1883'. The 'Client ID' field is set to 'Leave blank for auto generated'. The 'Keep alive time (s)' is set to '60' and 'Use clean session' is checked. The 'Enable secure (SSL/TLS) connection' and 'Use legacy MQTT 3.1 support' options are unchecked.

Properties	
Name	TTN-vehicle-battery-monitor
Connection Security Messages	
Server	eu.thethings.network
Port	1883
<input type="checkbox"/> Enable secure (SSL/TLS) connection	
Client ID	Leave blank for auto generated
Keep alive time (s)	60
<input checked="" type="checkbox"/> Use clean session	
<input type="checkbox"/> Use legacy MQTT 3.1 support	

MQTT Input - Security

Vervolgens moeten we een gebruikersnaam en wachtwoord ingeven in het tabblad **Security**. Hiervoor moeten we het **App ID** (username) en de **Access Key** (password) van de The Things Network applicatie achterhalen. Deze gegevens kan je terugvinden op de console van The Things Network.

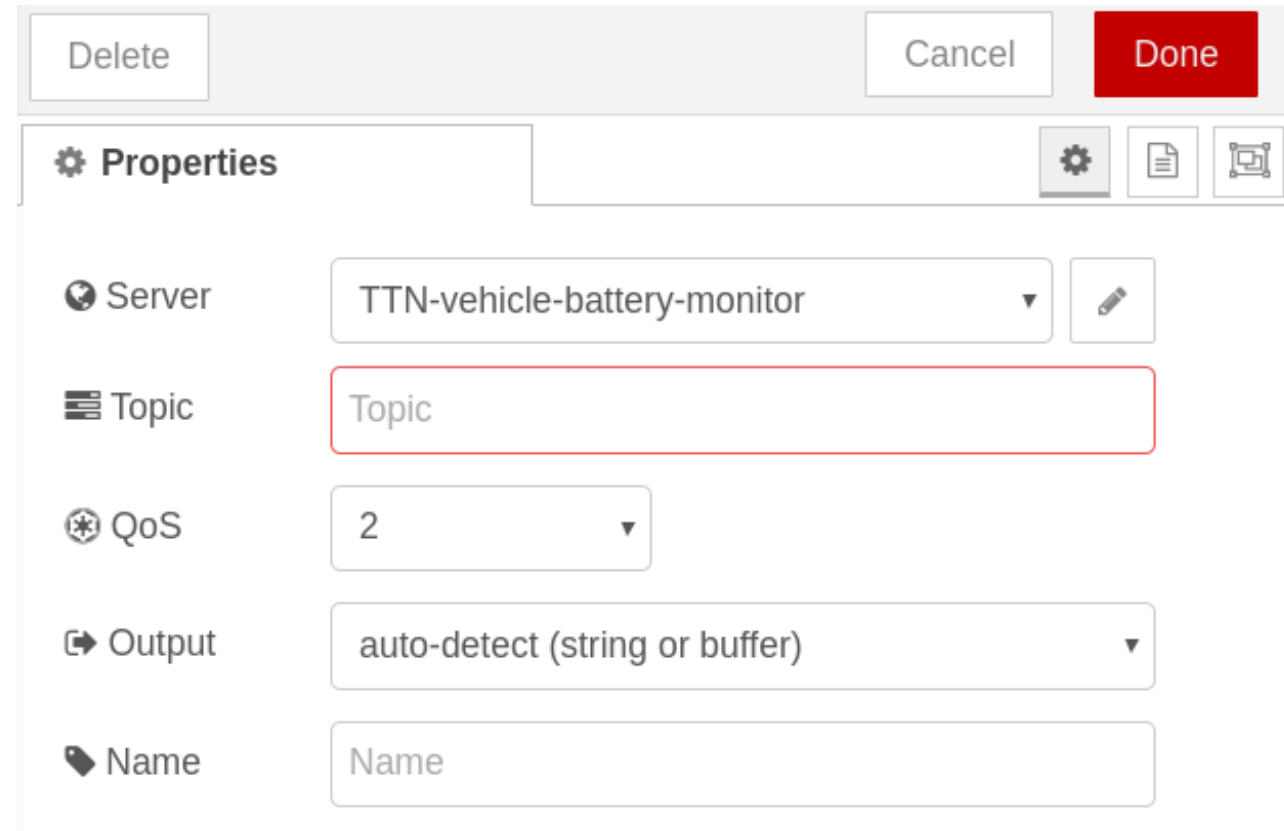
Klik vervolgens bovenaan rechts op de knop **Add**.



The screenshot shows a web interface for configuring an MQTT input. At the top, there is a 'Properties' tab with a gear icon and a document icon. Below the tab, there is a 'Name' field with the value 'TTN-vehicle-battery-monitor'. Underneath, there are three tabs: 'Connection', 'Security', and 'Messages'. The 'Security' tab is currently selected. In the 'Security' tab, there is a 'Username' field with the value 'vehicle-battery-monitor' and a 'Password' field with a masked password (dots) and a toggle icon for visibility.

MQTT Input - Het Topic instellen

Als laatste dienen we ook in te stellen waar onze data kan worden bekomen. Dit doen we door het topic in te stellen op basis van de naam die we aan ons device hebben gegeven. Dit kunnen we terug vinden op de [console](#) van The Things Network. Klik bovenaan rechts op Devices en kopieer de naam van je device.



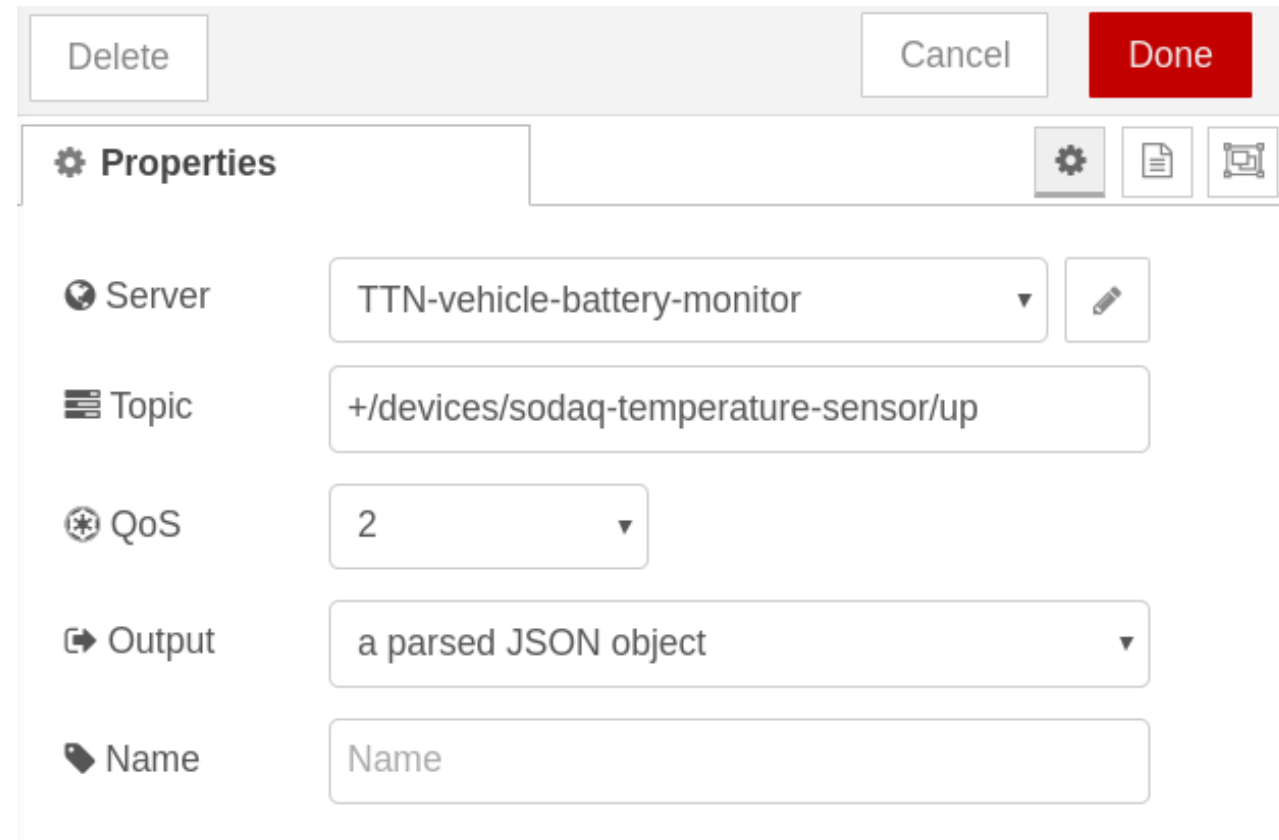
The screenshot shows the 'Properties' configuration window for an MQTT input. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. The 'Properties' section is divided into two tabs: 'Properties' (selected) and 'Advanced'. The 'Properties' tab contains the following settings:

- Server:** A dropdown menu showing 'TTN-vehicle-battery-monitor' with a pencil icon to its right.
- Topic:** A text input field containing the word 'Topic', which is highlighted with a red border.
- QoS:** A dropdown menu showing the value '2'.
- Output:** A dropdown menu showing 'auto-detect (string or buffer)'.
- Name:** A text input field containing the word 'Name'.

MQTT In - Het Topic Instellen

Het `device ID` dien je in het topic `+/devices/XXXX/up` in te vullen in plaats van `XXXX`.

Voor een device met id `sodaq-temperature-sensor` wordt dit dus:
`+/devices/sodaq-temperature-sensor/up`



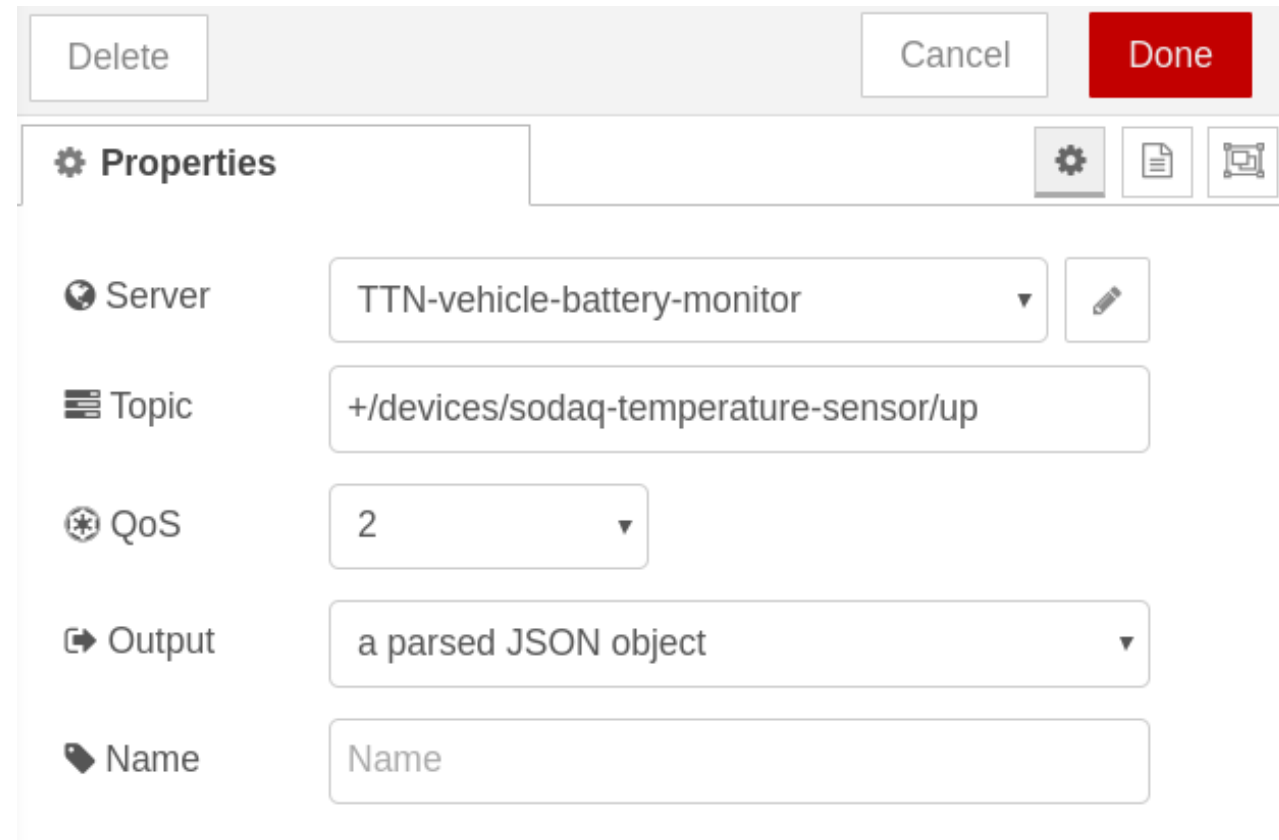
The screenshot shows the 'Properties' dialog box for an MQTT topic. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below the title bar, there are three icons: a gear (Settings), a document (List), and a square with a circle (Details). The 'Properties' section is active, showing the following fields:

- Server:** A dropdown menu showing 'TTN-vehicle-battery-monitor' with an edit icon to its right.
- Topic:** A text input field containing the topic `+/devices/sodaq-temperature-sensor/up`.
- QoS:** A dropdown menu showing the value '2'.
- Output:** A dropdown menu showing 'a parsed JSON object'.
- Name:** A text input field containing the placeholder text 'Name'.

MQTT In - Output als JS Object

Als laatste dienen we de mqtt node **Output** ook in te stellen zodat deze een JavaScript object terug geeft in plaats van een pure JSON string. Dit laat ons toe om later makkelijker te filteren. Selecteer als output **A parsed JSON object**.

Klik als laatste bovenaan rechts op **Done**.



The screenshot shows the configuration window for an MQTT node. At the top, there are three buttons: "Delete", "Cancel", and "Done". Below these is a "Properties" tab. The configuration is as follows:

Property	Value
Server	TTN-vehicle-battery-monitor
Topic	+/devices/sodaq-temperature-sensor/up
QoS	2
Output	a parsed JSON object
Name	Name

Delete

Cancel

Done

Properties



 Server

TTN-vehicle-battery-monitor ▼



 Topic

+/devices/sodaq-temperature-sensor/up

 QoS

2 ▼

 Output

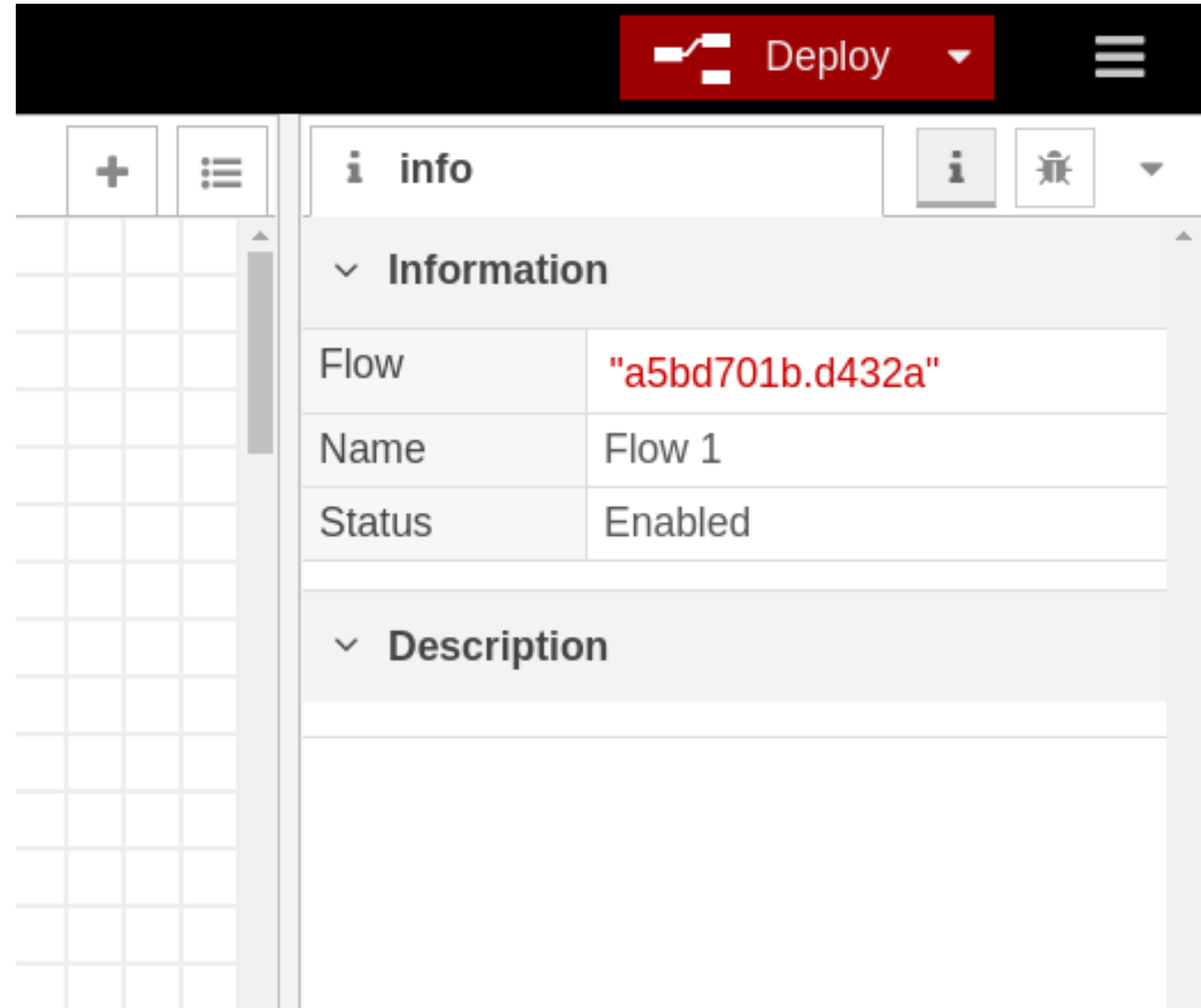
a parsed JSON object ▼

 Name

Name

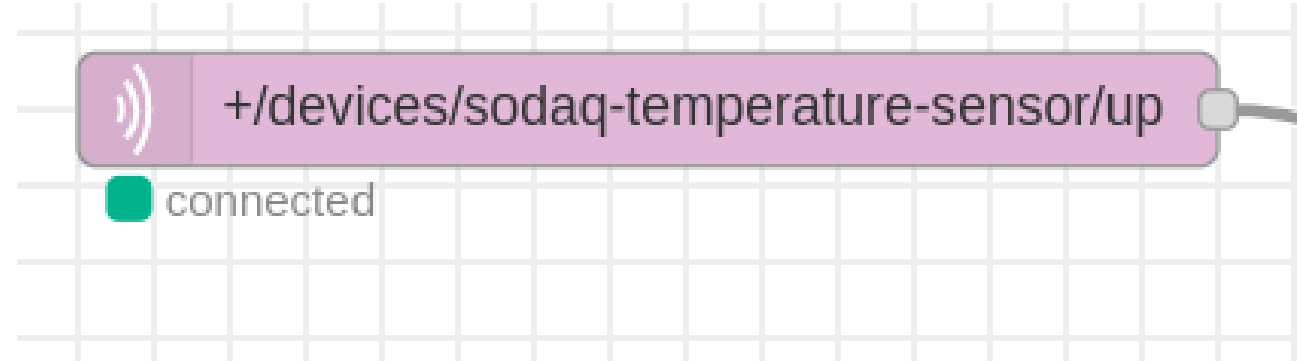
Deployen van de flow

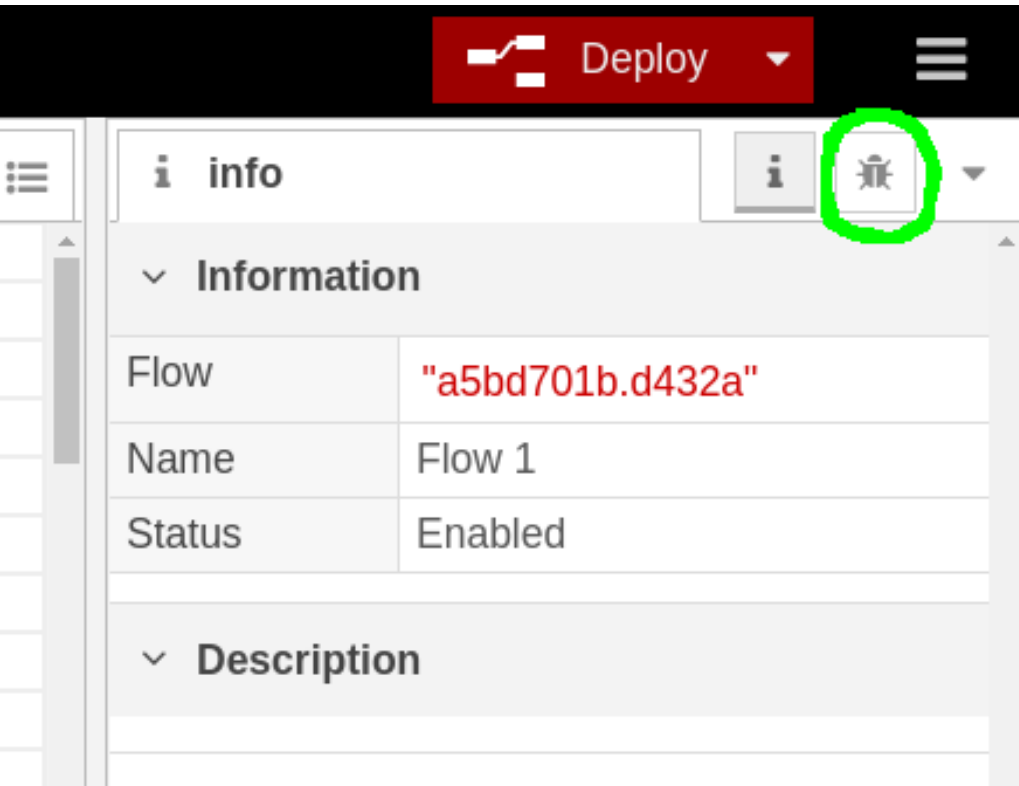
Klik bovenaan helemaal rechts op **Deploy** om de flow te activeren. Telkens je een aanpassing doet aan de flow dien je deze opnieuw te deployen.



MQTT - Connected

Indien alles goed is verlopen zou er nu onder `mqtt input node` `connected` moeten staan.





Show Debug Messages



Indien je nu op het kleine debug beestje klikt rechts bovenaan in Node RED dan zou je reeds data moeten zien binnenkomen van je sensor.

Data zonder sensoren

Als je geen sensor hebt kan je toch data *faken* door naar de console van The Things Network te surfen en naar jouw device te gaan.

Daar heb je dan de sectie `Simulate Uplink` . Hier kan je een raw byte payload ingeven. Bv.: `06 BD 22` . Dit komt overeen met een temperatuur van `17.25` en een batterij percentage van `34` .

Data zonder sensoren

Applications >  vehicle-battery-monitor > Devices >  sodaq-temperature-sensor

SIMULATE UPLINK

FPort	Payload
<input type="text" value="1"/>	<input type="text" value="06 BD 22"/> ✓ 3 bytes

Klik op **Send** om de data te simuleren. Je zou dit vervolgens moeten zien binnenkomen in Node-RED.

Payload Fields

Zoals te zien in de figuur zit onze data gedecodeerd in het veld `payload_fields`. Dit komt er zo in te staan doordat we een decoder op de TTN hebben ingesteld voor onze data.

debug

i

▼

all nodes

3/12/2020, 8:22:35 PM node: a1b20336.8da99

vehicle-battery-monitor/devices/sodaq-temperature-sensor/up : msg.payload : Object

▼ object

app_id: "vehicle-battery-monitor"

dev_id: "sodaq-temperature-sensor"

hardware_serial: "006C429BB8FB4955"

port: 1

counter: 24

payload_raw: "Br0i"

▼ payload_fields: object

battery: 34

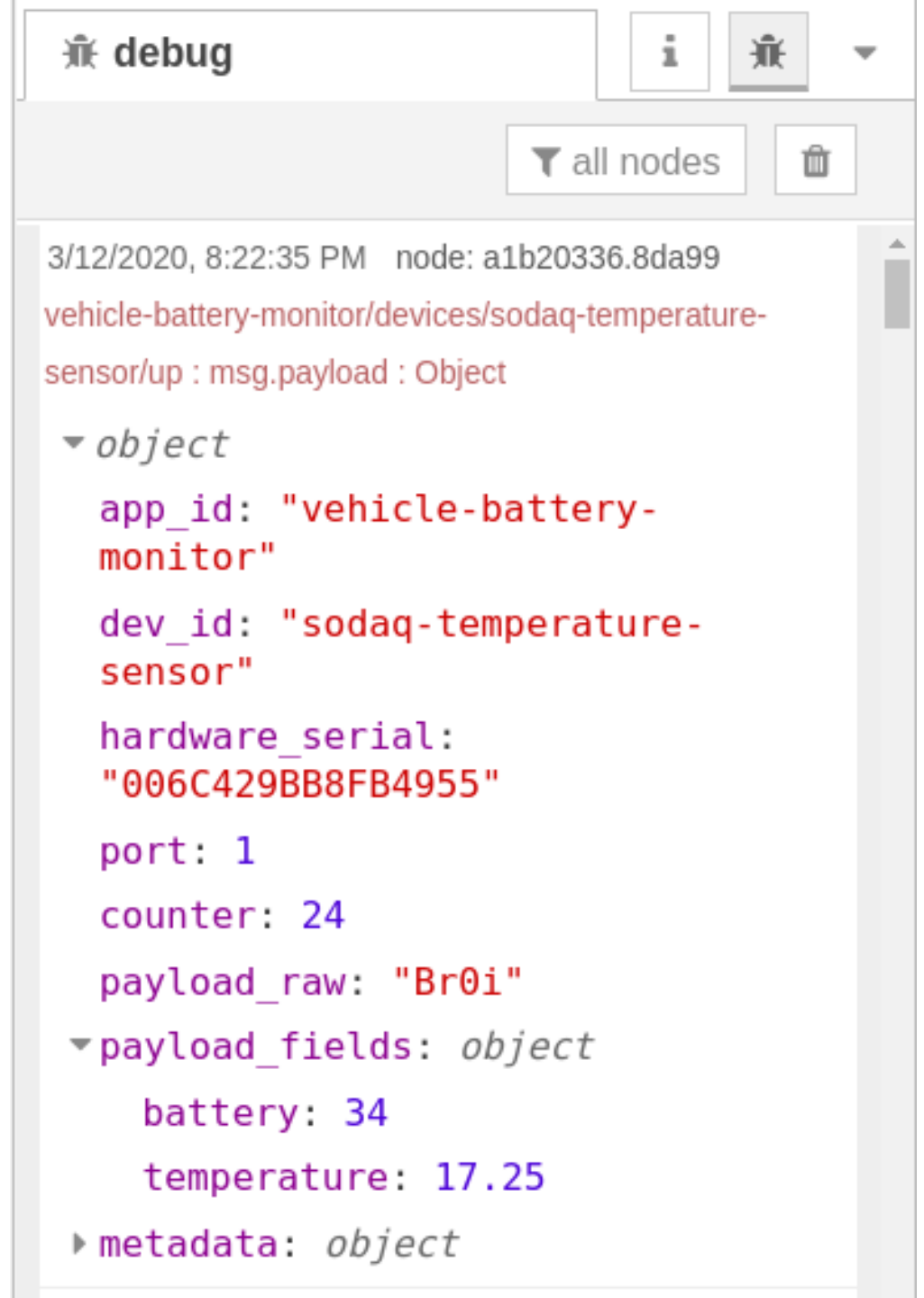
temperature: 17.25

► metadata: object

Filteren - Te veel data

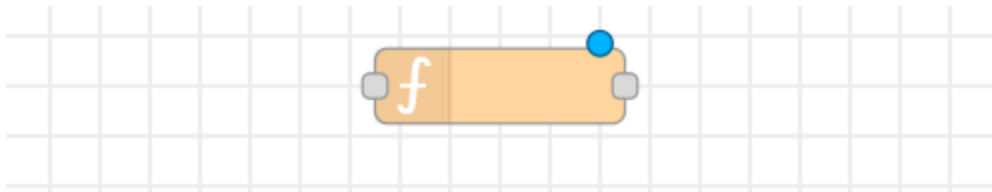
Op dit moment krijgen we niet enkel onze data binnen van de TTN maar ook allerlei metadata (`app_id` , `dev_id` , `port` , `counter`).

We dienen eerst onze data (die zich bevind onder `payload_fields`) er uit te filteren vooraleer deze verder kan verwerkt worden.



Filteren - Te veel data

Hiervoor kan je gebruik maken van de algemene `function node`.



Als je hier op dubbelklikt kan je een stukje JavaScript code bouwen om uit te voeren.

Filteren - Temperatuur

Om het payload field met de temperatuur eruit te filteren kan volgende code wordt gebruikt.

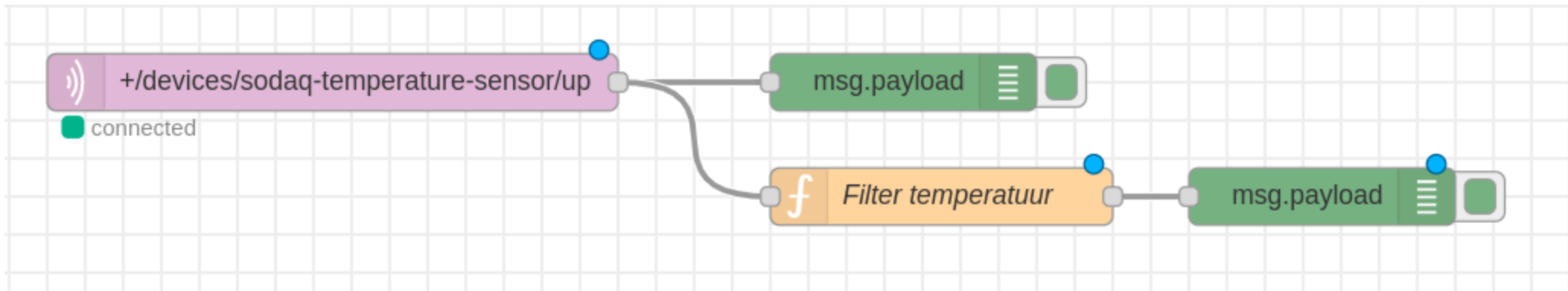
```
msg.payload = msg.payload.payload_fields.temperature;  
return msg;
```

Geef de node best een goede Name zoals Filter temperatuur .



Filteren - Temperatuur

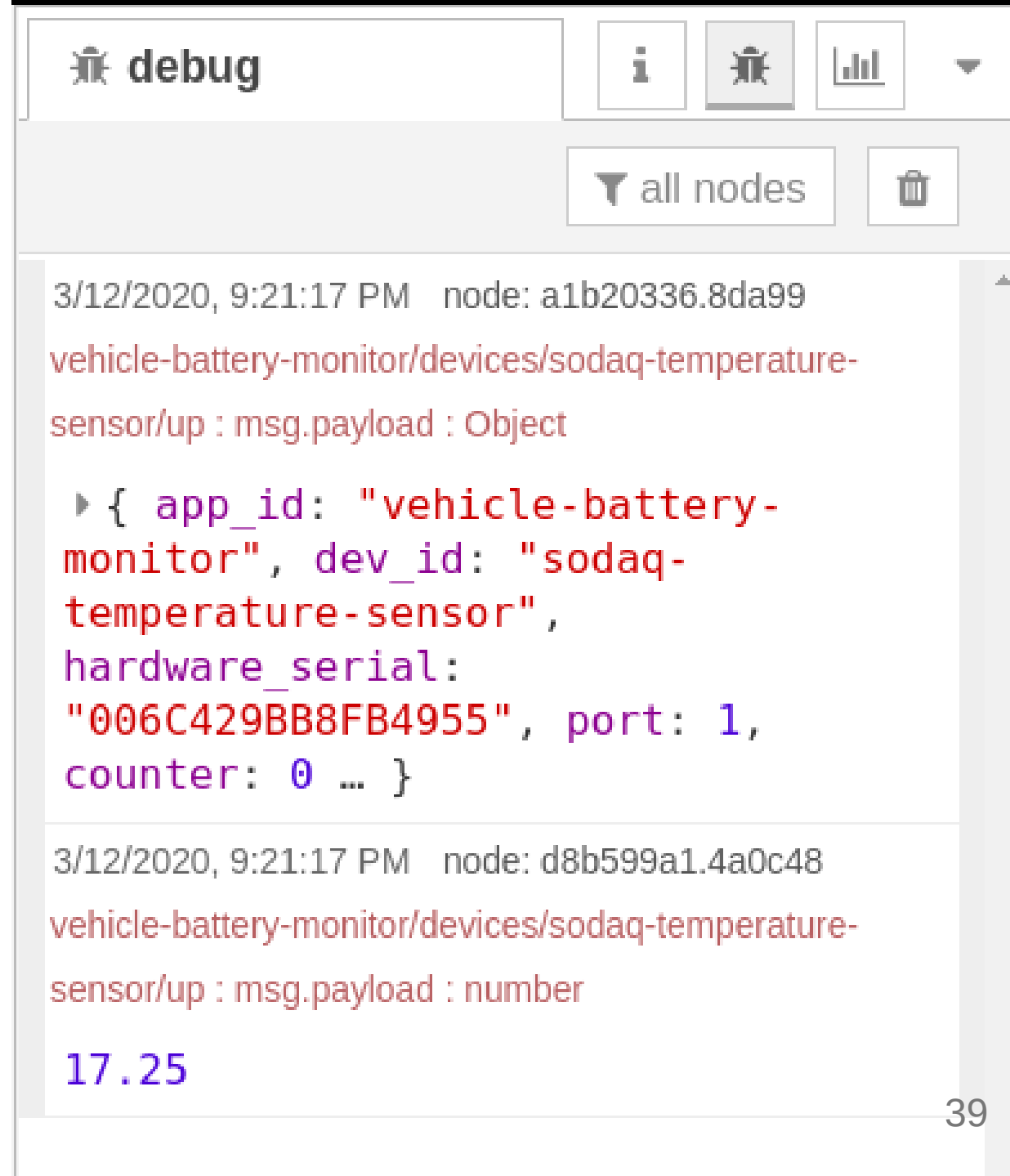
Als je op `Done` klikt kan je nu de `function` node verbinden met de `mqtt input` node. Het is ook een goed idee om aan de uitgang van de `function` node een `debug` node te hangen zoals in de figuur. Dit laat toe de output te bekijken.



Filteren - Temperatuur

Als je de flow deployed en opnieuw via de console van de TTN data simuleert zou het effect moeten te zien zijn in Node-RED debug.

Merk op dat we de temperatuur nu apart krijgen.



The image shows the Node-RED Debug Console interface. At the top, there's a 'debug' tab with icons for information, error, and logs. Below the tab, there's a filter dropdown set to 'all nodes' and a trash icon. The main area displays two log entries. The first entry, from node 'a1b20336.8da99', shows a message payload as an object with fields: app_id, dev_id, hardware_serial, port, and counter. The second entry, from node 'd8b599a1.4a0c48', shows the same message payload but the temperature value is displayed as a separate line, '17.25'.

```
3/12/2020, 9:21:17 PM  node: a1b20336.8da99
vehicle-battery-monitor/devices/sodaq-temperature-
sensor/up : msg.payload : Object

  ▶ {  app_id: "vehicle-battery-
      monitor", dev_id: "sodaq-
      temperature-sensor",
      hardware_serial:
      "006C429BB8FB4955", port: 1,
      counter: 0 ... }

3/12/2020, 9:21:17 PM  node: d8b599a1.4a0c48
vehicle-battery-monitor/devices/sodaq-temperature-
sensor/up : msg.payload : number

17.25
```

Oefening - Batterij percentage

Probeer nu hetzelfde voor de batterij. Je zou volgende resultaat moeten bekomen.

