Lecture 8. NodeRED and Databases

2143399 ICE CAPSTONE

03.25.2019

Class Schedule

March 18	Lecture 7 – Open IoT platform
March 25	Lecture 8 – Open IoT platform and database
April 1	Advanced data analytics and machine learning (guess lecture)
April 10 (Wednesday)	**MS3 presentation**
April 15	No class (Songkran day)
April 22 *	Lecture 9: Data analytics
May 1 (Wednesday)	**MS4 presentation**
May 13	**Open book & open note final exam (2 hours)** **+ discussion on conference papers/submission preparation**

ISE do Support Travel Expenses to Present a Conference Paper

Each student can receive this support at most two times

Conditions

- Paper is included in IEEEXplore, ISI or SCOPUS databases
- One support (15,000Baht) per student per paper
- Be full-time ISE student when the paper is submitted for review
- Give presentation of technical paper at the conference
- Submit application form at least 45 days before the conference begins

IEEE Smart Cities

5th IEEE International Smart Cities Conference

October 14 - 17, 2019 Casablanca, Morocco

Paper submission deadline:	June 1st, 2019
Notification of acceptance:	July 15th, 2019
Camera ready version:	July 31th, 2019

Home Keynote/Plenary Sessions News Committee Program

Home / Call for Conference Papers

Call for Conference Papers

Download Call For Papers *PDF*

The Fifth IEEE Annual International Smart Cities Conference (ISC2 2019) will be held in Casablanca, Morocco during October 14 – 17, 2019. As the flagship conference sponsored by the IEEE Smart Cities Technical Community that is a coalition of six IEEE technical societies and organizations, ISC2-2019 will bring together researchers and practitioners in a collaborative smart cities discussion, including scholars, citizens, policymakers, administrators, infrastructure operators, industry representatives, economists, sociologists, and academicians. Technical exchanges within the research community will encompass panels, plenary talks, technical sessions, tutorials, workshops and exhibitions. The conference theme is: Frugality and Inclusion paving the way towards Future Smart Sustainable Cities & Communities.

.

Other Upcoming IEEEE PES Conferences

2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)

29 Sept - 2 Oct 2019 | Bucharest, Romania

Full paper submission Due: 7 April 2019

11th IEEE PES Asia-Pacific Power and Energy Engineering Conference 2019 (APPEEC)

1-4 December 2019 | Macao, China

Submission Deadline: 30 June 2019

From the Class Last Week

Workshop 1: Install NodeRED

Workshop 2: Build your first flow

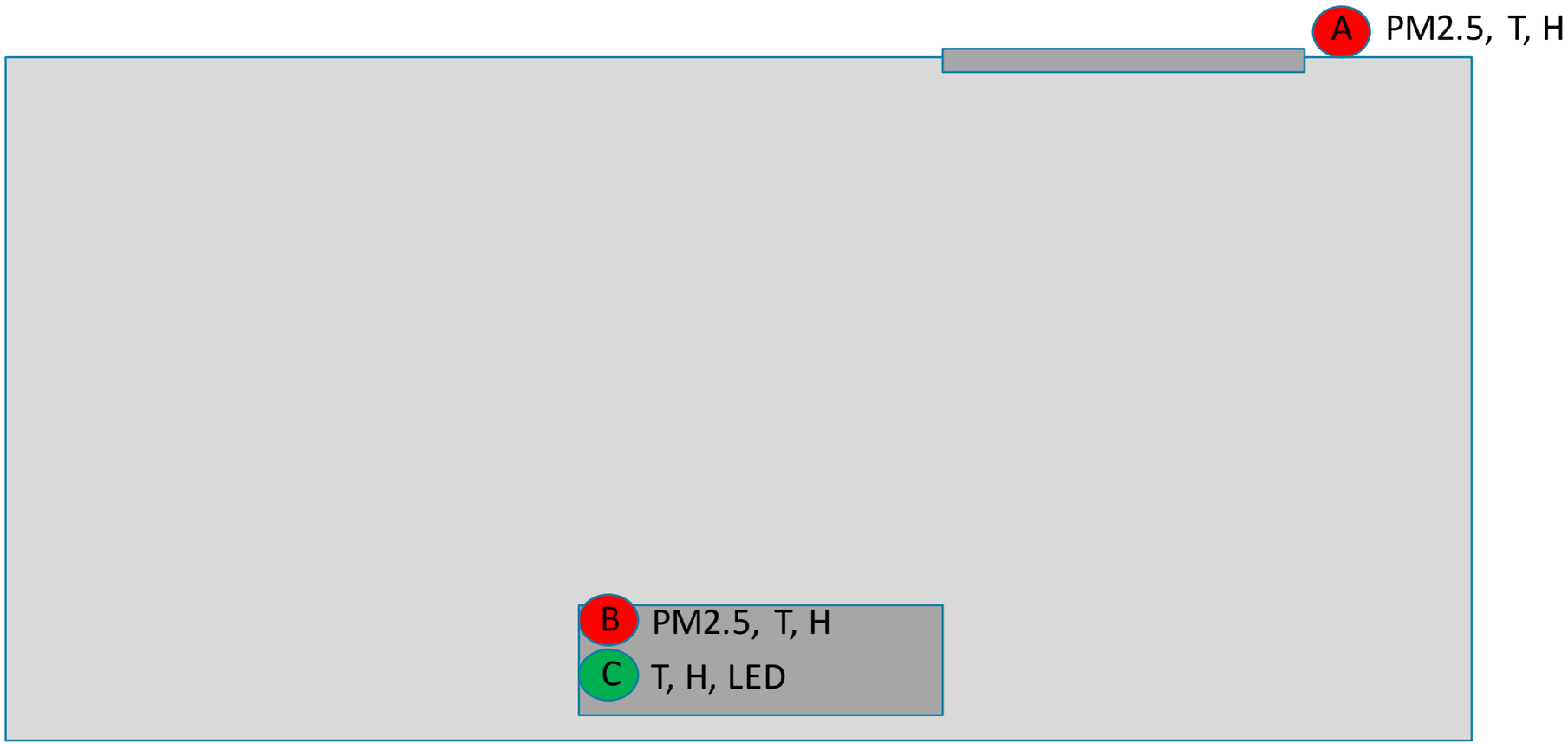
Workshop 3: Get weather data and send alert

Workshop 4: UI nodes from node-red-dashboard

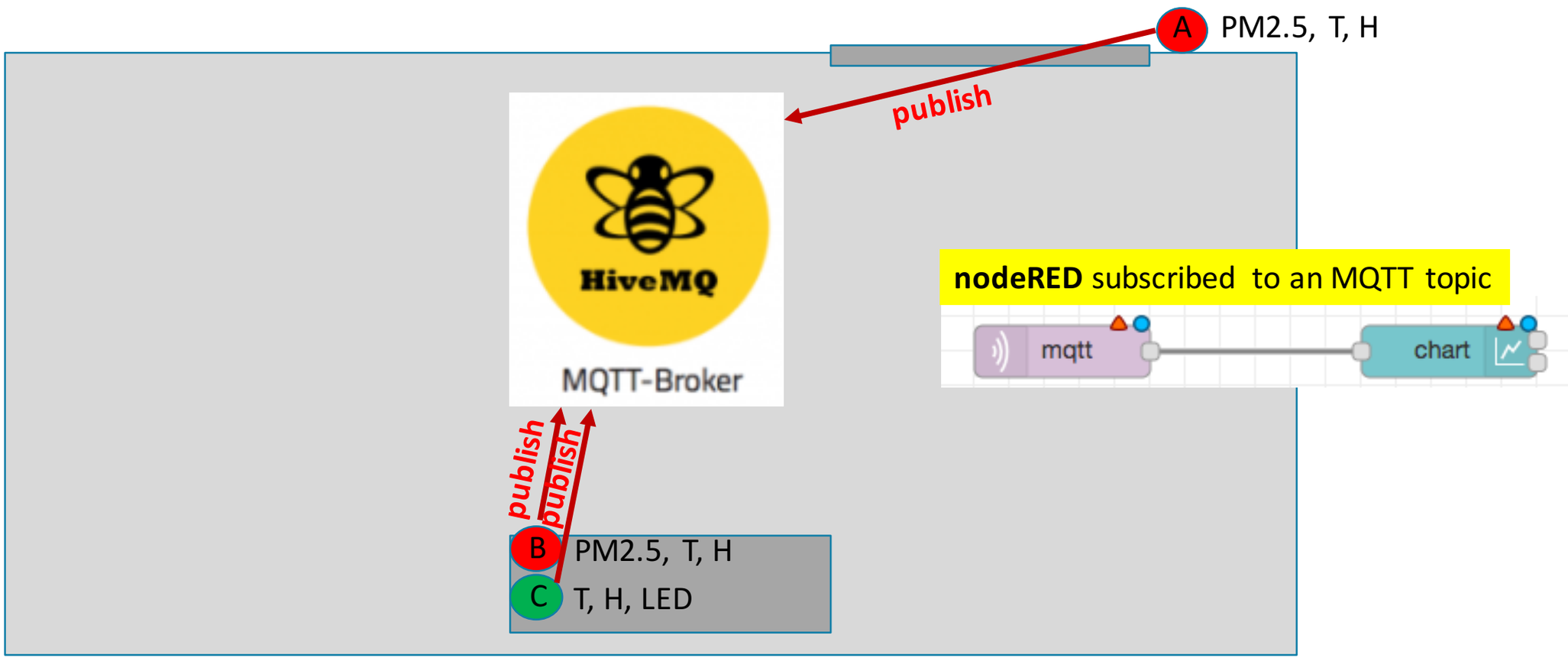
Workshop 5: MQTT

Workshop 6: Arduino, MQTT and NodeRED***

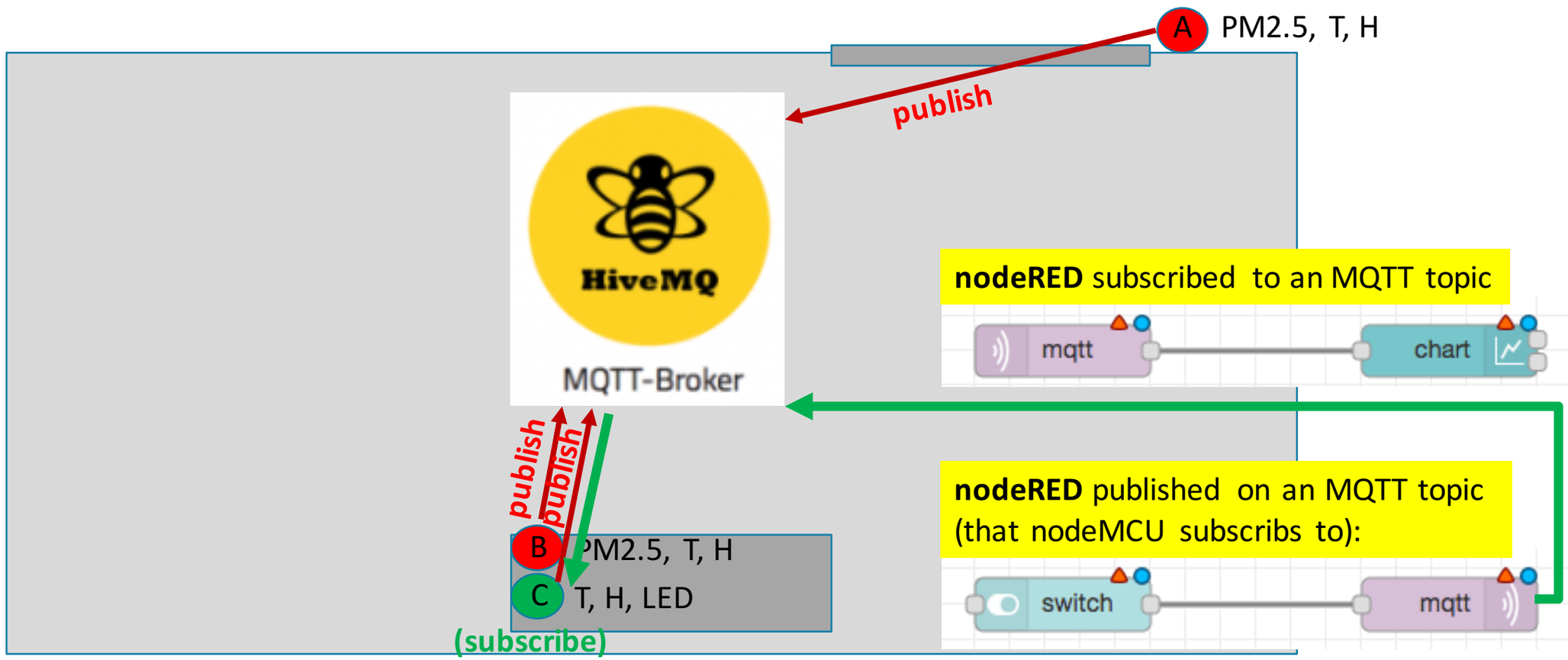
Sensor Locations



MQTT Concept: Publish



MQTT Concept: Subscribe



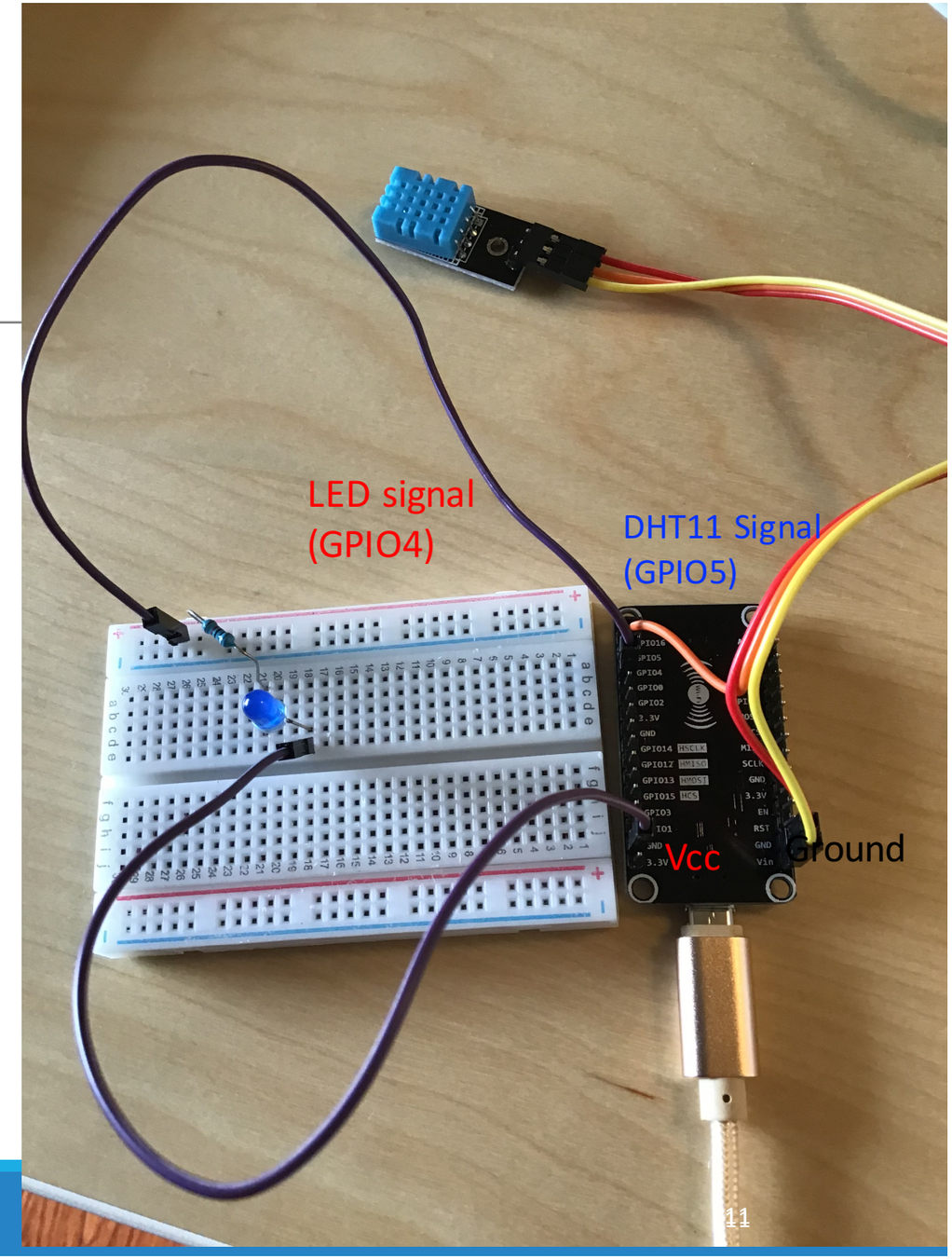
Workshop 6. Arduino, MQTT and NodeRED

We will learn how to publish data from Arduino (NodeMCU-DHT11) and subscribe to the data we publish, using MQTT and NodeRED. We will also use MQTT to control an LED.

Each Group to Connect LED and DHT11 to NodeMCU

DHT11 signal (GPIO5)

LED signal (GPIO4)



Edit & Upload the Code

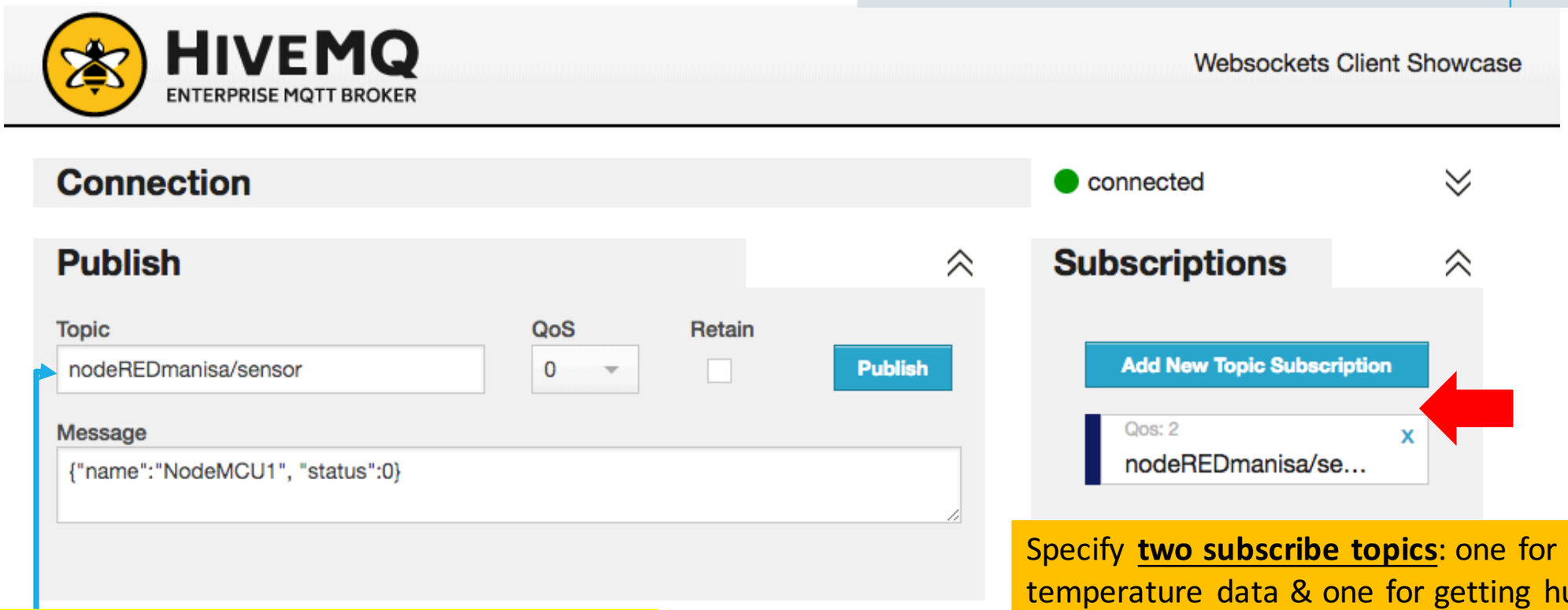
Github code: https://github.com/ICECapstone/MQTT_DHT11_LED

- Replace **ssid** with your WiFi ssid
- Replace **password** with your WiFi password
- **mqtt_server**, please use "broker.mqttdashboard.com”
- NodeMCU to **publish** temp/humidity to MQTT broker
  - pubTopicT = “GROUPNAME/temperature”
  - pubTopicH = “GROUPNAME/humidity”
- NodeMCU to **subscribe** to a topic from MQTT broker
  - subTopicLED = “GROUPNAME/LED”

GroupName:
- vCANTEEN
- LOCKER
- PARKING
- CUMAP
- HYPERBORA
- MONKOL

Set up Pub/Sub Topics

NodeMCU Pub Topics:		Sub Topics:
• vCANTEEN/temperature	.../humidity	.../LED
• LOCKER/temperature	.../humidity	.../LED
• PARKING/temperature	.../humidity	.../LED
• CUMAP/temperature	.../humidity	.../LED
• HYPERBORA/temperature	.../humidity	.../LED
• MONKOL/temperature	.../humidity	.../LED



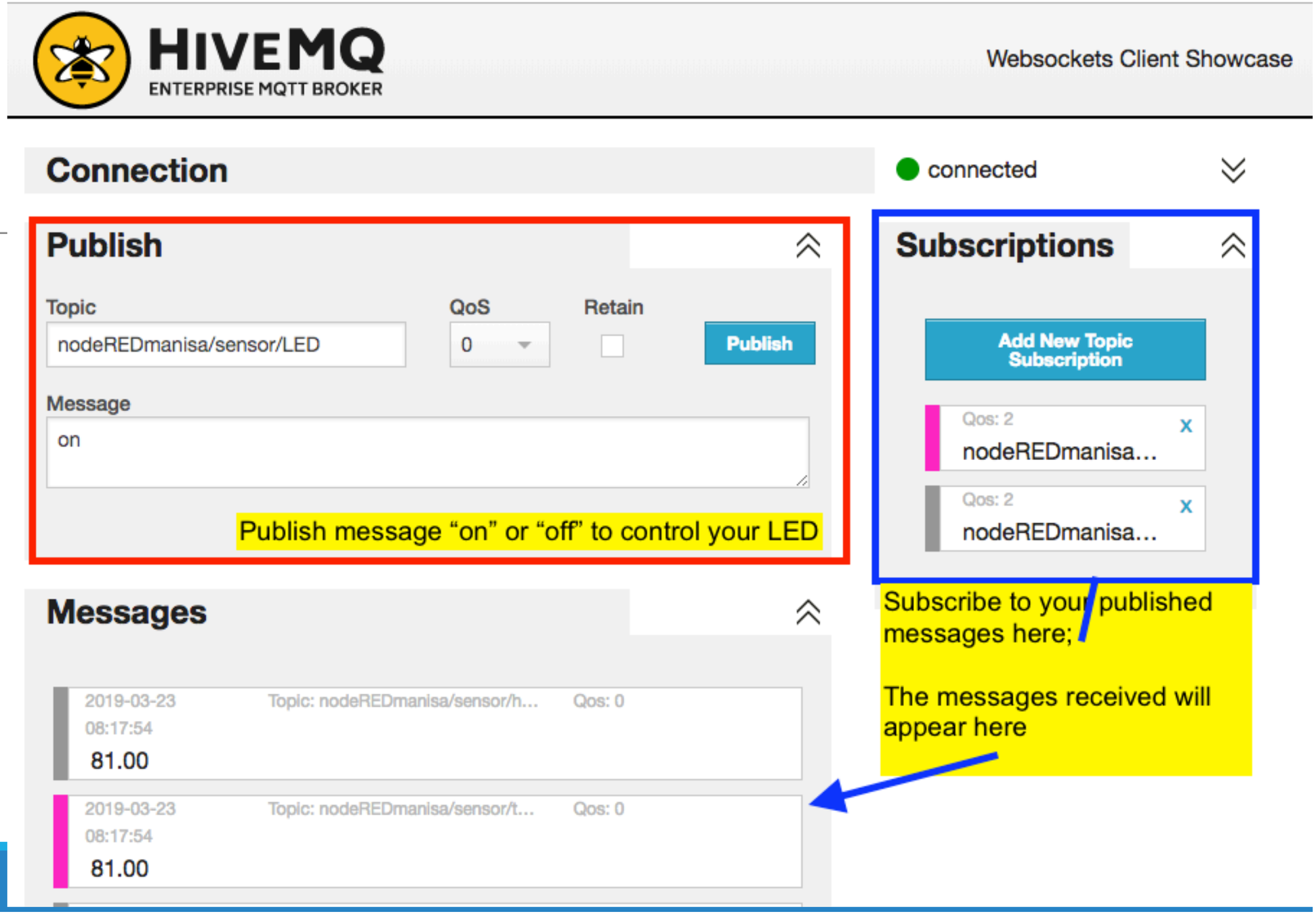
Specify **one publish topic** for controlling an LED:

- GROUPNAME/LED

Specify **two subscribe topics**: one for getting temperature data & one for getting humidity data.

- GROUPNAME/temperature
- GROUPNAME/humidity

Check MQTT Messages



Let us Get the Data to NodeRED Dashboard

Start NodeRED

- Use command: node-red
- Visit your server IP:

http://127.0.0.1:1880

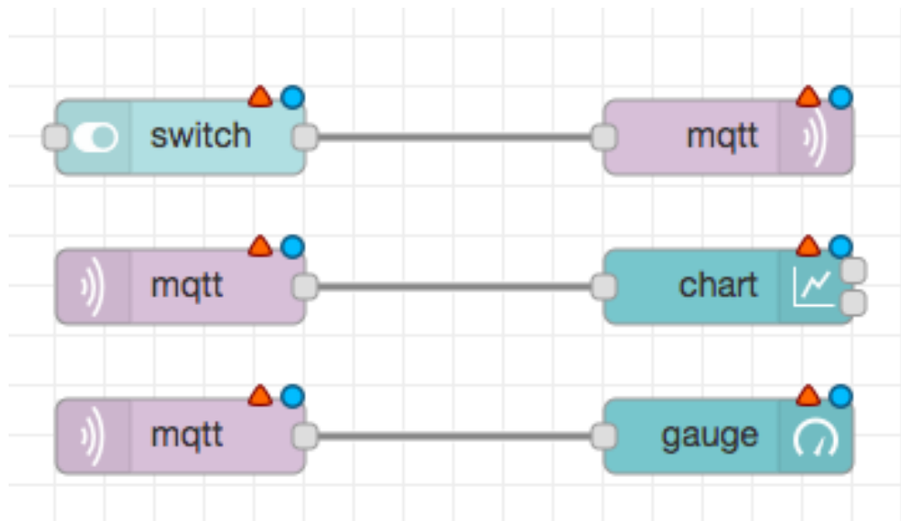
ManisaMacBook — node-red NVM_RC_VERSION= — 80×45

```
Last login: Sat Feb 23 18:02:25 on console
Manisas-MacBook-Air-388:~ ManisaMacBook$ node -v
v8.10.0
Manisas-MacBook-Air-388:~ ManisaMacBook$ node-red
1 Mar 06:02:11 - [info]

Welcome to Node-RED
===================

1 Mar 06:02:11 - [info] Node-RED version: v0.19.5
1 Mar 06:02:11 - [info] Node.js  version: v8.10.0
1 Mar 06:02:11 - [info] Darwin 16.7.0 x64 LE
1 Mar 06:02:11 - [info] Loading palette nodes
1 Mar 06:02:12 - [warn] rpi-gpio : Raspberry Pi specific node set inactive
1 Mar 06:02:13 - [info] Dashboard version 2.12.2 started at /ui
1 Mar 06:02:14 - [info] Settings file  : /Users/ManisaMacBook/.node-red/settings
.js
1 Mar 06:02:14 - [info] Context store  : 'default' [module=memory]
1 Mar 06:02:14 - [info] User directory : /Users/ManisaMacBook/.node-red
1 Mar 06:02:14 - [warn] Projects disabled : editorTheme.projects.enabled=false
1 Mar 06:02:14 - [info] Flows file     : /Users/ManisaMacBook/.node-red/flows_Ma
nisas-MacBook-Air-388.local.json
1 Mar 06:02:14 - [info] Creating new flow file
1 Mar 06:02:14 - [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.
---------------------------------------------------------------------

1 Mar 06:02:14 - [info] Starting flows
1 Mar 06:02:14 - [info] Started flows
1 Mar 06:02:14 - [info] Server now running at http://127.0.0.1:1880/
```

Create the Flow
(To listen to your group's sensor)

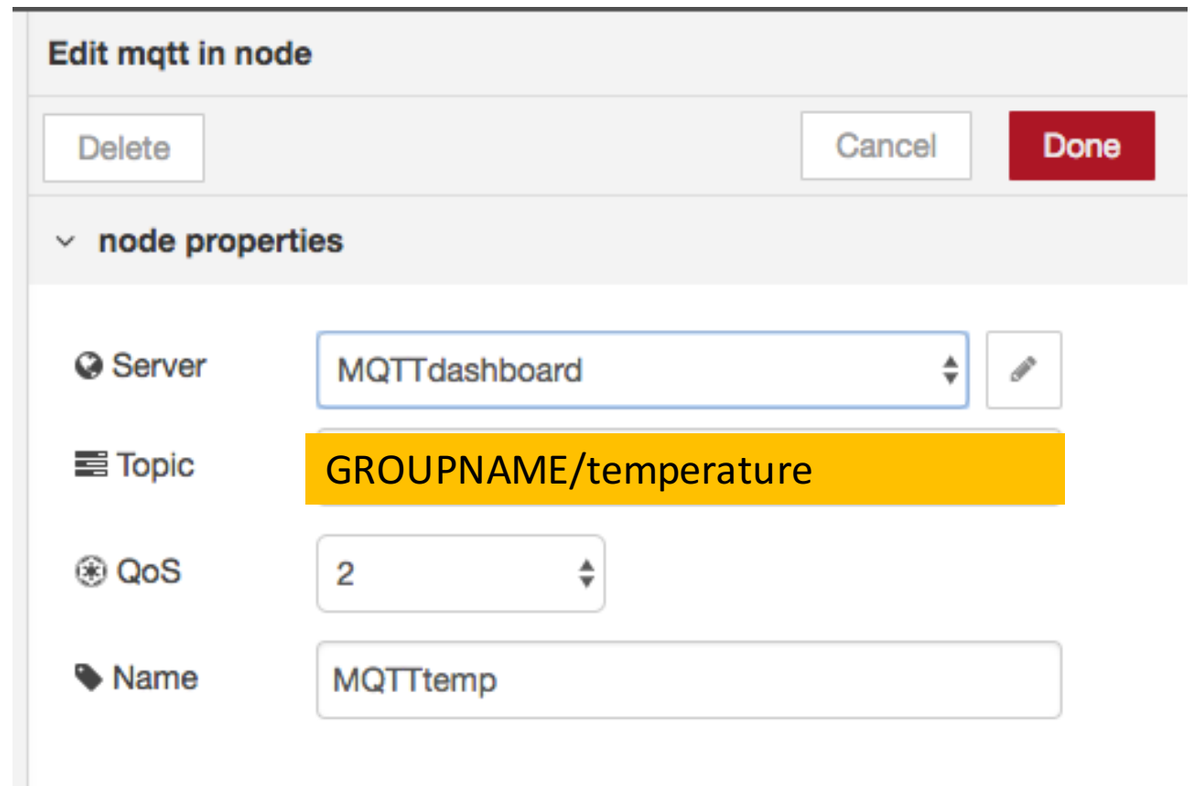
•Drag & drop: two **mqtt** nodes (input), **switch, chart, gauge** nodes (dashboard), one **mqtt** node (output).

• Wire them together as shown.



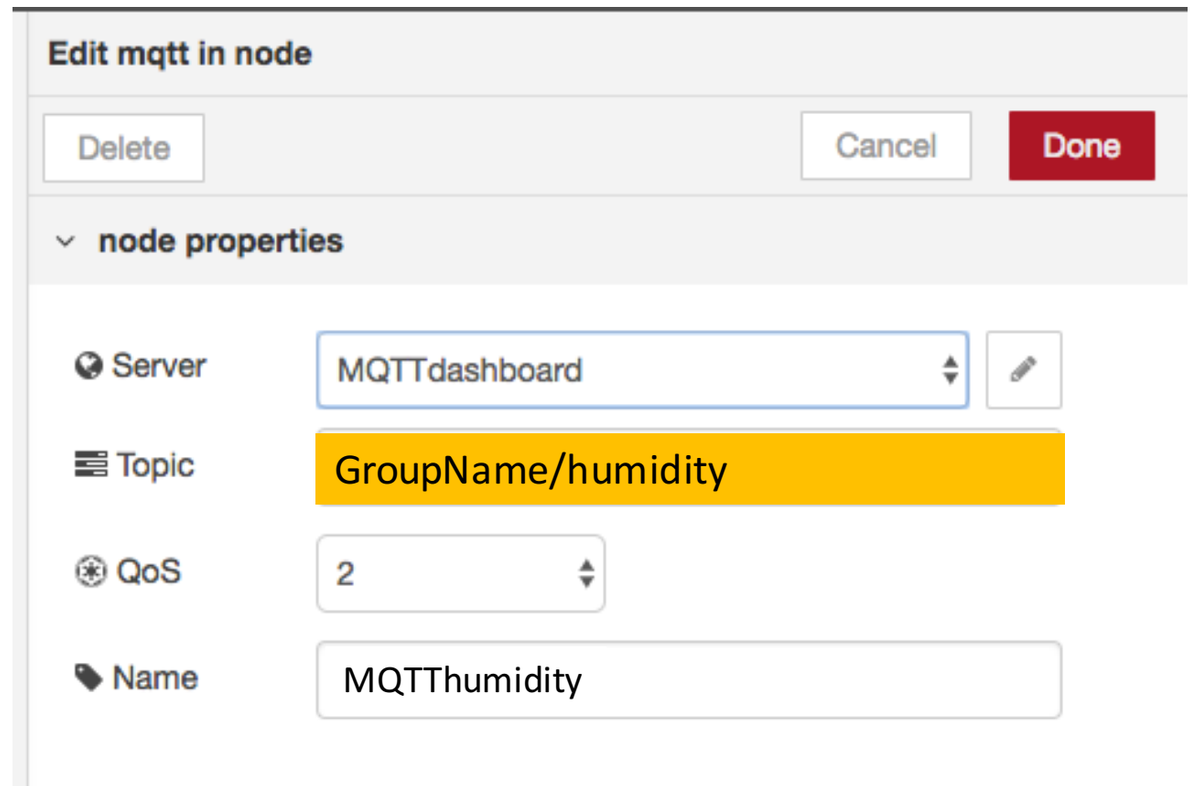
Configure the 1st mqtt Input Node

- Configure the **mqtt** node to subscribe to temperature data
- Server:
  - user the same server as the previous exercise
- Topic:
  - **unique topic name (to subscribe to the published temperature)**
- Name:
  - MQTTtemp (optional)
- Click "Done"



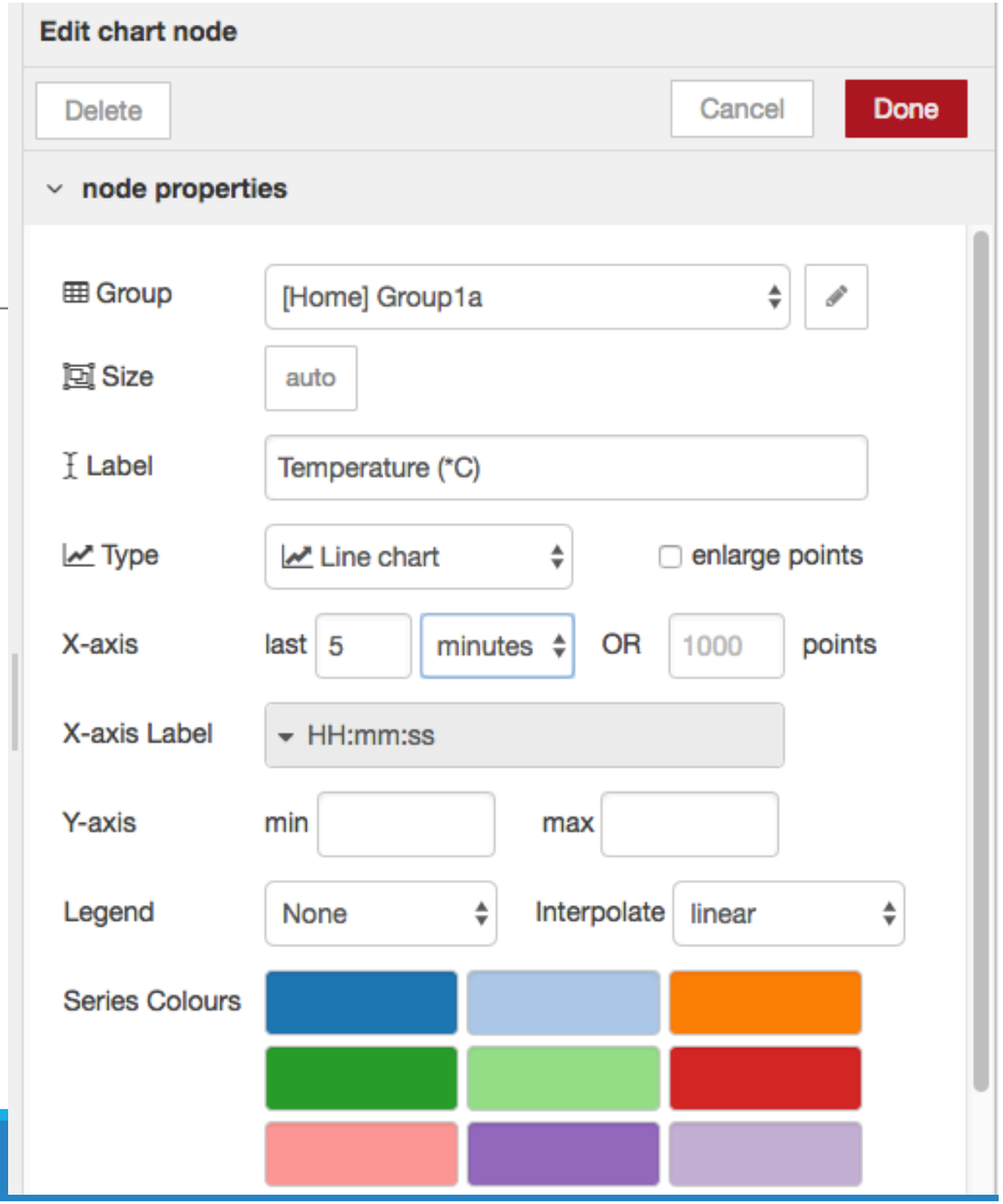
Configure the 2nd mqtt Input Node

- Configure the **mqtt** node to subscribe to humidity data
- Server,
  - use same server
- Topic:
  - **any unique topic name (to subscribe to humidity topic)**
- Name:
  - MQTThumidity (optional)
- Click "Done"



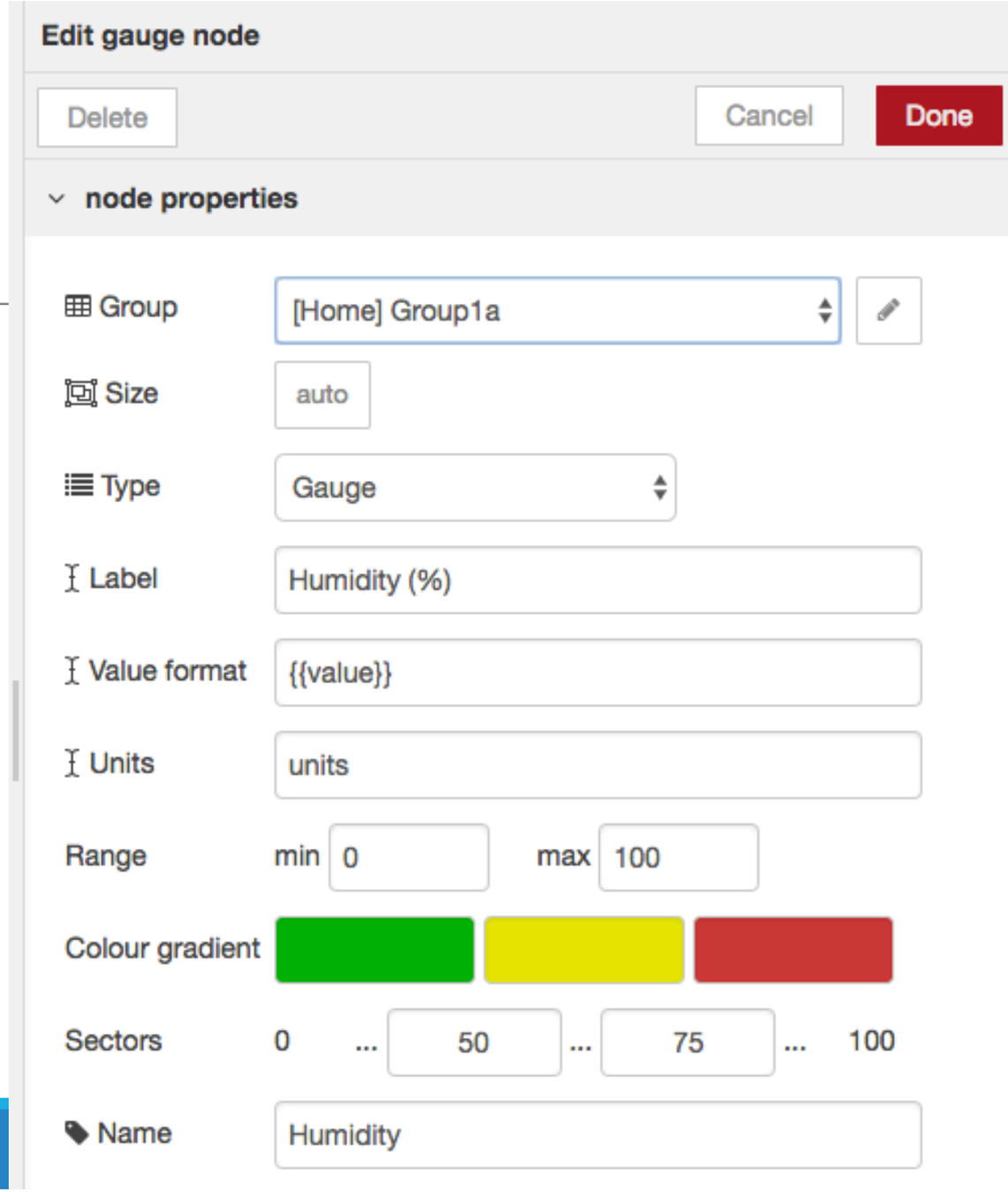
Edit the Chart Node

- Group:
  - Please feel free to use the existing or create a new UI group
- X-axis: last 5 minutes
- Name: Temperature
- Hit “Done”



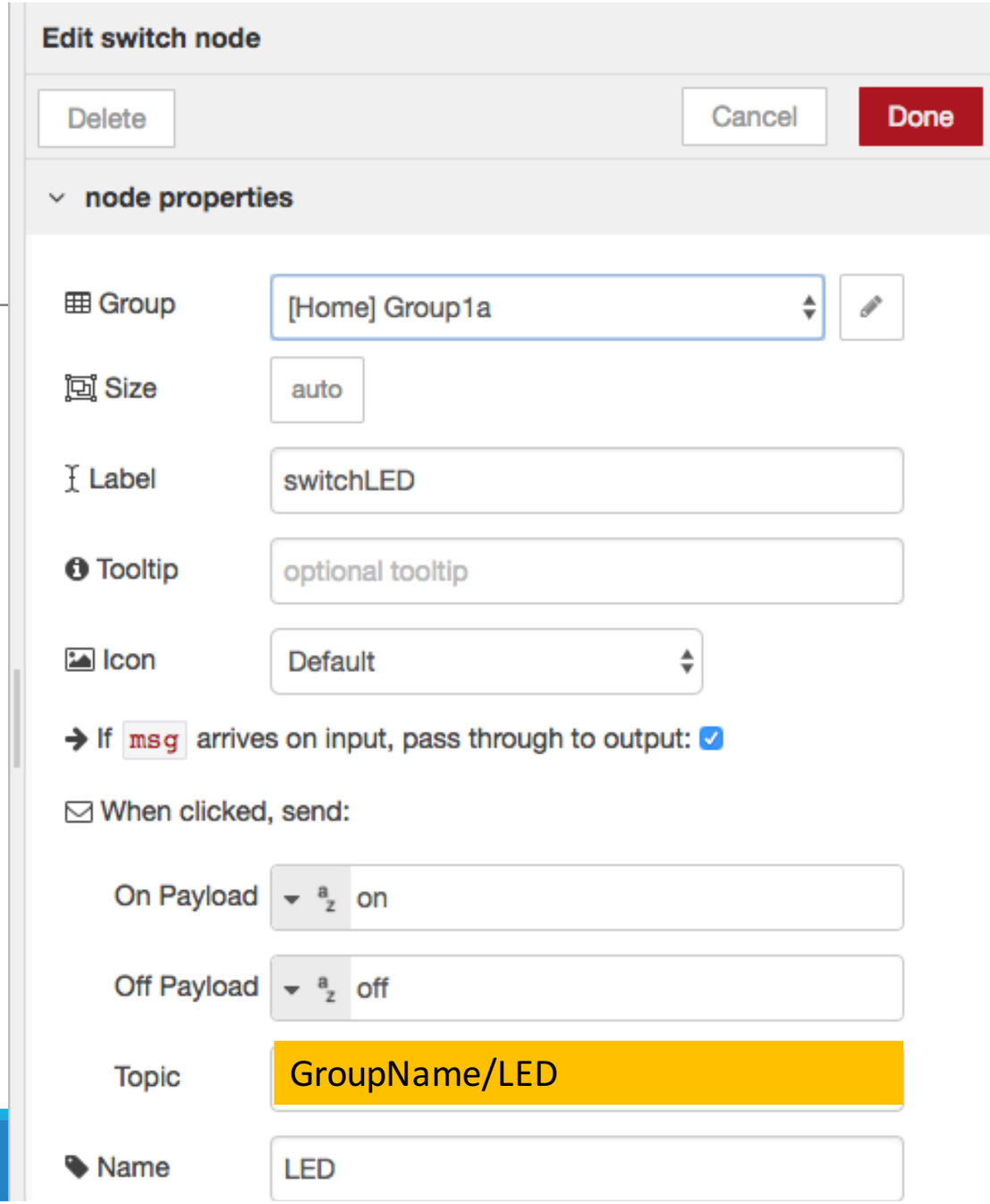
Edit the Gauge Node

- Group:
  - Please feel free to use the existing or create a new UI group
- Range: 0 - 100
- Name: Humidity
- Hit “Done”



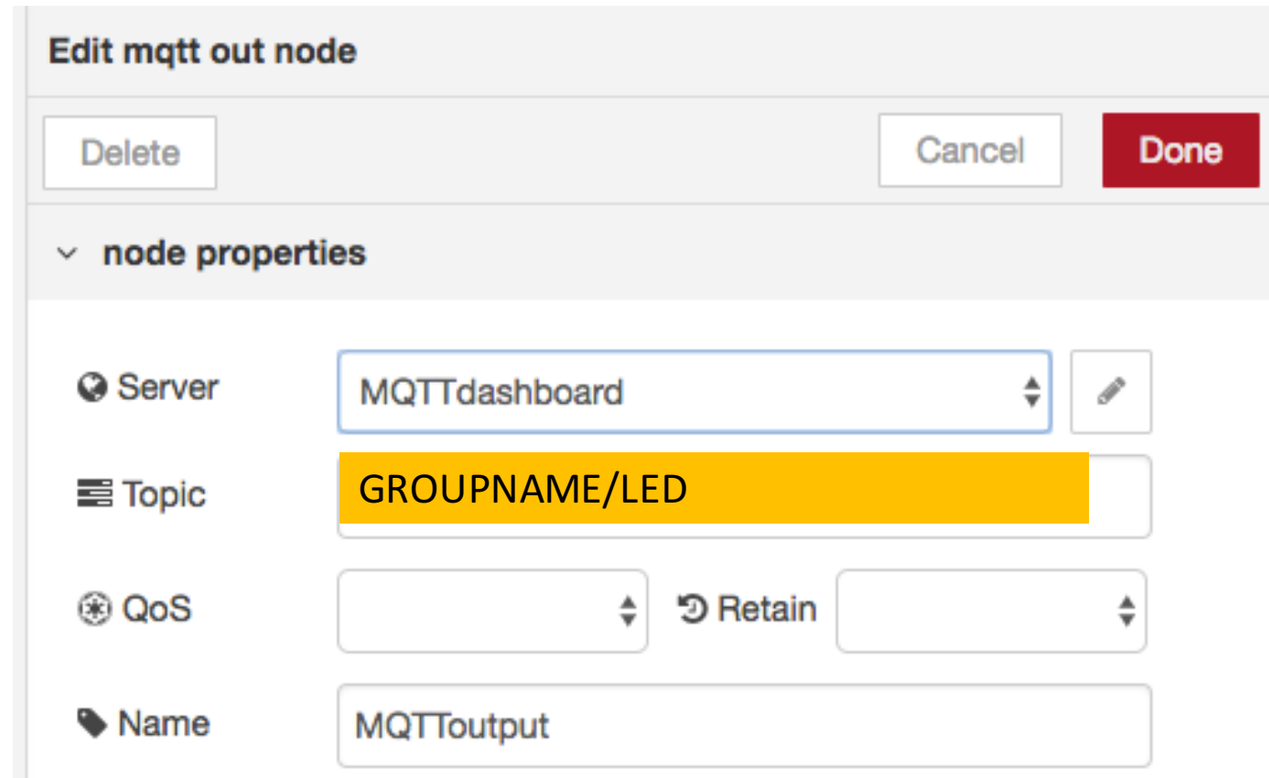
Edit the Switch Node

- Group:
  - Please feel free to use the existing or create a new UI group
- Label: switchLED
- When clicked, send:
  - On Payload (String) on
  - Off Payload (String) off
  - Topic: **unique topic name (to publish for LED control)**
- Hit "Done"



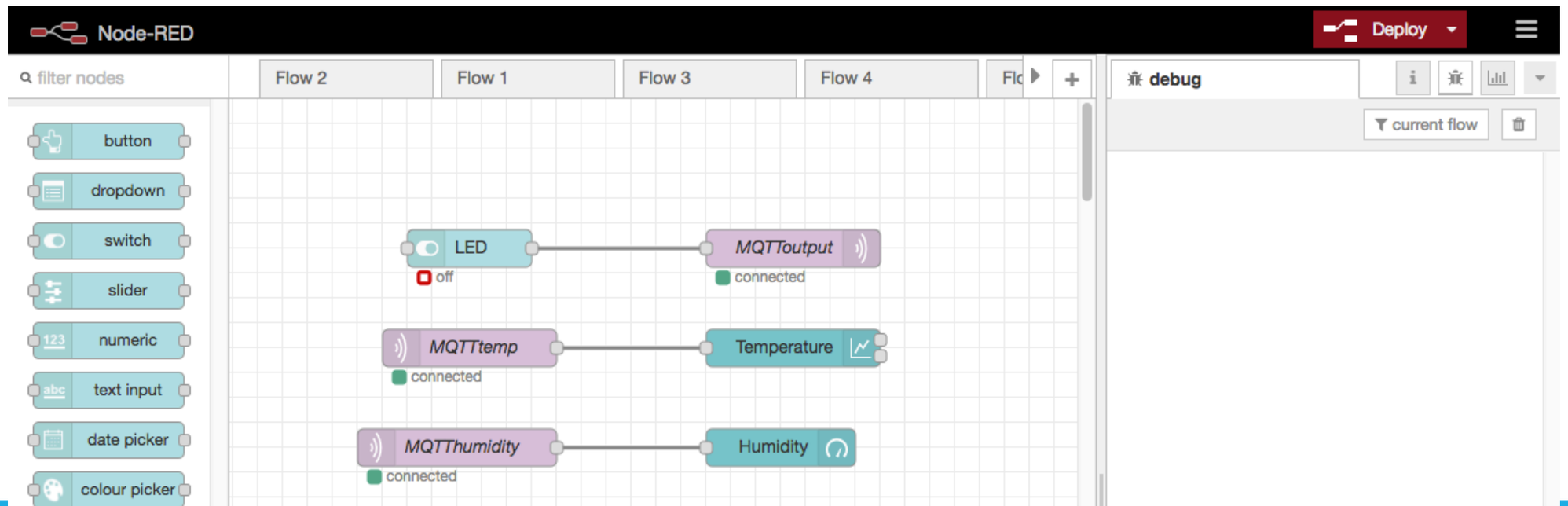
Configure the mqtt Output Node

- Configure the **mqtt** output node to act on the message received from the switch
- Under Server,
  - use same server
- Topic: **unique topic name (to publish for LED control)**
- Name: MQTToutput
- Click “Done”



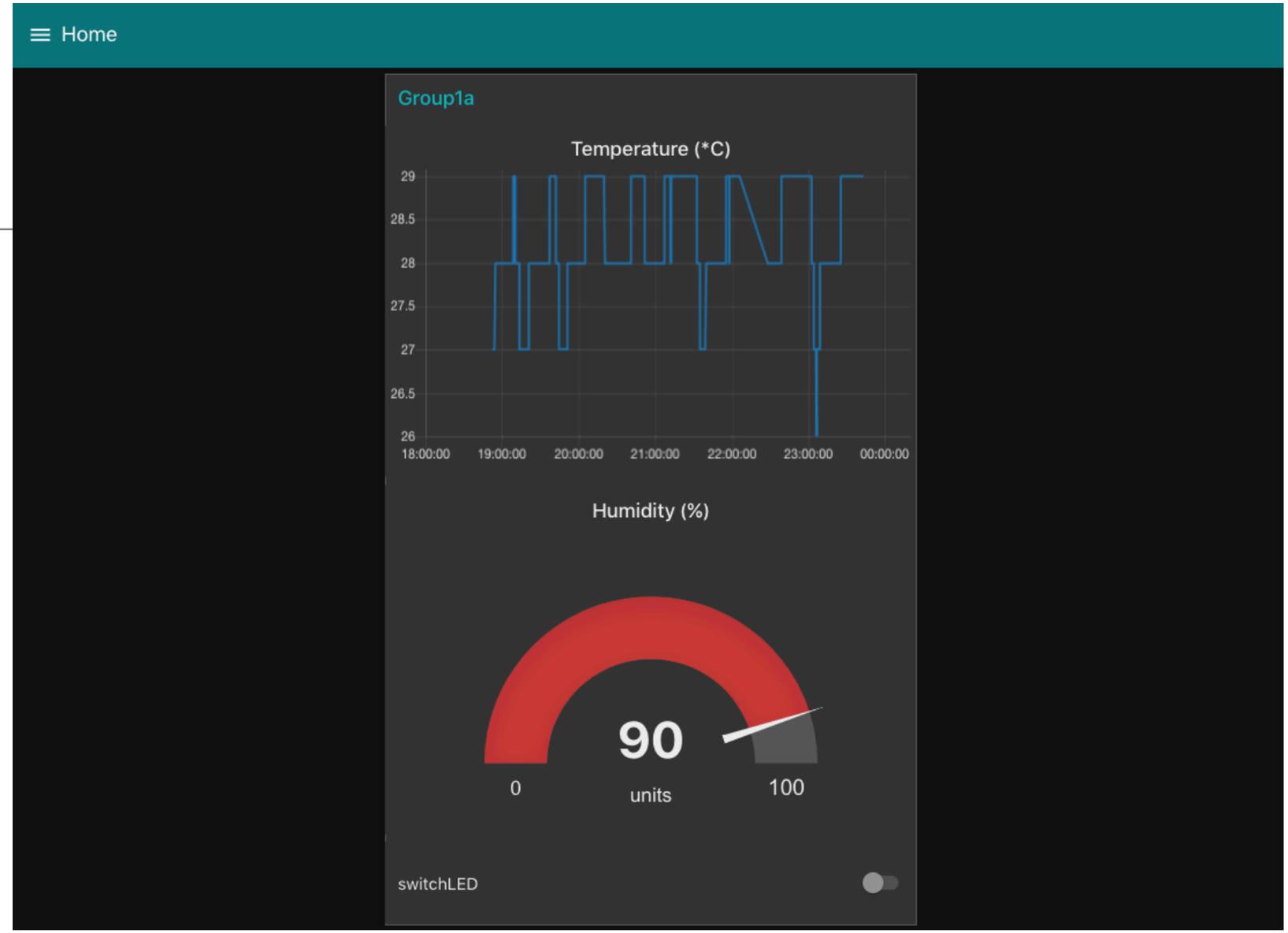
Deploy the Flow

- Your nodeRED flow is now ready for deployment.
- We can deploy it now, but there will be no data coming in.

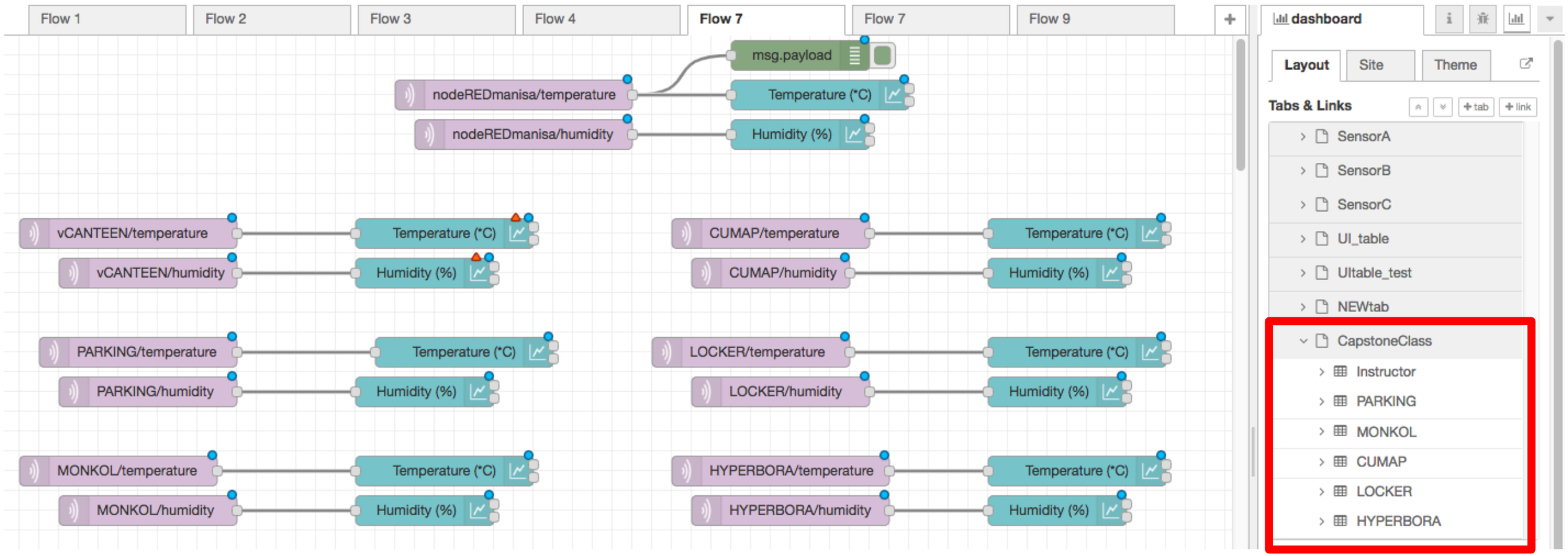


Done!

- You will now see your live plots
- Please switch the button ON/OFF and see if you can control the LED!



You can also Listen to Your Friends' Sensors



Steps

- **Connect your sensor, upload codes to your NodeMCU**
  - Check if this is successful by listening to your group's published messages using MQTT broker
- **Create nodeRED UI**
  - 6 UI groups, displaying sensor data for each group
  - Topics: vCANTEEN/temperature, vCANTEEN/humidity
  - Topics: HYPERBORA/temperature, HYPERBORA/humidity
  - Topics: LOCKER/temperature, LOCKER/humidity
  - Topics: MONKOL/temperatuer, MONKOL/humidity
  - Topics: PARKING/temperature, PARKING/humidity
  - Topics: CUMAP/temperature, CUMAP/humidity

Summary

- This example shows you how to publish the data from NodeMCU to NodeRED.
- You can also create a control button that publishes control message to MQTT broker, where your NodeMCU subscribes to the control message.
- This is one method to retrieve data from multiple sensors
- We will now move on to saving data to the database!!

Introducing Two Types of Databases

- Relational (SQL)
  - MySQL
  - Oracle
  - PostgreSQL
  - Microsoft SQL server

- non-relational (no-SQL)
  - MongoDB
  - Cassandra
  - CouchDB
  - Dynamo DB
  - Firebase

SQL = Structured Query Language

- **S**tructured **Q**uery **L**anguage (SQL) is a language that allows you to write database queries
- **CREATE TABLE** *tablename* (
  column1 datatype,
  column2 datatype,
  column3 datatype,
  …)
- **INSERT INTO** *tablename* (*column1, column2, column3,…*) **VALUES** (*value1, value2, value3, …*)
- **SELECT** * **FROM** *tablename*
- **DELETE FROM** *tablename*
- **DROP TABLE** *tablename*

*** Blue = SQL syntax/keywords**

SQL Database Structure

- It uses tables to store data
- A column is a property of the table
- A row is a data record
- Have a clear schema!!

IdNum	LName	FName
1876	CHIN	JACK
1114	GREENWALD	JANICE
1556	PENNINGTON	MICHAEL
1354	PARKER	MARY
1130	WOOD	DEBORAH

Fixed Schema

- Cannot add additional undefined field(s) along with the data: 999, P., Manisa, mpipatta@vt.edu.
- This will not work b/c the email column does not exist in the table.
- The table has to be modified to accept new value.

Fixed Schema: All records have to follow this schema

IdNum	LName	FName
1876	CHIN	JACK
1114	GREENWALD	JANICE
1556	PENNINGTON	MICHAEL
1354	PARKER	MARY
1130	WOOD	DEBORAH

Relations == Relational Database

Users

ID	Last Name	First Name
1	Brown	John
2	Smith	Robert
…	…	…

Products

ID	Name	Price (Baht)
1	Food1	35
2	Food2	30
…	…	…

Often works with multiple tables that have relationships

Orders

ID	User_ID	Product_ID
1	1	2
2	2	1
3	2	2

NoSQL – using MongoDB as an Example

Humongous

Because it can store a LOT of data

MongoDB Structure

- Database

MongoDB Deployments

Create from backup | Create new

Development and Utility — Single-node deployments intended for environments that do not require high availability.

DEPLOYMENT	TYPE	RAM	SIZE	SIZE ON DISK
ds117422/mongo	Sandbox	shared	116.25 KB	16.00 MB

- Collection

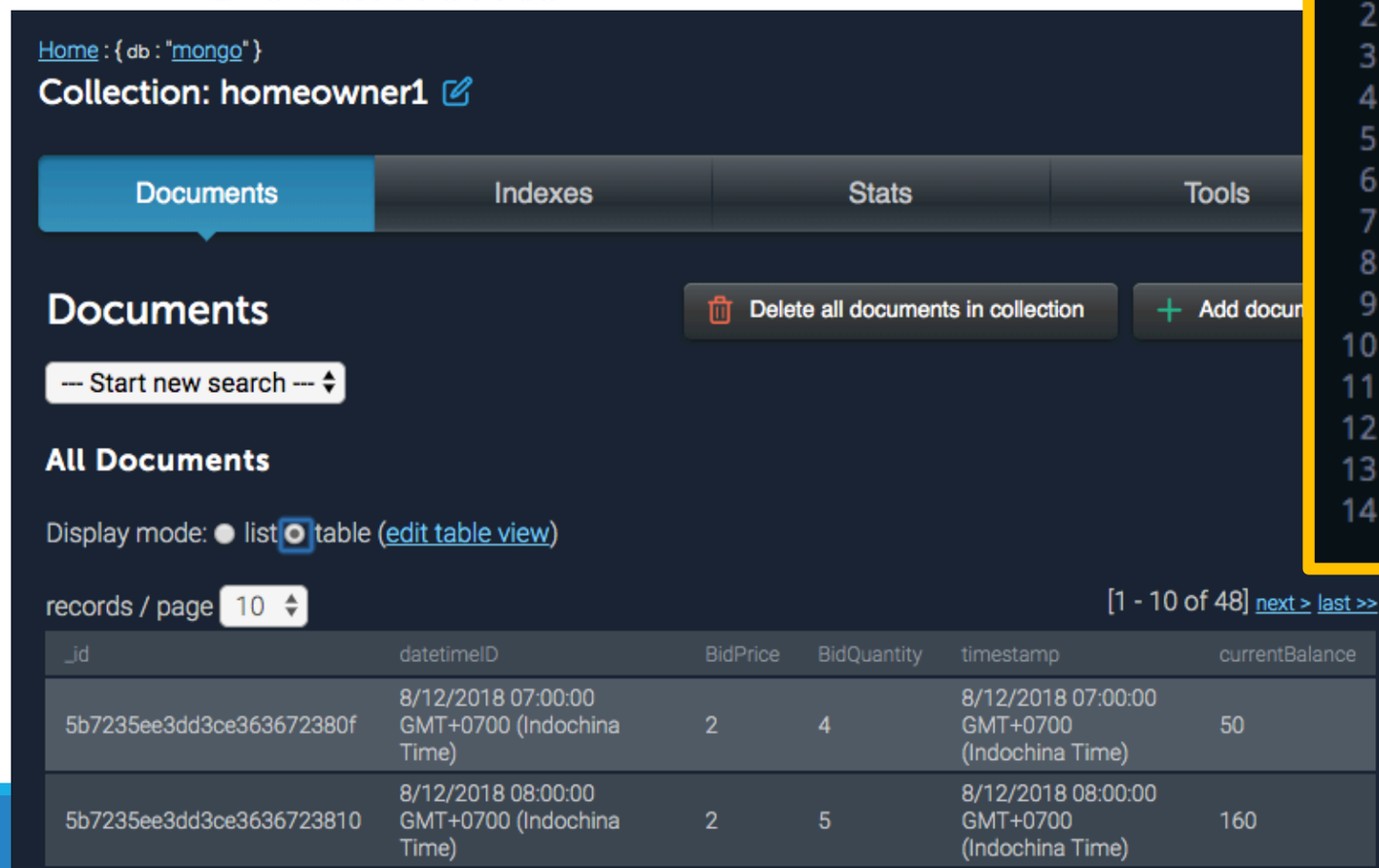
Collections

Delete all collections | Add collection

NAME	DOCUMENTS	CAPPED?	SIZE
arduinoweather	3	false	8.31 KB
balance	51	false	13.56 KB
homeowner1	48	false	13.23 KB

MongoDB Structure (Cont'd)

- Documents




```
{
    "_id": {
        "$oid": "5b7235ee3dd3ce363672380f"
    },
    "datetimeID": {
        "$date": "2018-08-12T00:00:00.000Z"
    },
    "BidPrice": 2,
    "BidQuantity": 4,
    "currentBalance": 50,
    "timestamp": {
        "$date": "2018-08-12T00:00:00.000Z"
    }
}
```

Data Structure

No Schema!

ID:1	Fname: 'John'	Lname: 'Brown'	
ID:2	Fname: 'Robert'		age: 21

No Relations

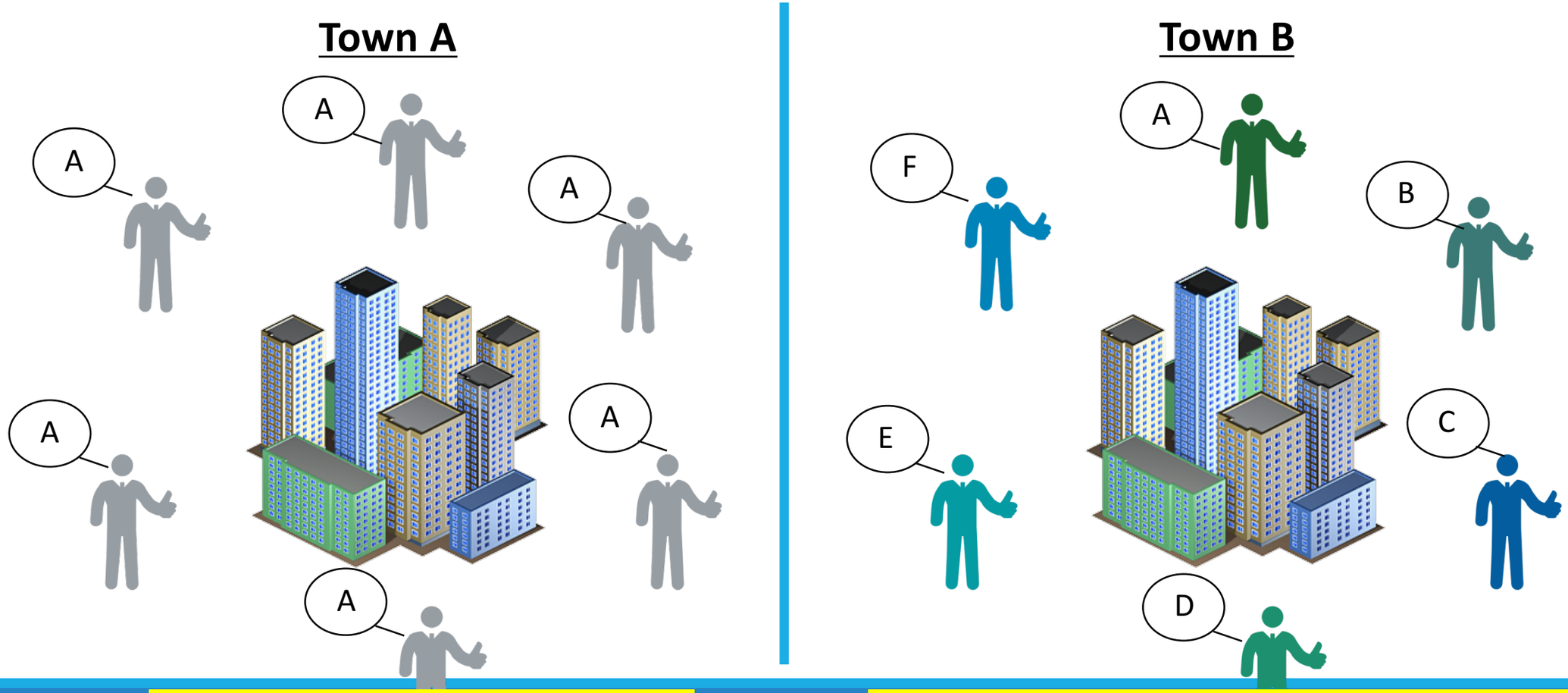
No/Few Relations!

Orders

{ id: 1, user: {id: 1, fname:'John', lname:'Brown'}, product: {id:2, name: 'Food2', price: 30} }
{ id: 2, user: {id: 2, fname:'Robert', lname:'Smith'}, product: {id:1, name: 'Food1', price: 35} }
{ id: 3, user: {id: 2, fname:'Robert', lname:'Smith'}, product: {id:2, name: 'Food2', price: 30} }
{…}

SQL vs NoSQL

Schemas vs Schema-less



Everyone speaks the same language.

No universal understanding. Everyone is different.

SQL vs NoSQL

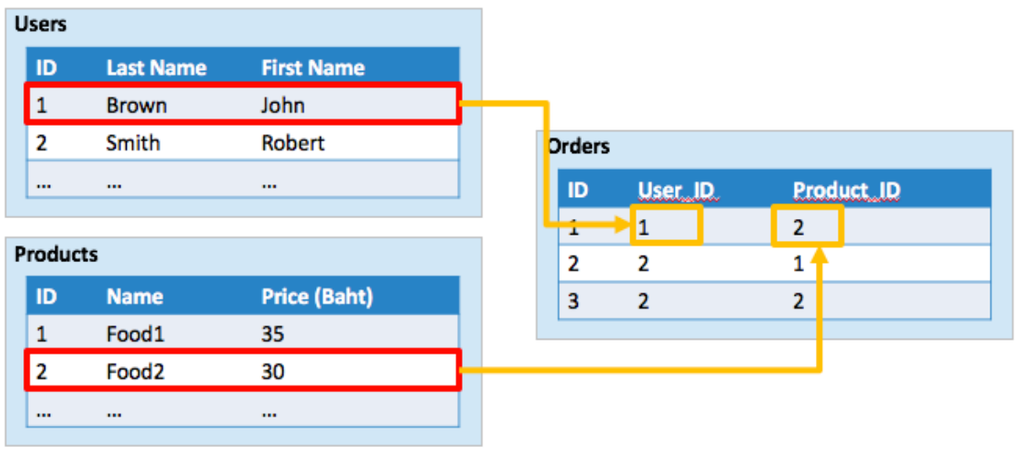
Schemas vs Schema-less

- Pros
  - Ensure data integrity (cannot add additional fields on the fly)
  - Predictable layout
- Cons
  - Restrictive: need pre-defined schemas
  - Require up-front preparation
  - Change in structure would be difficult

- Has dynamic schema for unstructured data
- Pros
  - You can create databases without having to first define their structure
  - New columns can be added without affecting existing rows or application performance
  - Each database can have its own unique structure
  - The syntax can vary from database to database, and
  - You can add fields as you go.
- Cons
  - Cannot guarantee you have certain data

SQL vs NoSQL

Relations! vs no/few Relations



Orders

```
{ id: 1, user: {id: 1, fname:'John', lname:'Brown'}, product: {id:2, name: 'Food2', price: 30} }
{ id: 2, user: {id: 2, fname:'Robert', lname:'Smith'}, product: {id:1, name: 'Food1', price: 35} }
{ id: 3, user: {id: 2, fname:'Robert', lname:'Smith'}, product: {id:2, name: 'Food2', price: 30} }
{...}
```

Data are distributed across multiple tables
+ Update once

Typically, data are put in one collection.
+ fast queries
- Duplicate data

Scalability

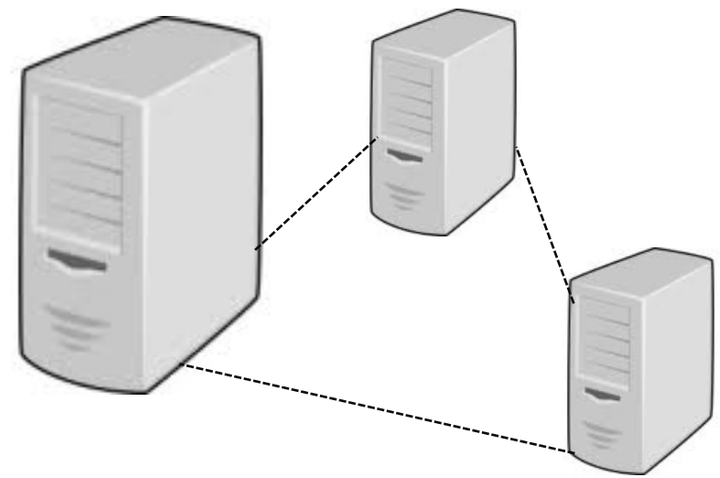
SQL

(Vertically scalable)

+ On a single server
+ Add capability by increasing CPU, RAM, storage
+ Limitations for lots of read & write queries per second

NoSQL

(Horizontally scalable)



+ Can be on several servers
+ Add servers to handle more traffic
+ Great performance for mass read & write

This makes NoSQL databases the preferred choice for large or ever-changing data sets.

Which One to Use

- Either approach can be used
- Typically both systems are used for different data type

SQL

- Have clear schemas
- Use a lot of relations
- Work with data that changes frequently (need to update)

NoSQL

- Good choice for databases with no clear schema definitions.
- Read & write a lot, not so much of updates
- Scaling is important

Virginia Tech's Project (2012-2017)

BEMOSS is a Building Energy Management Open Source Software (BEMOSS) solution that is engineered to improve sensing and control of equipment in small- and medium-sized commercial buildings.

BEMOSS monitoring and control:

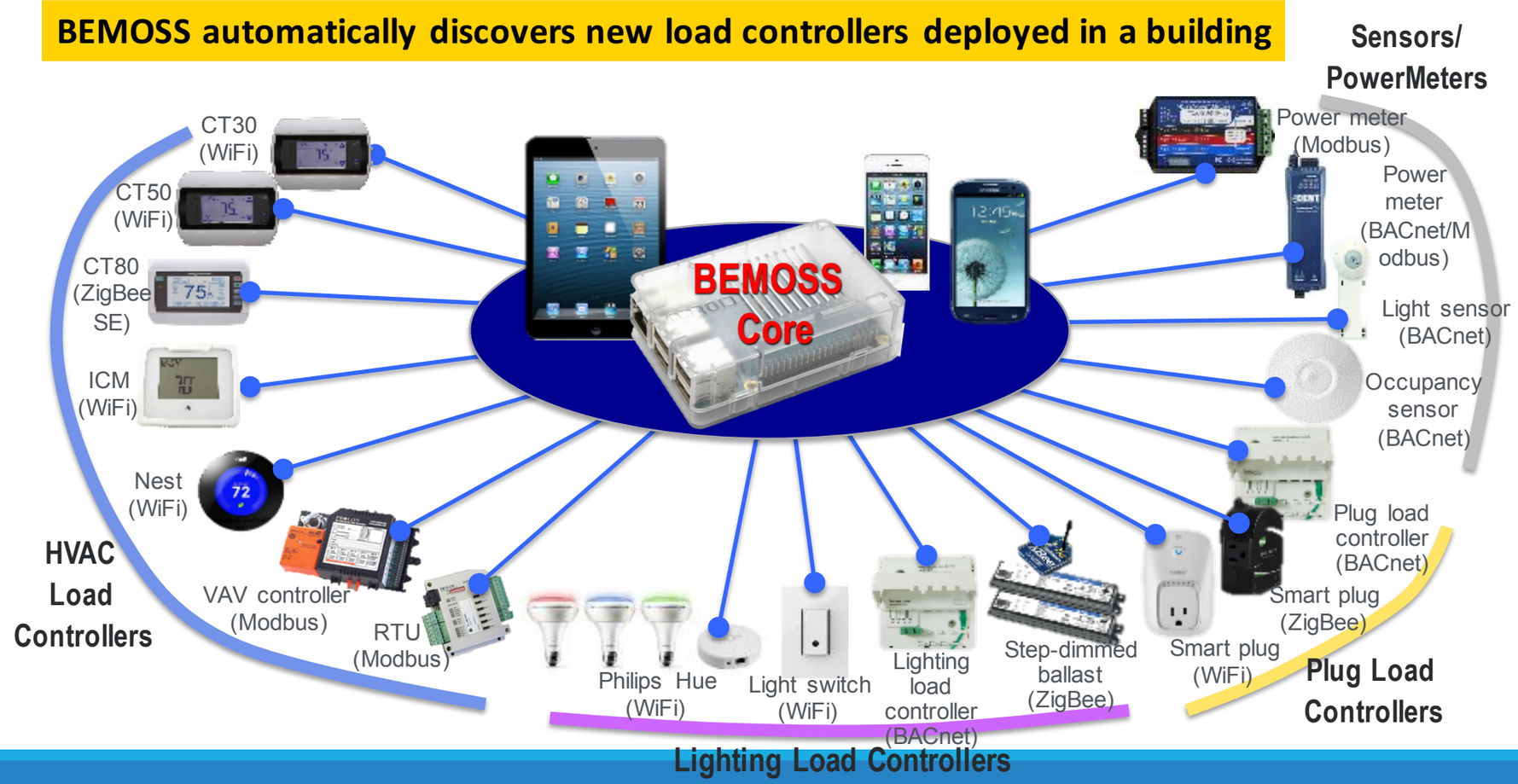
Three major loads in buildings
- HVAC
- Lighting loads
- Plug loads

BEMOSS value:

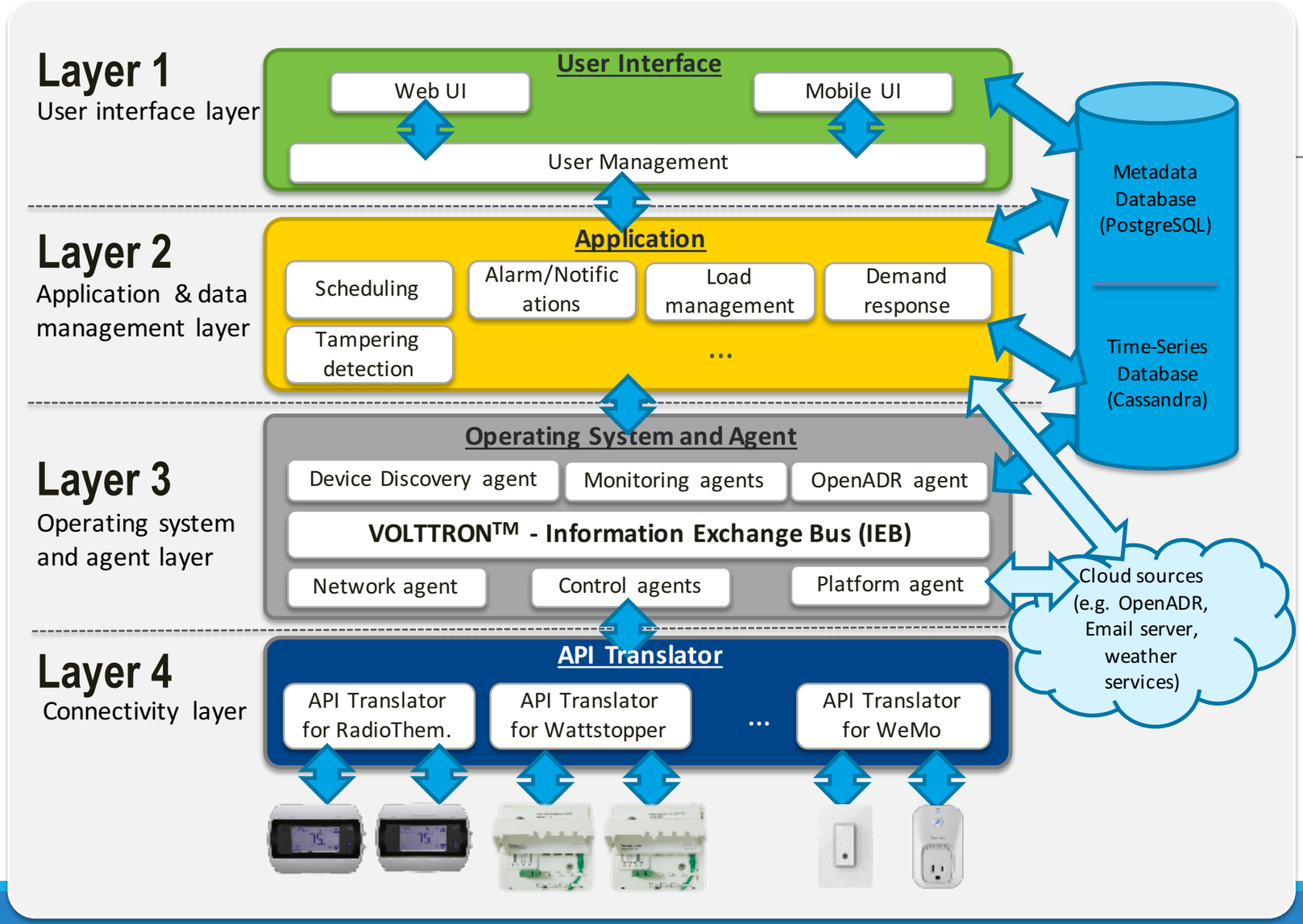
Improves energy efficiency and facilitates demand response implementation in buildings.

BEMOSS Plug & Play

BEMOSS automatically discovers new load controllers deployed in a building



BEMOSS Software Architecture

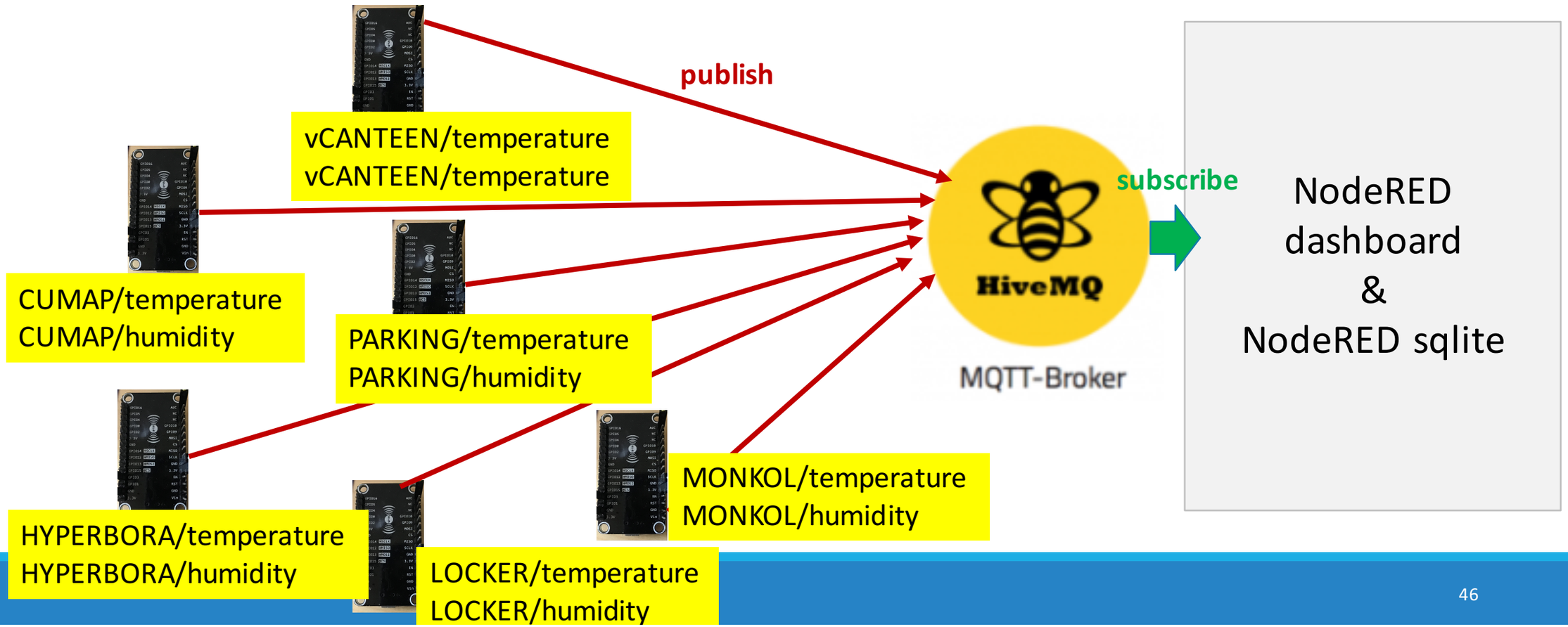


Two types of databases

- SQL
  - User table
  - Device tables
  - Session tables
- NoSQL
  - Time series data

Today's Objective

Get multiple sensors' data, display on the dashboard and save them to a database



Hands-on:

Workshop 7: NodeRED and SQL

Workshop 8: Getting two or more sensor inputs to the dashboard

Workshop 9: Display sensor data on NodeRED UI and save to the database

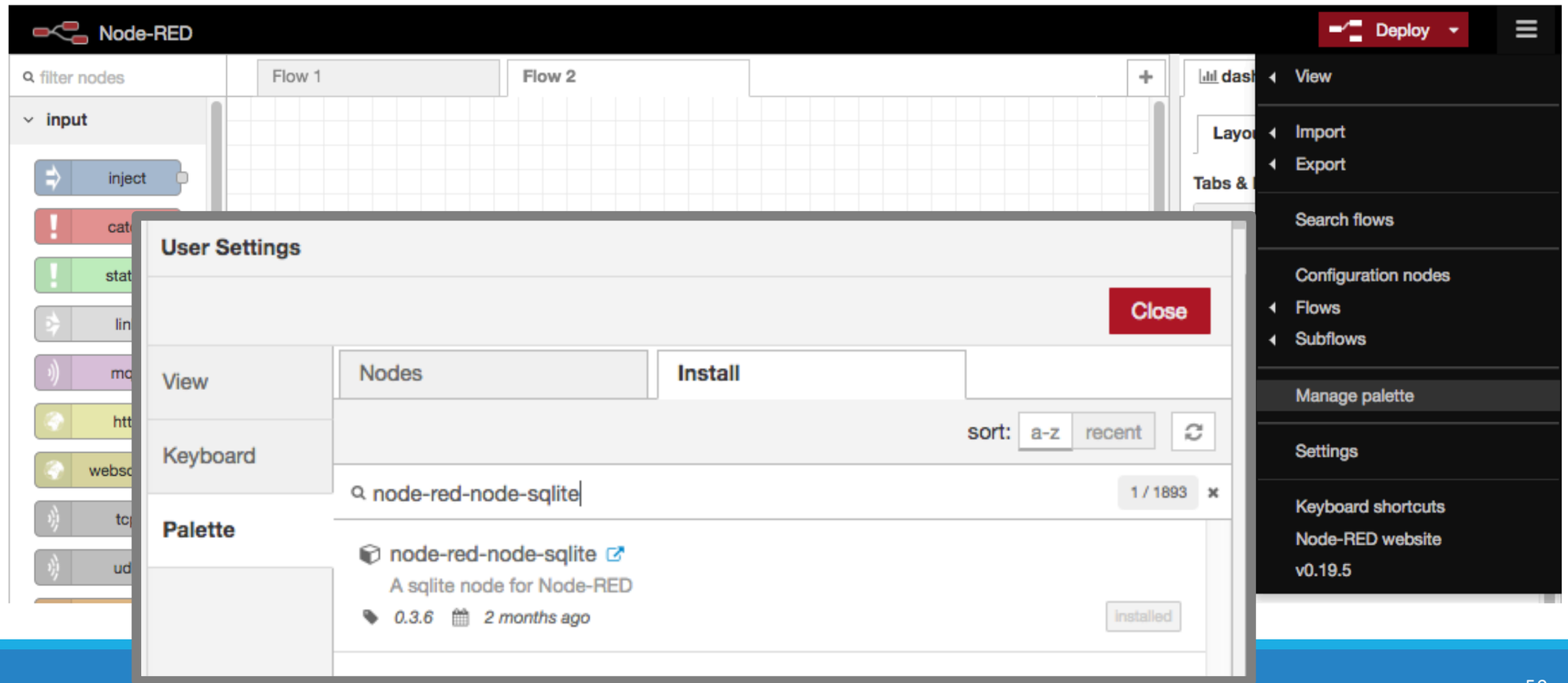
Workshop 7. NodeRED and SQL

Alternative NodeRED version

- Register at https://fred.sensetecnic.com/
- Free for 24 hours

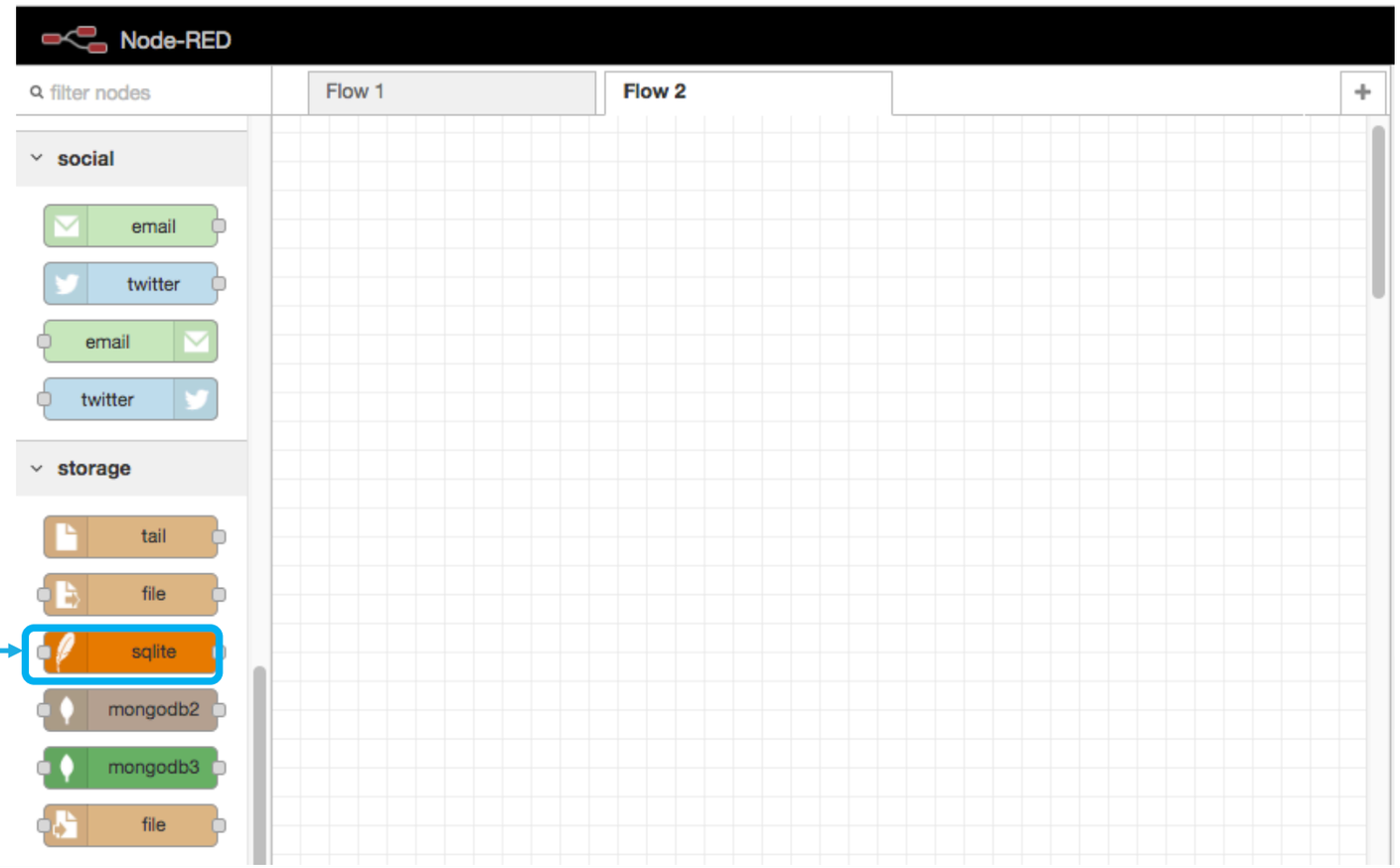
Install node-red-node-sqlite

- Go to ‘Manage palette’ and install ‘node-red-node-sqlite’.



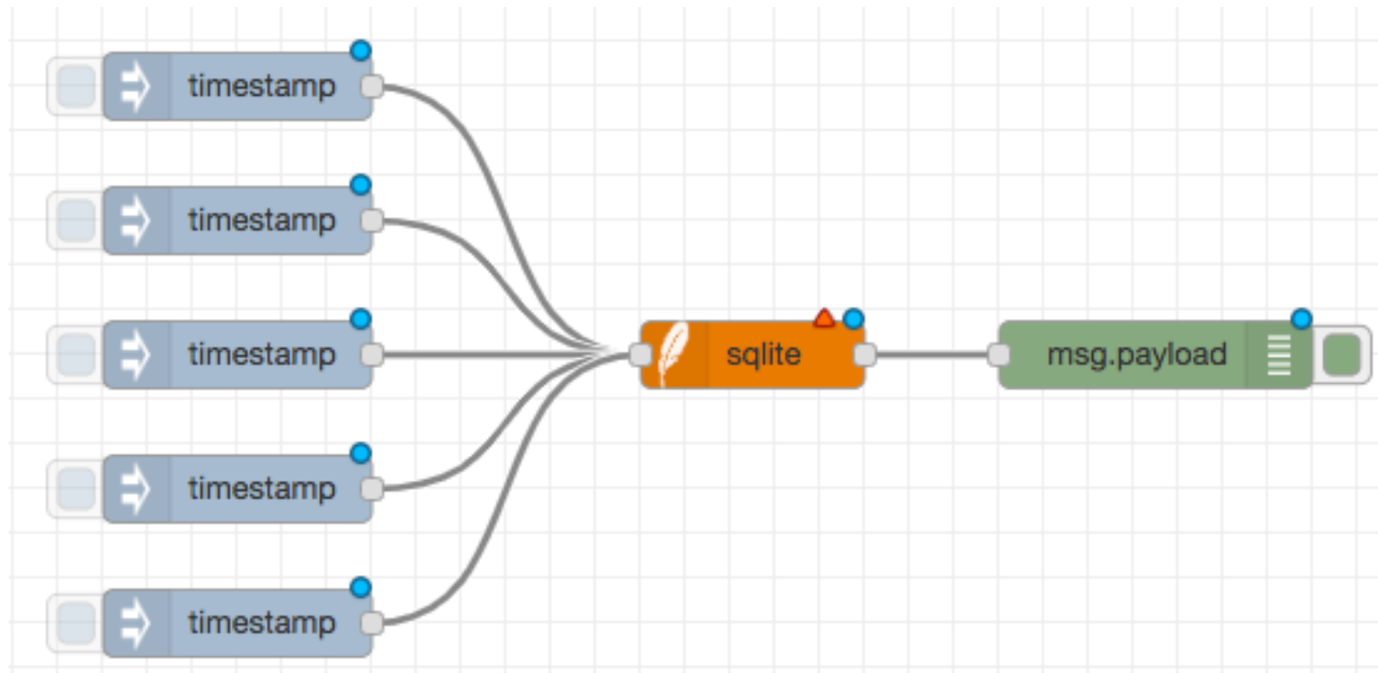
sqlite Node

- **sqlite** node under 'storage' is now available for use



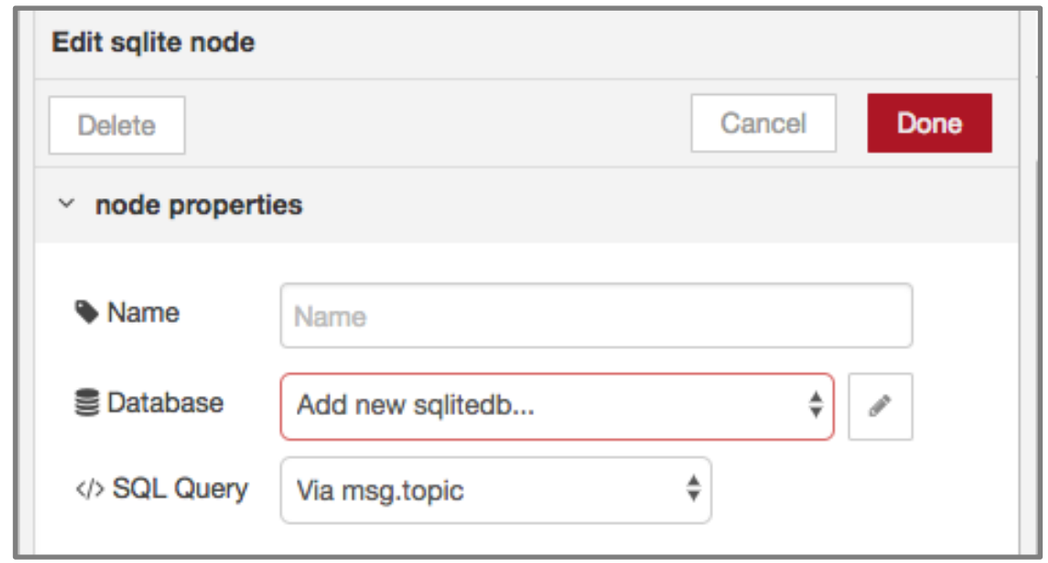
Create a Flow

- Add five **inject** nodes (input), one **sqlite** node (storage) and one **debug** node (output)
- Wire them together as shown

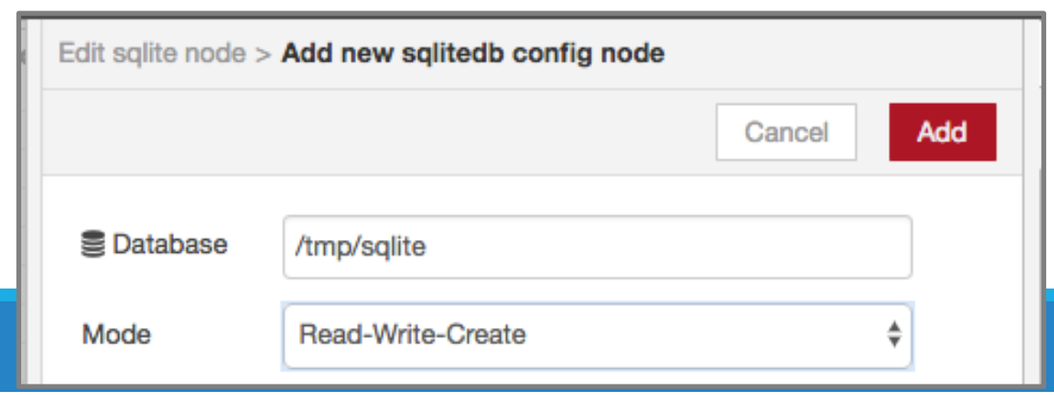


Configure the sqlite node

- Double click the sqlite node:
  - Click 'Add new sqlitedb…'



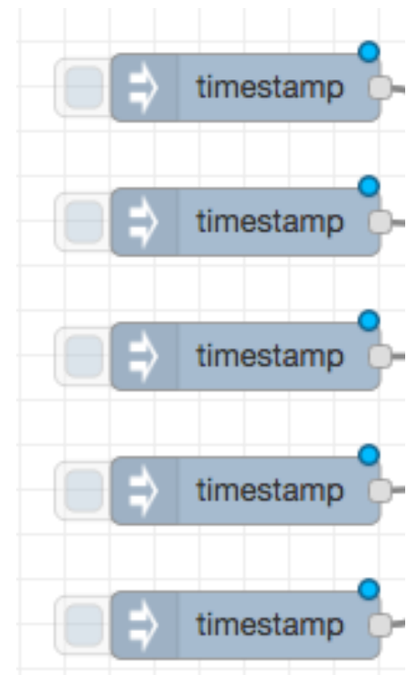
- Enter Database as:
  - /tmp/sqlite
  - Click 'Add'



- Click 'Done'

Configure Five Input Nodes

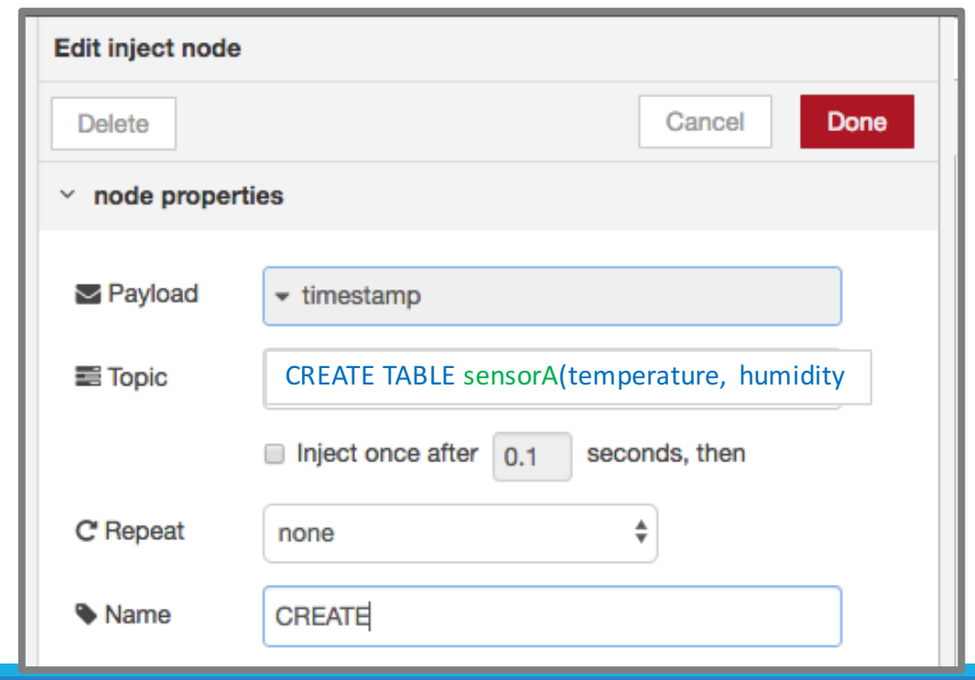
- We will configure the five input nodes to interact with our SQL database as follows:
  - CREATE
  - INSERT
  - SELECT
  - DELETE
  - DROP TABLE



- SQL commands will be used as “Topic”

CREAT

- Topic:
  - **CREATE TABLE sensorA(id INTEGER PRIMARY KEY AUTOINCREMENT, temperature NUMERIC, humidity NUMERIC, currentdate DATE, currenttime TIME, device TEXT)**



- Name:
  - CREATE

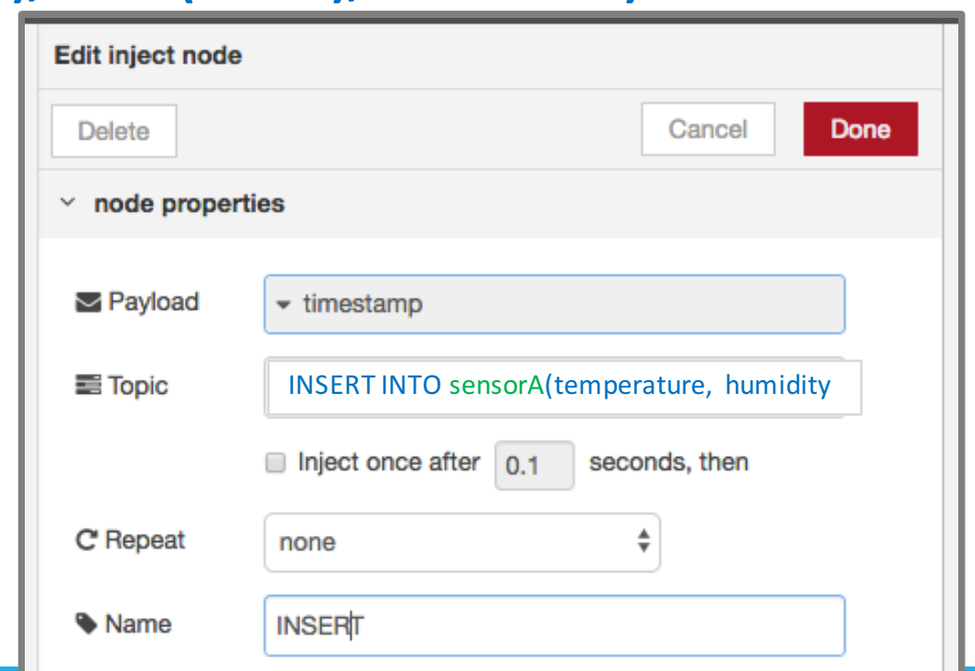
- Click ‘Done’

INSERT

- Topic:
  - **INSERT INTO sensorA(temperature, humidity, currentdate, currenttime, device) values(22.4, 48, date('now'), time('now'), "manual")**

- Name:
  - INSERT

- Click ‘Done’

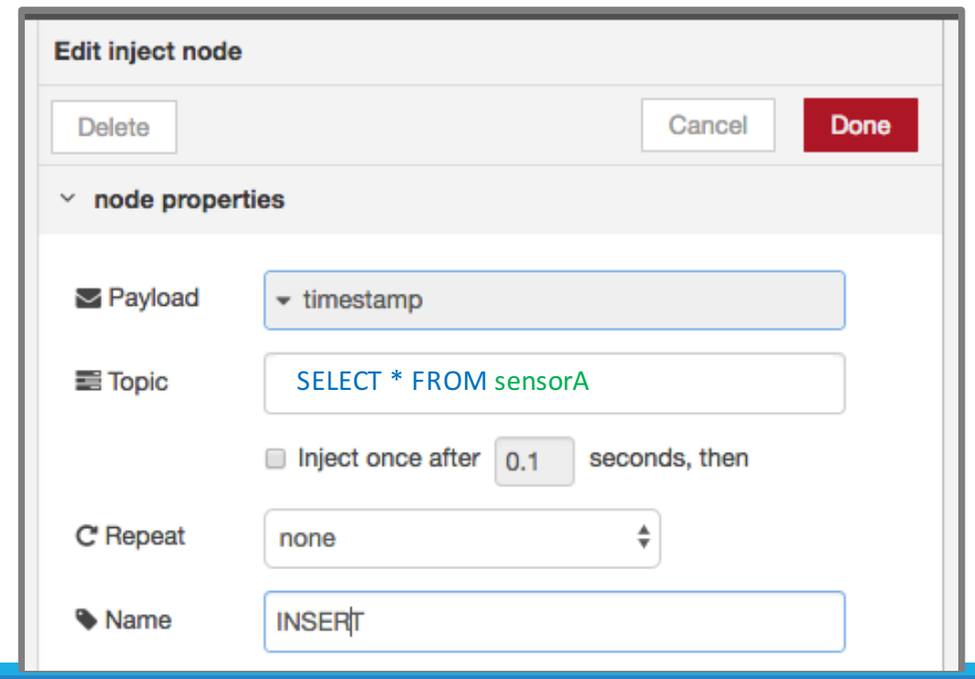


SELECT

- Topic:
  - **SELECT * FROM sensorA**

- Name:
  - SELECT

- Click ‘Done’

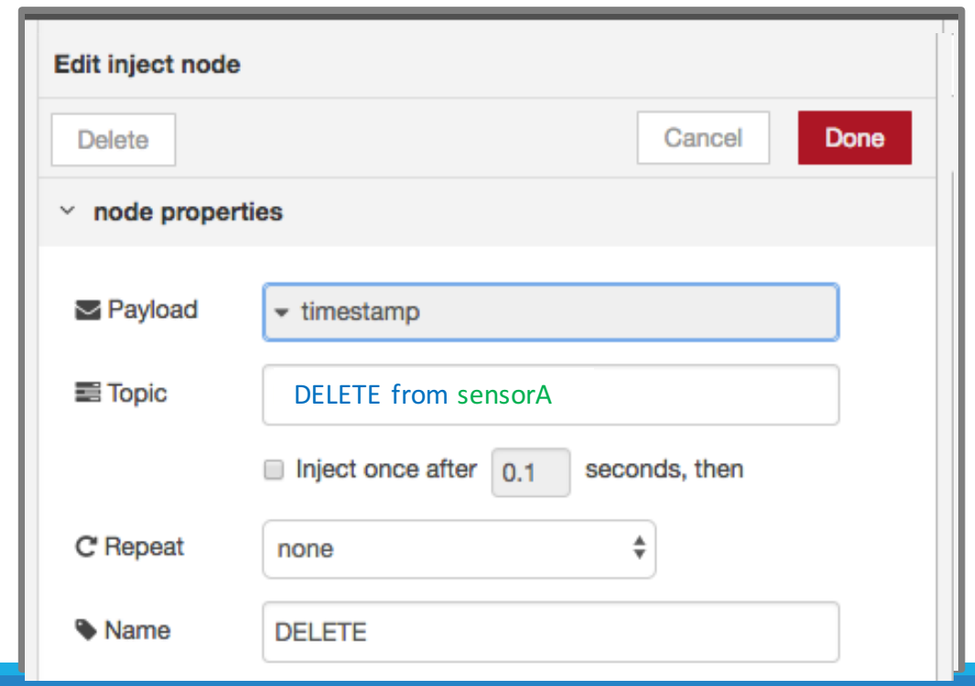


DELETE

- Topic:
  - **DELETE from sensorA**

- Name:
  - DELETE

- Click ‘Done’

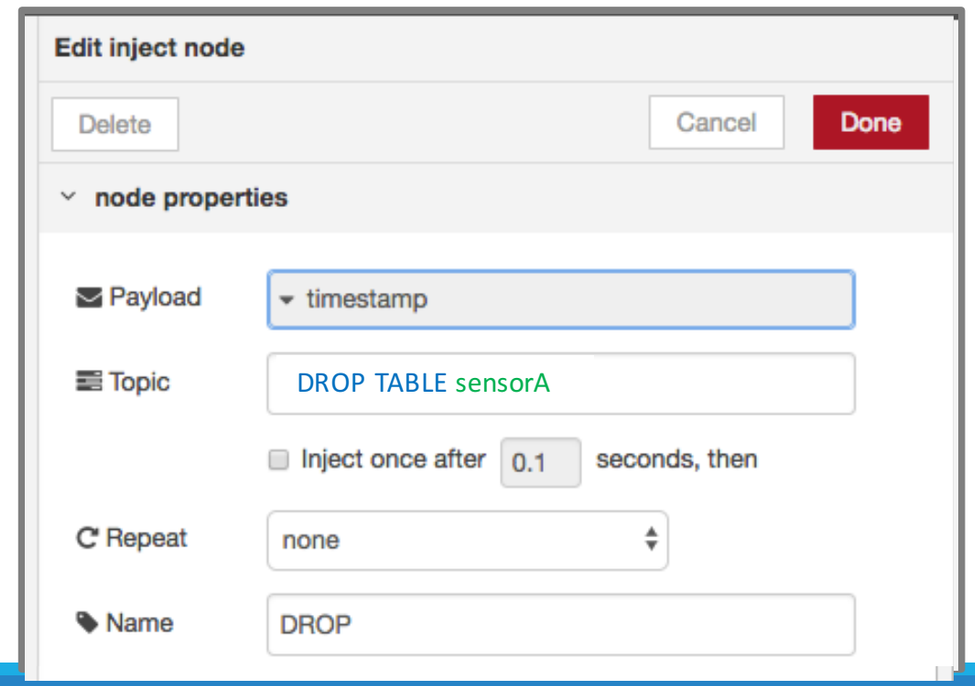


DROP

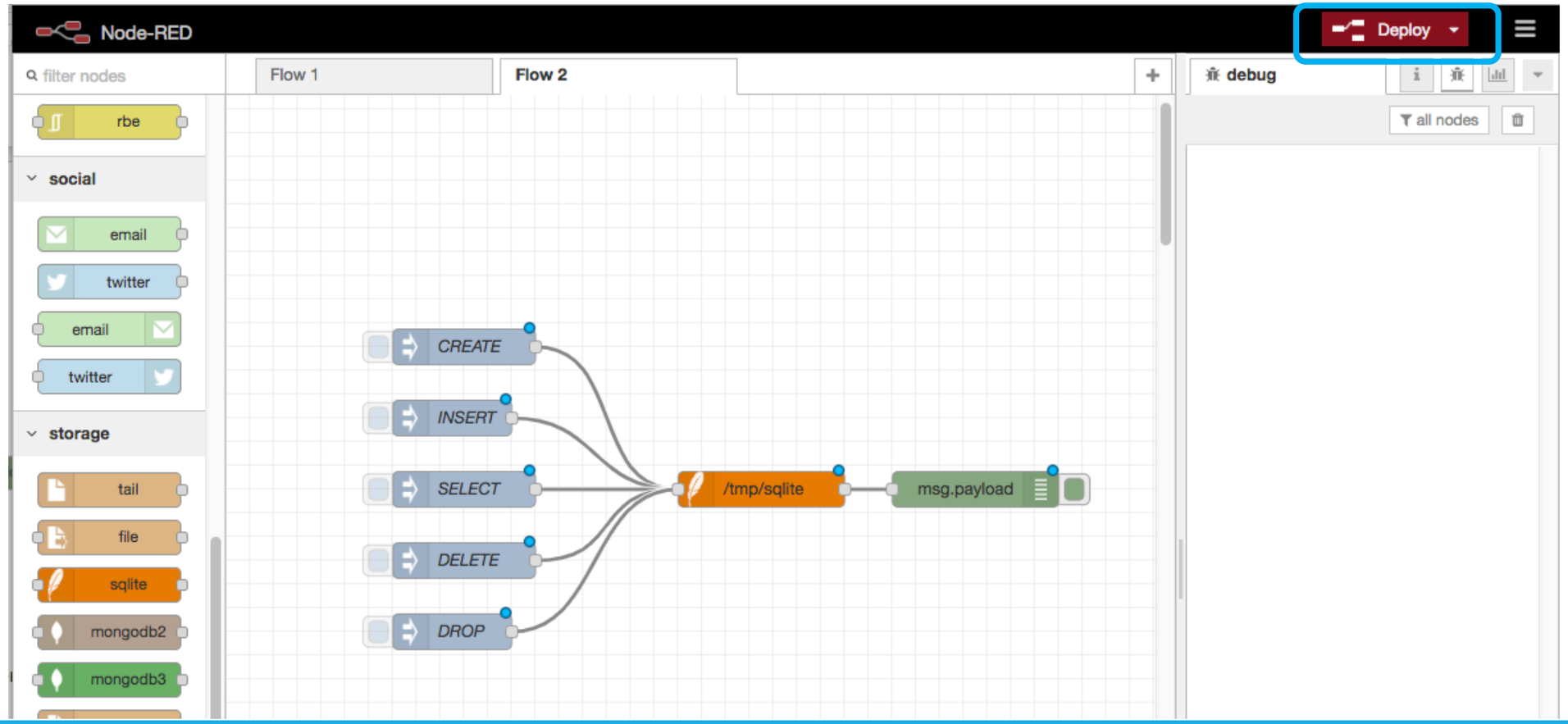
- Topic:
  - **DROP TABLE sensorA**

- Name:
  - DROP

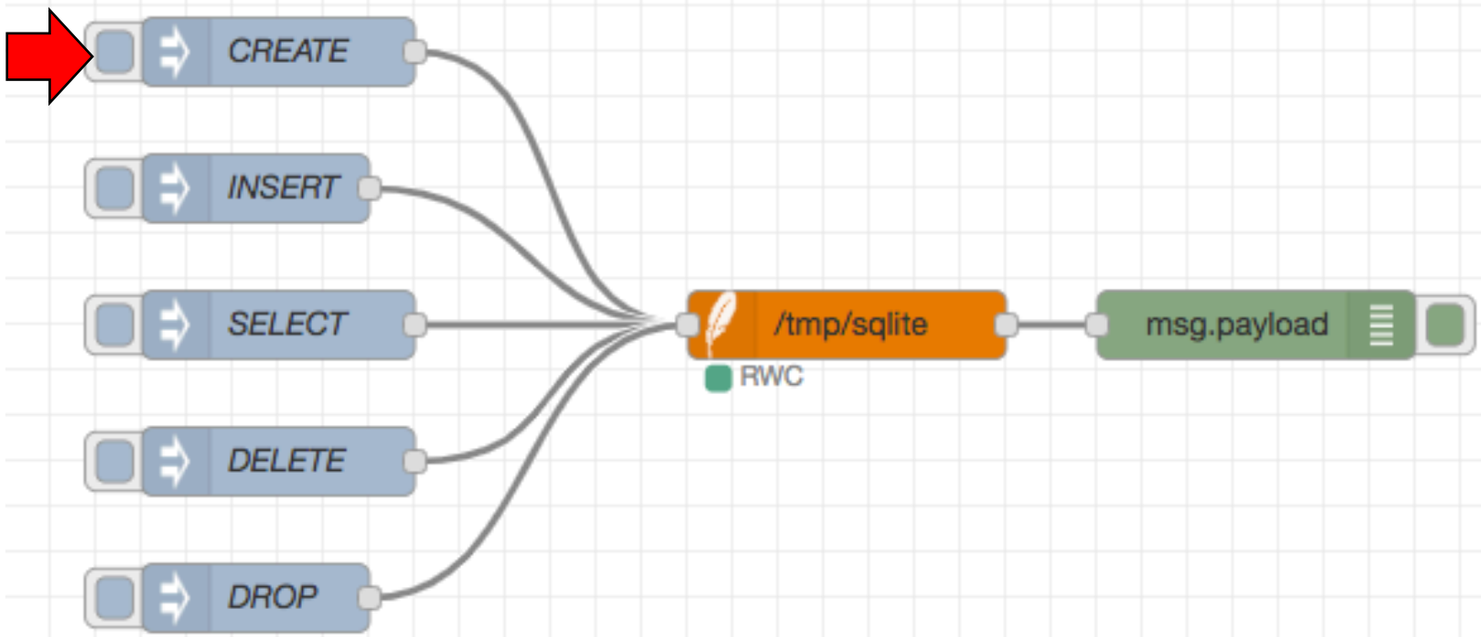
- Click ‘Done’



Deploy your Flow and Test



Test Your Flow:
a) Create a Table

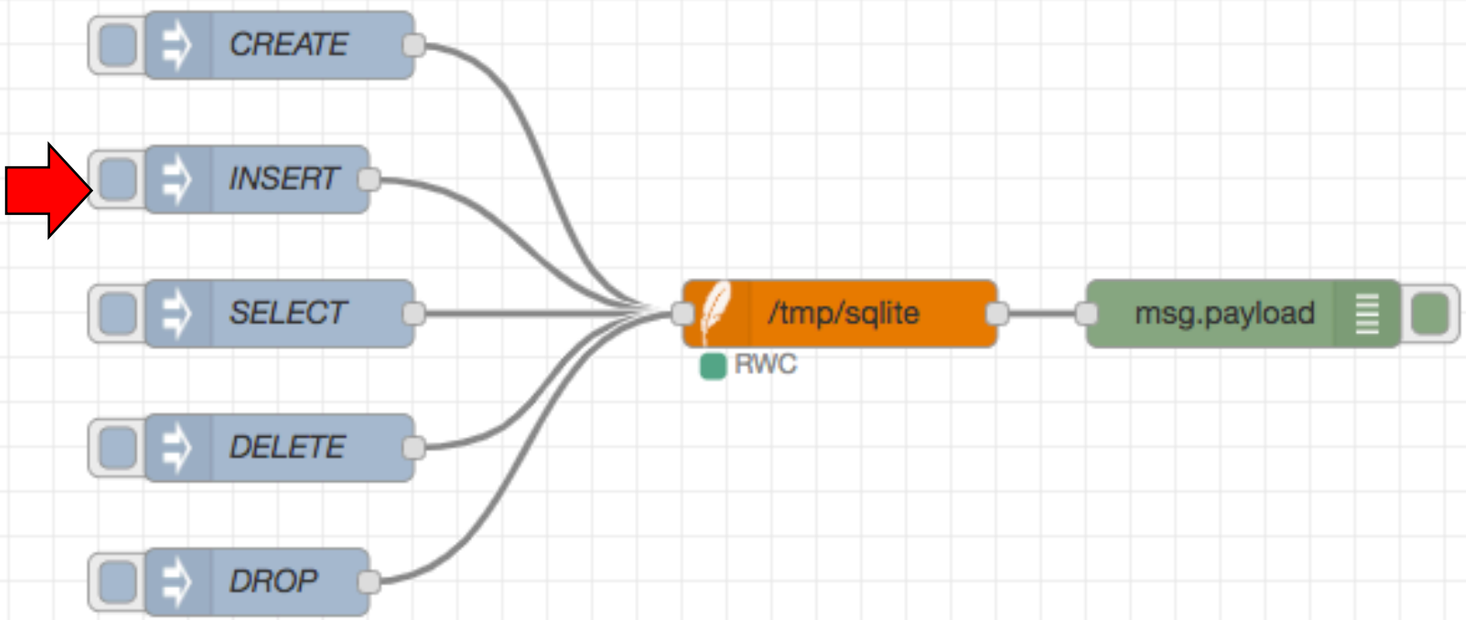


3/6/2019, 1:07:31 PM node: 4082f403.8e4aec

CREATE TABLE dhtreadings(id INTEGER PRIMARY KEY AUTOINCREMENT, temperature NUMERIC, humidity NUMERIC, currentdate DATE, currenttime TIME, device TEXT) : msg.payload : array[0]

[*empty*]

Test Your Flow:
b) Insert a Row in sensorA



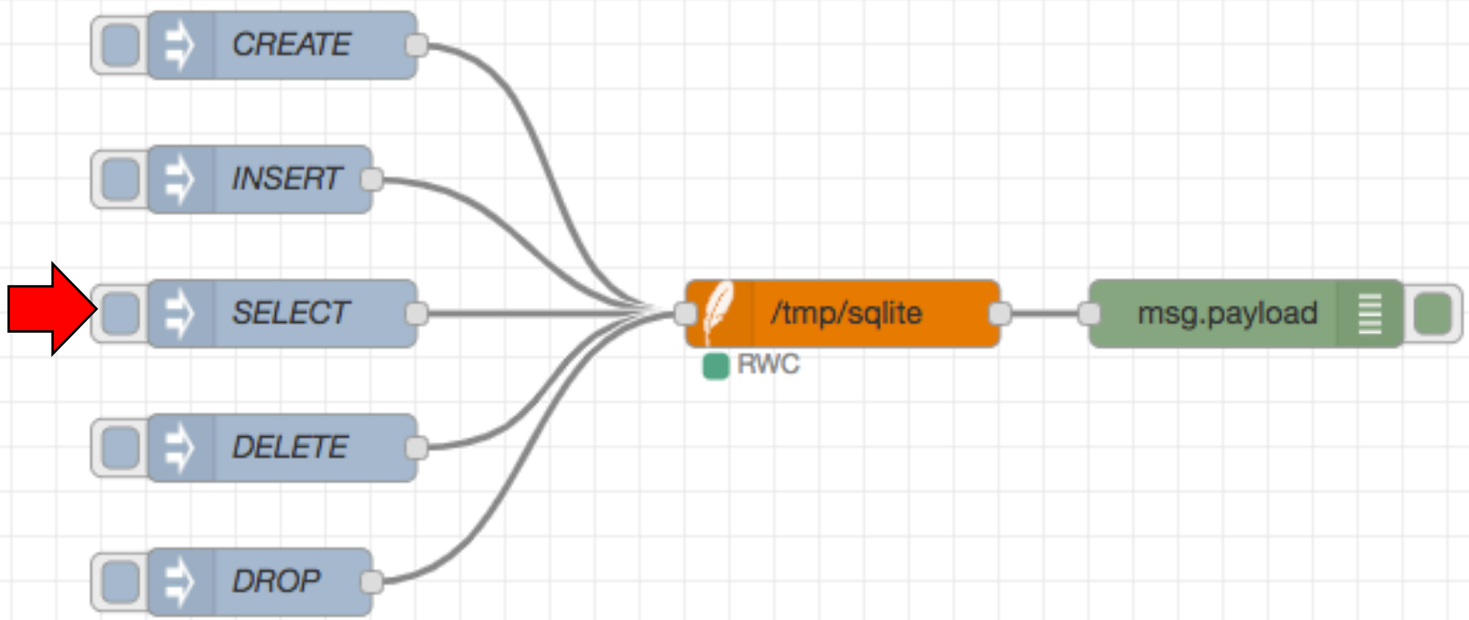
3/6/2019, 1:07:31 PM node: 4082f403.8e4aec
CREATE TABLE dhtreadings(id INTEGER PRIMARY KEY AUTOINCREMENT, temperature NUMERIC, humidity NUMERIC, currentdate DATE, currenttime TIME, device TEXT) : msg.payload : array[0]

[empty]

3/6/2019, 1:07:46 PM node: 4082f403.8e4aec
INSERT INTO dhtreadings(temperature, humidity, currentdate, currenttime, device) values(22.4, 48, date('now'), time('now'), "manual") : msg.payload : array[0]

[empty]

Test Your Flow:
c) Select your Data



3/6/2019, 1:07:31 PM node: 4082f403.8e4aec
CREATE TABLE dhtreadings(id INTEGER PRIMARY KEY AUTOINCREMENT, temperature NUMERIC, humidity NUMERIC, currentdate DATE, currenttime TIME, device TEXT) : msg.payload : array[0]

```
[ empty ]
```

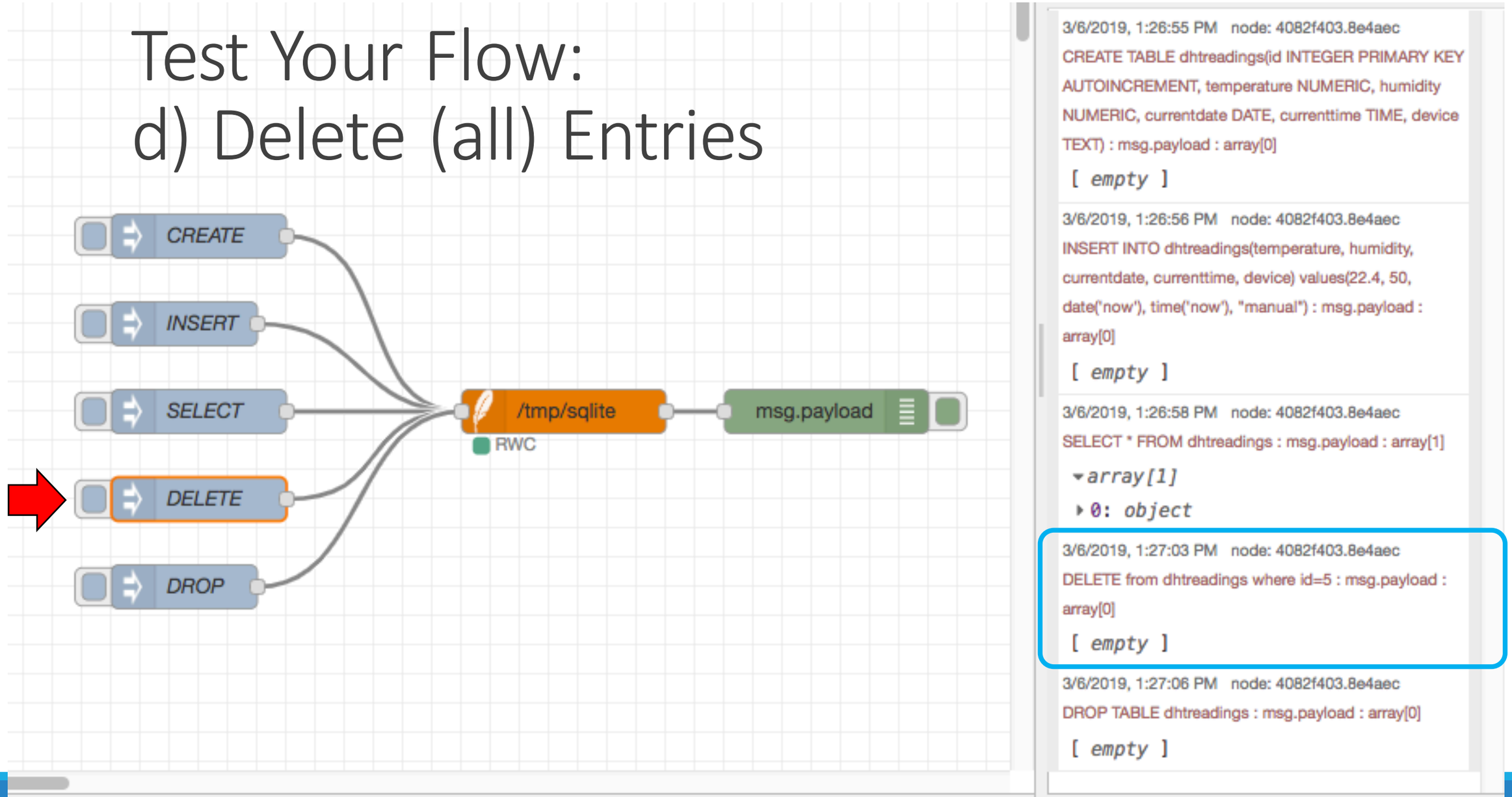
3/6/2019, 1:07:46 PM node: 4082f403.8e4aec
INSERT INTO dhtreadings(temperature, humidity, currentdate, currenttime, device) values(22.4, 48, date('now'), time('now'), "manual") : msg.payload : array[0]

```
[ empty ]
```

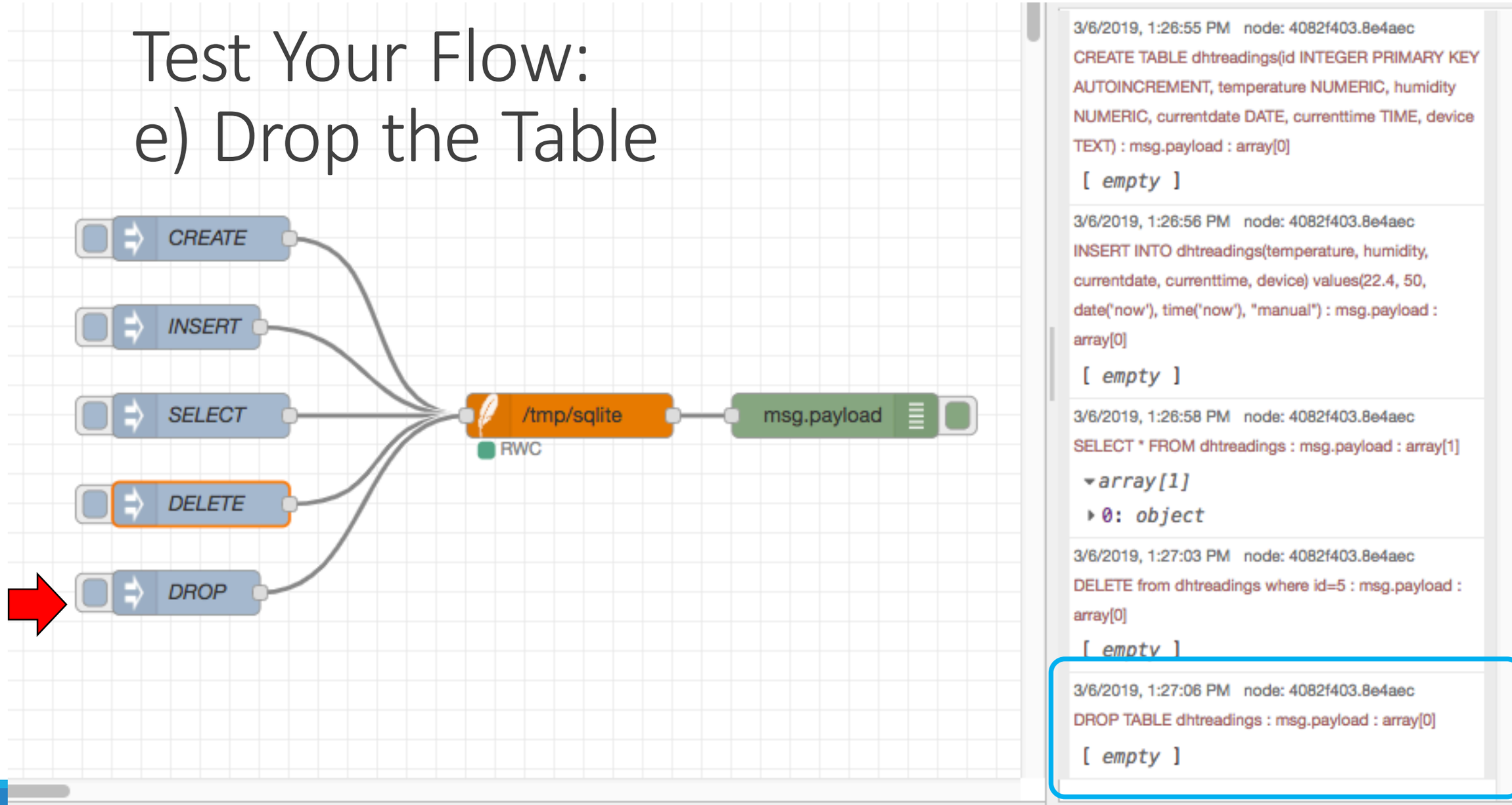
3/6/2019, 1:08:07 PM node: 4082f403.8e4aec
SELECT * FROM dhtreadings : msg.payload : array[1]

```
▾array[1]
 ▾0: object
    id: 1
    temperature: 22.4
    humidity: 48
    currentdate: "2019-03-06"
    currenttime: "06:07:46"
    device: "manual"
```

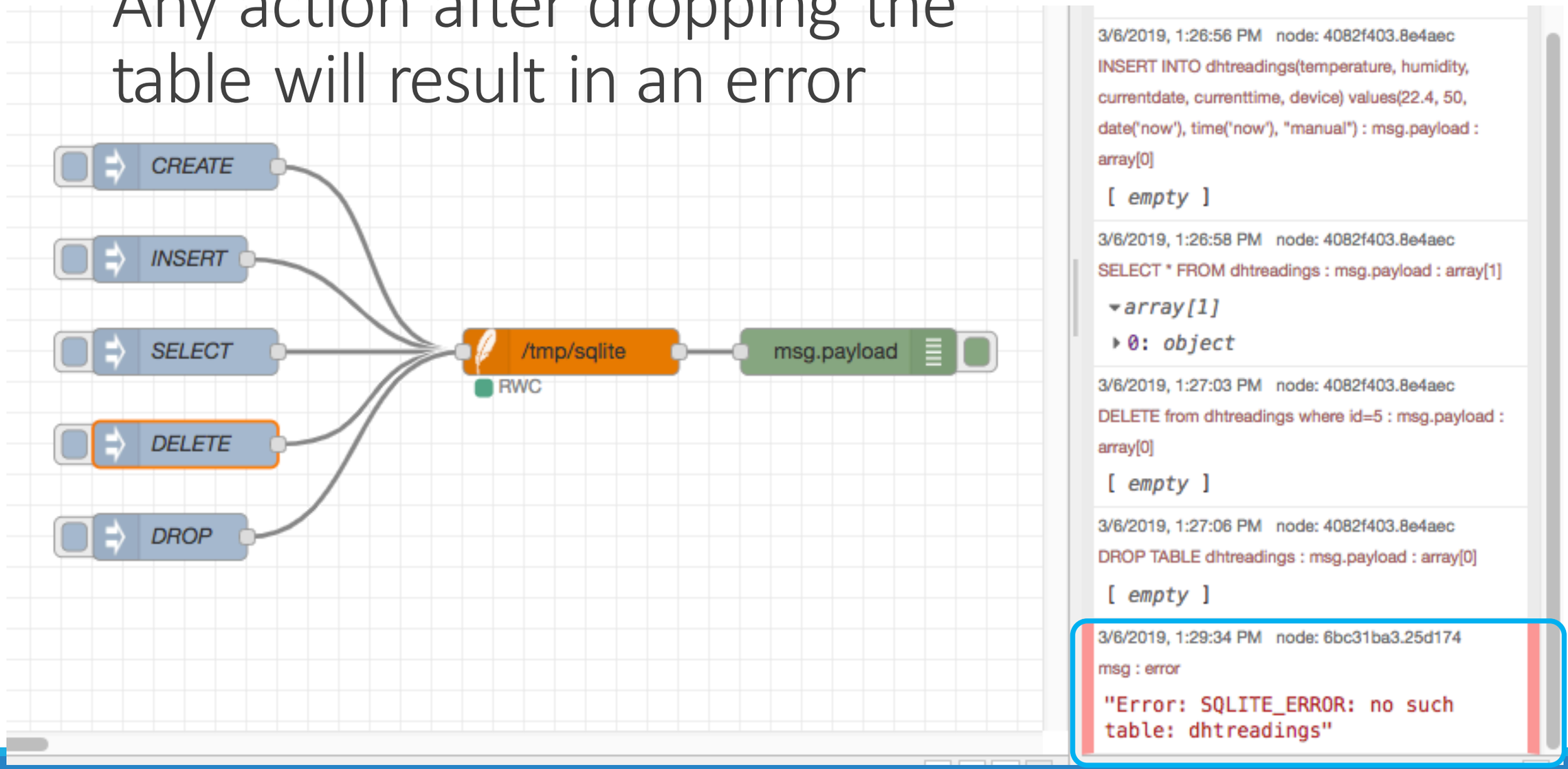
Test Your Flow:
d) Delete (all) Entries



Test Your Flow:
e) Drop the Table



Any action after dropping the table will result in an error



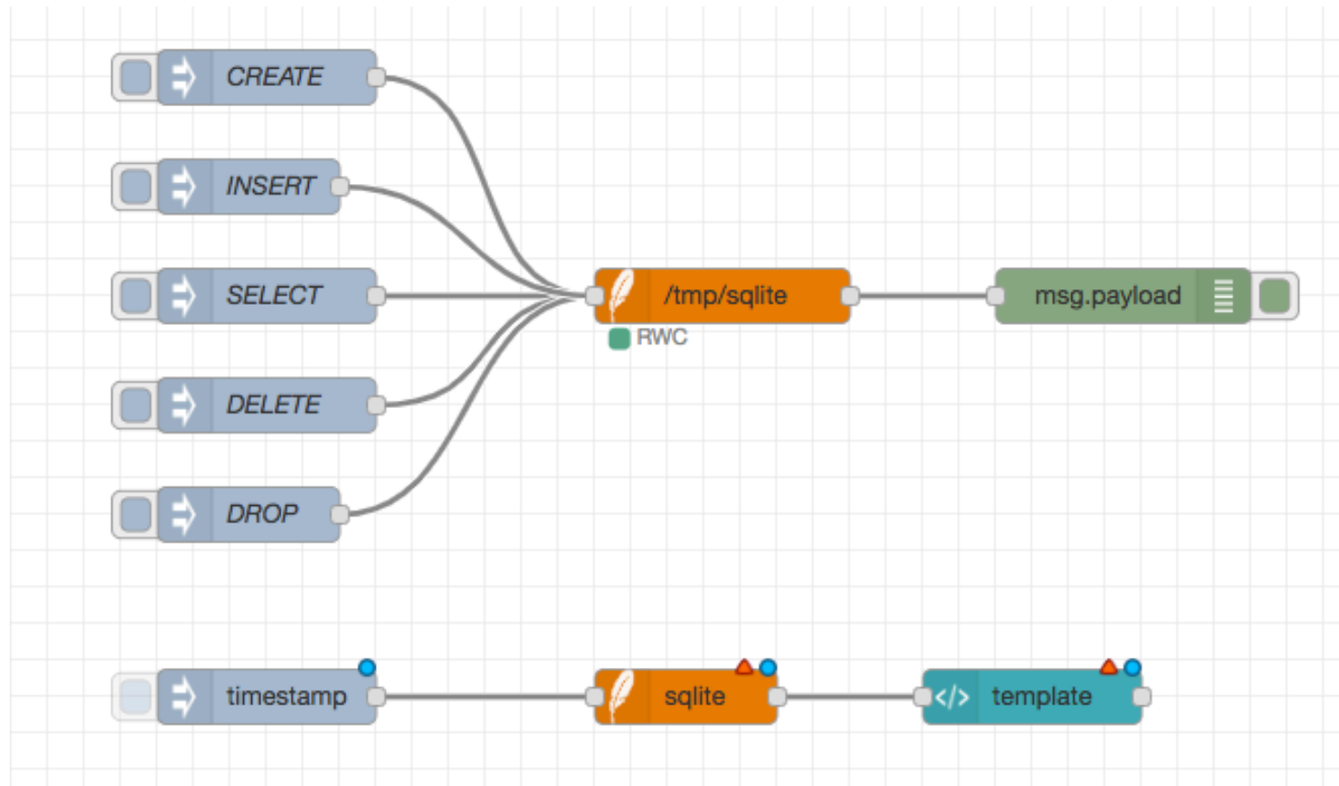
Tip: You may also DELETE a specific row

- If you have more than one entry, and would like to delete a specific entry, use topic:
  - **DELETE from sensorA where id=X**

You can replace X with any specific row id that you would like to delete.

Display Data from the Database in a Table Format

- On the same flow:
  - Drag & drop the **inject** node (input), the **sqlite** node (storage) and the **template** node (dashboard)
  - Wire them as shown

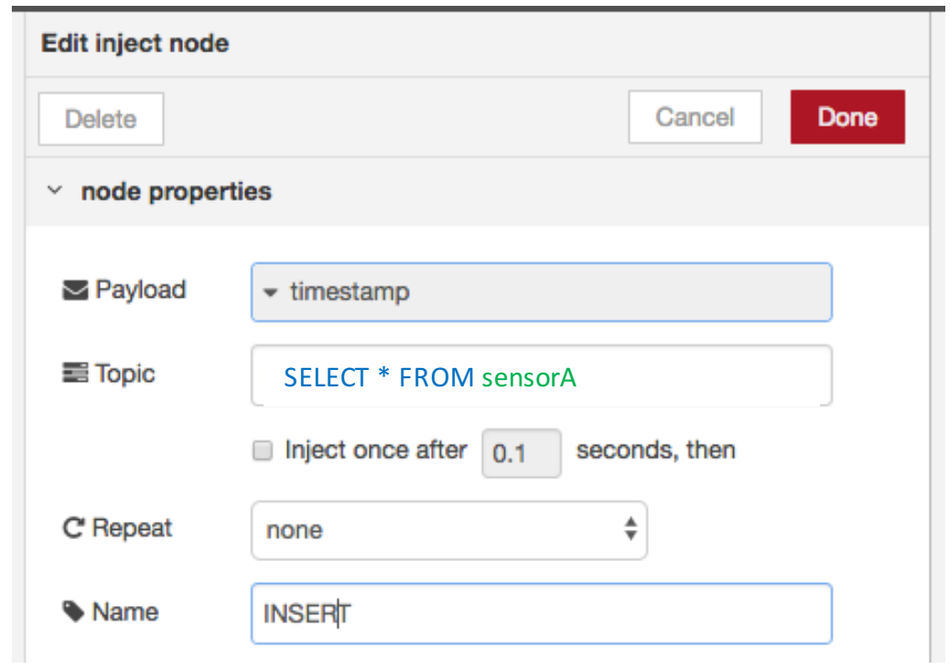


Configure the **Inject** Node

- Topic:
  - **SELECT * FROM sensorA**

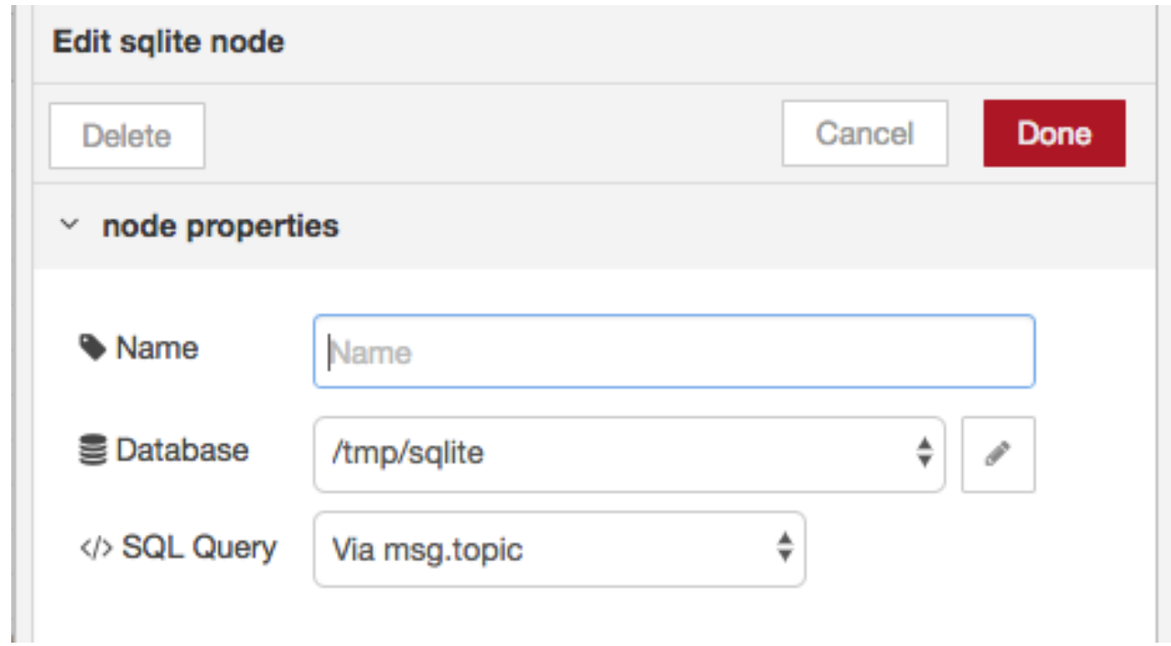
- Name:
  - SELECT

- Click ‘Done’



Configure the **sqlite** node

- Select:
  - /tmp/sqlite
- Click ‘Done’



Configure the **template** node

- Select your UI page to display under Group
- Put this code to the template →
- Click ‘Done’

```
<table style="width:100%">
<tr>
    <th>Index</th>
    <th>Date</th>
    <th>Time</th>
    <th>Temperature</th>
    <th>Humidity</th>
    <th>Device</th>
</tr>
<tr ng-repeat="x in msg.payload | orderBy:'-'">
    <td>{{$index}}</td>
    <td>{{msg.payload[$index].currentdate}}</td>
    <td>{{msg.payload[$index].currenttime}}</td>
    <td>{{msg.payload[$index].temperature}}</td>
    <td>{{msg.payload[$index].humidity}}</td>
    <td>{{msg.payload[$index].device}}</td>
</tr></table>
```

Deploy Your Code and Test

- Test your new flow
- Check your UI, you should see a table!

Default

Index	Date	Time	Temperature	Humidity	Device
0	2019-03-07	11:38:36	22.4	48	manual
1	2019-03-07	11:39:14	22.4	48	manual

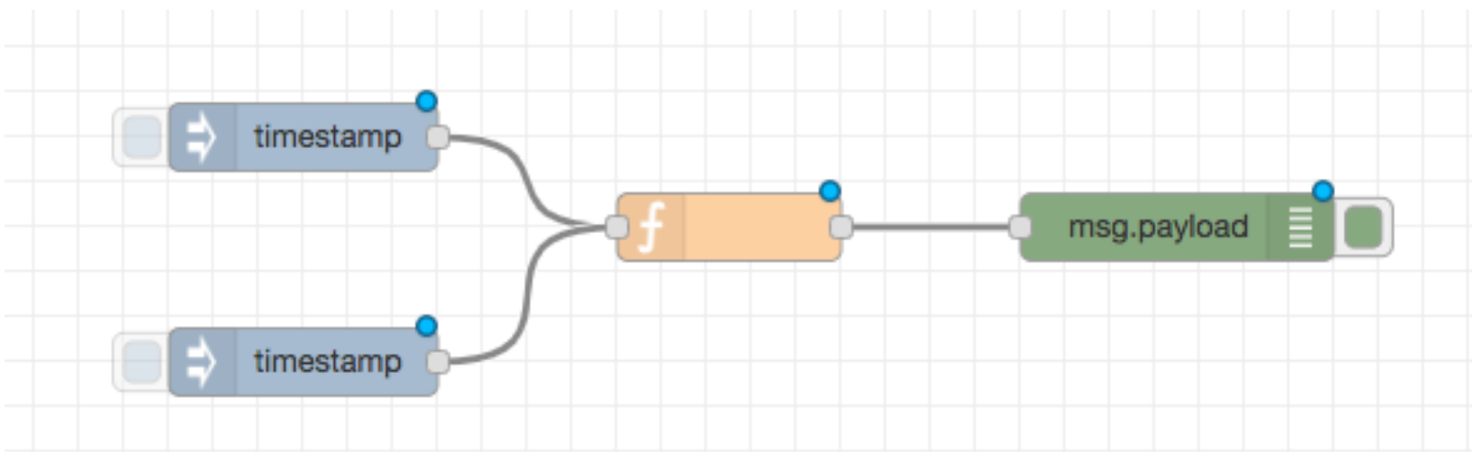
Summary

- This example shows you how to send simple SQL queries to an SQLite database with Node-RED.
  - CREATE
  - INSERT
  - SELECT
  - DELETE
  - DROP TABLE
- You also learn to retrieve the data from your **sqlite** database.
- This concept can be applied to create a complex database that stores useful readings.

Workshop 8. Getting two or more sensor inputs to the dashboard

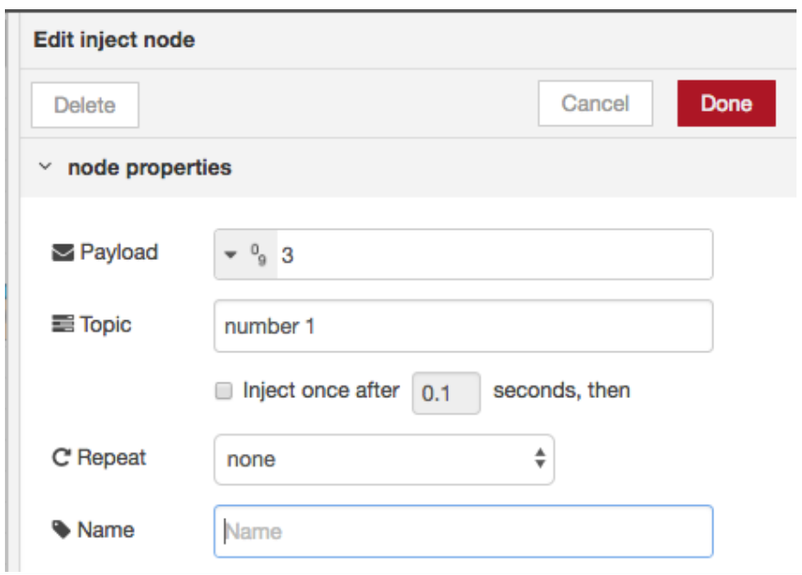
Create a Flow

- Add two **inject** nodes (input), one **function** node (function) and one **debug** node (output)
- Wire them together as shown

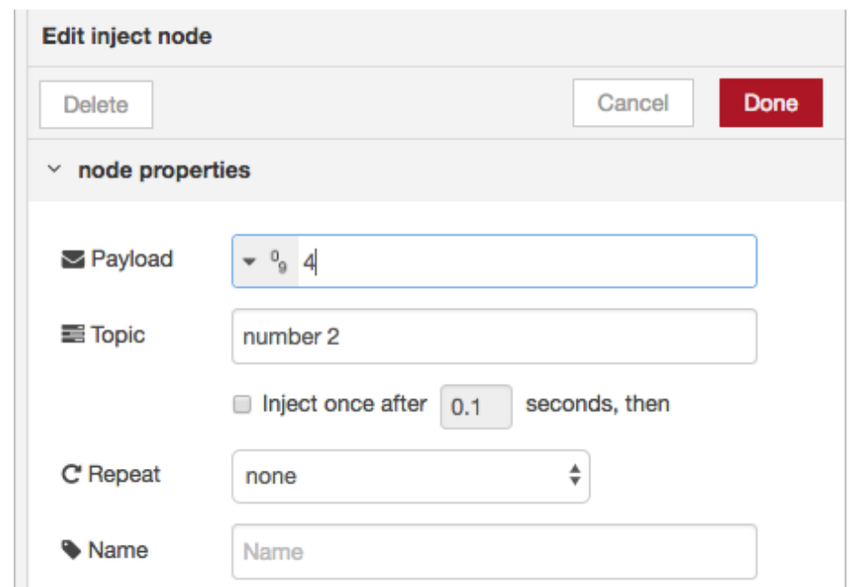


Configure the Inject Nodes

- Inject node 1:
  - Payload: type number; 3
  - Topic: number 1

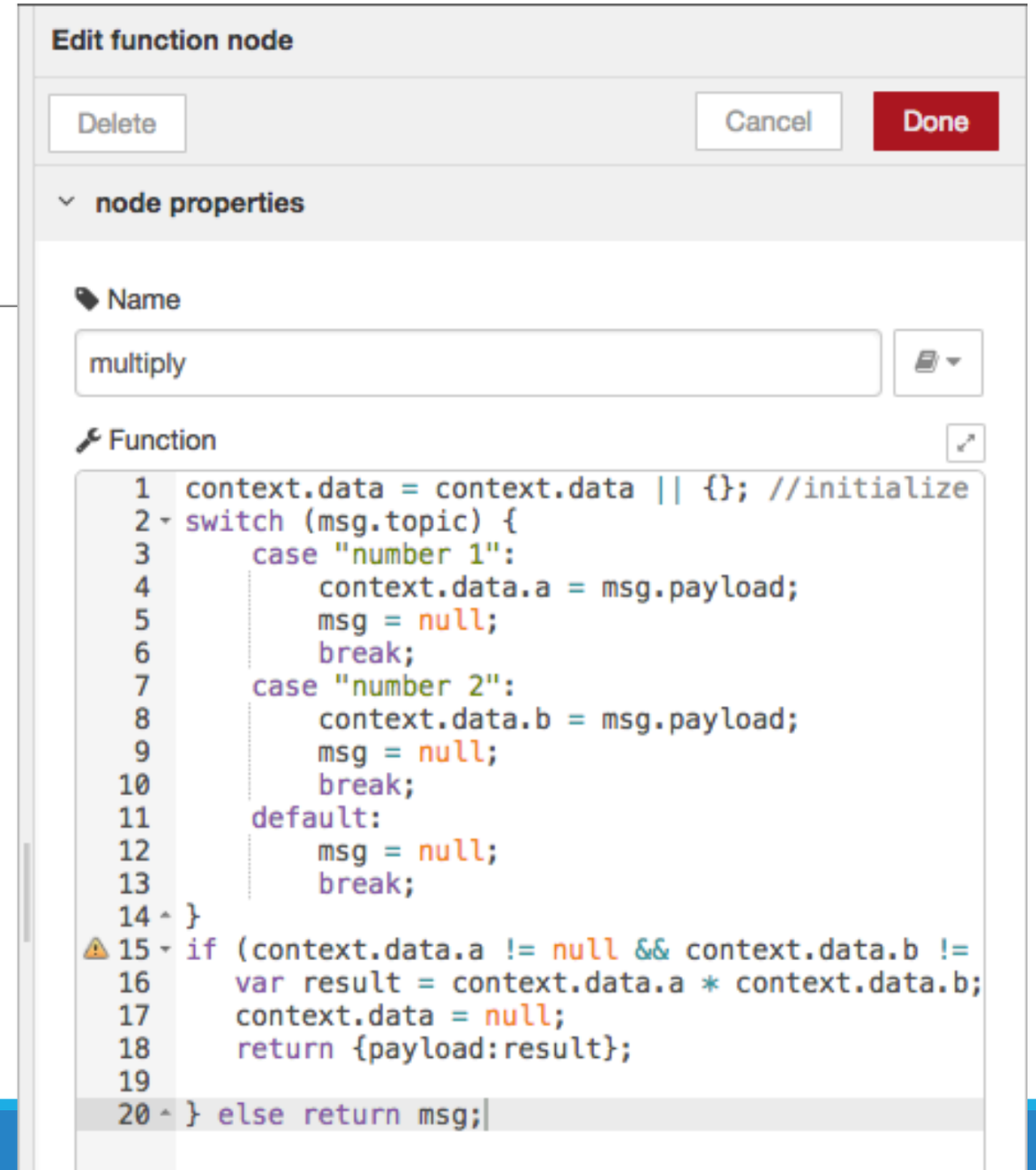


- Inject node 2:
  - Payload: type number; 4
  - Topic: number 2



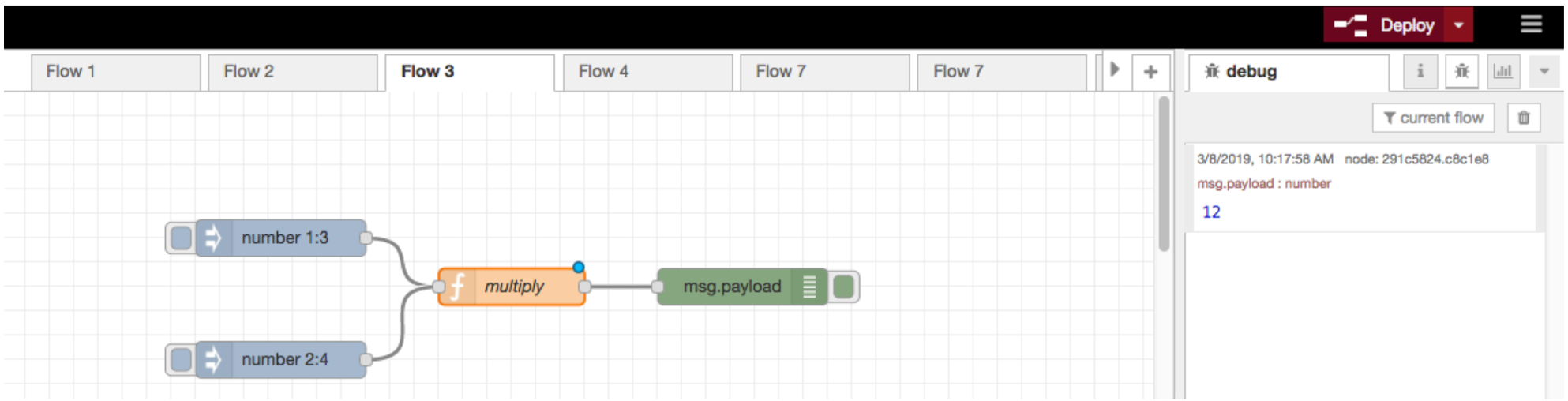
Configure the function Nodes

- This function multiplies two input numbers.



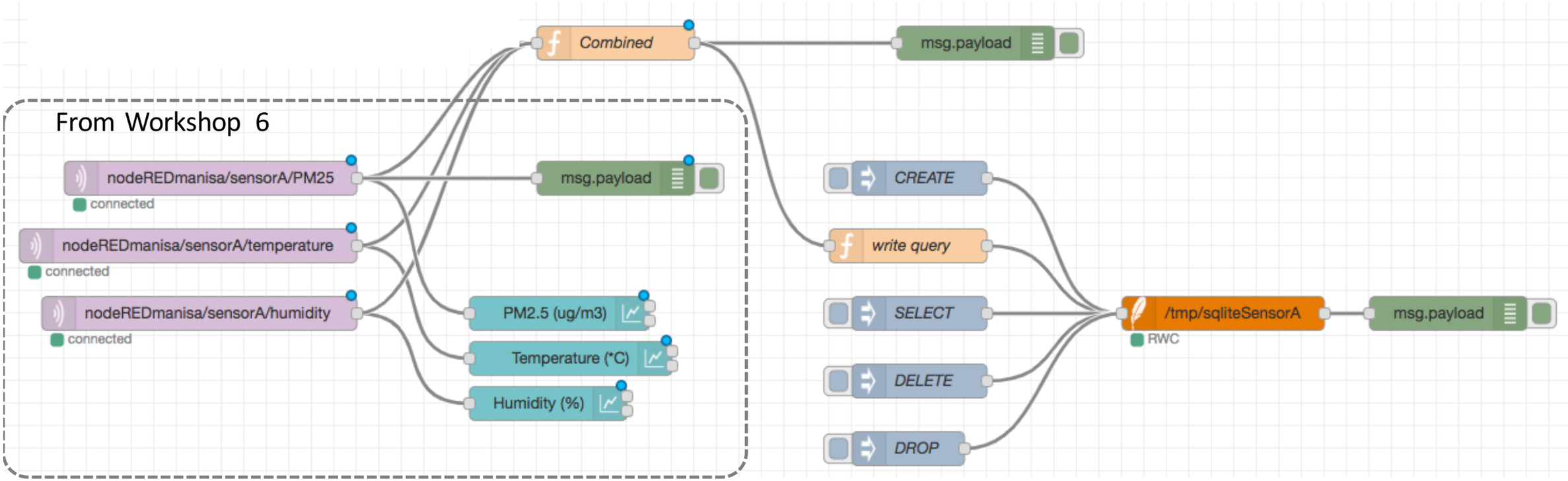
Deploy & Test Your Flow

- Deploy and see if you get 12 as the multiplying result.



Workshop 3. Subscribe to Sensor Data, Display and Save them to Your Database

In Class Activities



Steps

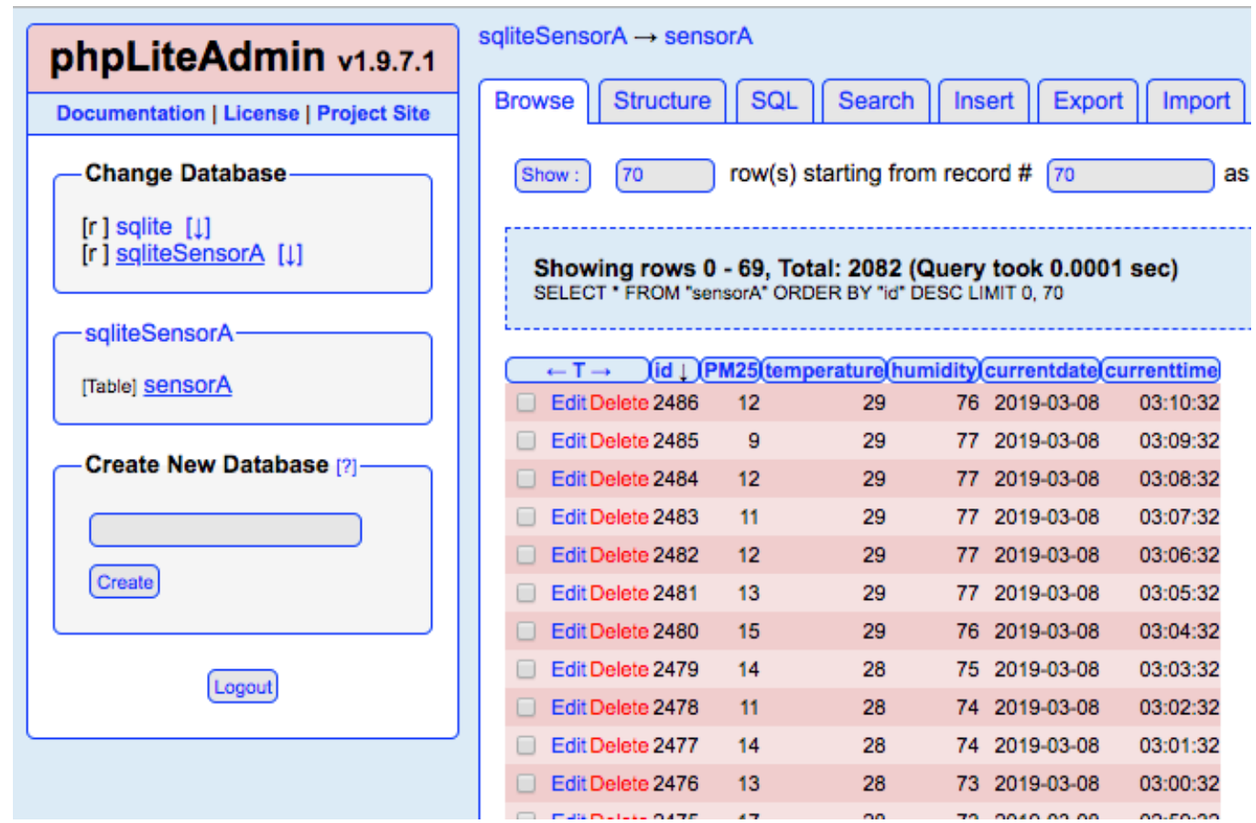
- **[From Workshop 6] Connect your sensor, upload codes to your NodeMCU**
  - Check if this is successful by listening to your group's published messages using MQTT broker
- **[From Workshop 6] Create nodeRED UI**
  - To display temperature/humidity data for your own group
- **Create nodeRED database connection**
  - **Step1:** Listen to your temp/humidity data using MQTT input nodes
  - **Step2:** ["Combined block"] Use the switch/case statement to join multiple data
    Hint: You may need to use "parseFloat" to change msg.payload to float.
  - **Step 3:** Create CREATE, SELECT, DELETE and DROP nodes and connect them with your SQLite node
  - **Step 4:** ["write query" block] Instead of using the INSERT node, you will need another function to create a new message to insert into the database…..with "topic": "INSERT INTO dbname …."

Combined Function Block (Example)

```
var newmsg = {};

var record = 0;

context.data = context.data || {};

switch(msg.topic){
        case "nodeREDmanisa/sensorA/PM25":
                context.data.PM25 = parseInt(msg.payload);
                msg = null;     break;
        case "nodeREDmanisa/sensorA/temperature":
                context.data.temperature = parseFloat(msg.payload);
                msg = null;     break;
        case "nodeREDmanisa/sensorA/humidity":
                context.data.humidity = parseFloat(msg.payload);
                msg = null;     break;
        default:    msg = null;     break;

}

if (context.data.PM25 != null &&  context.data.temperature != null &&  context.data.humidity != null) {
        var time = Date();
        context.data.time = time.toString();
        newmsg = { payload: context.data};
        return newmsg;
} else
        return msg;
```

Write Query Function Block (Example)

```
var newMsg = {
        "topic": "INSERT INTO sensorA(PM25, temperature, humidity, currentdate, currenttime) values(" +
                 msg.payload.PM25 + "," + msg.payload.temperature + "," + msg.payload.humidity +
                 ", date('now'), time('now'))”
 }
 return newMsg;
```

Additional Info

- **Location of your sqlite dB**
  - Search for **/tmp/sqlite** on your computer
  - For mac user, it should be located at the directory **/private/tmp**
- **phpliteadmin**
  - You may use phpliteadmin to manage your dB
  - You will need to install Apache Server
  - Install php
  - Download phpliteadming
  - Configure the database directory to poin to **/private/tmp**