IDA Lab
identification automation

UNIVERSITÀ DEL SALENTO

Hands-on 2 – Node.js

INTERNET OF THINGS – A.A. 2020/2021

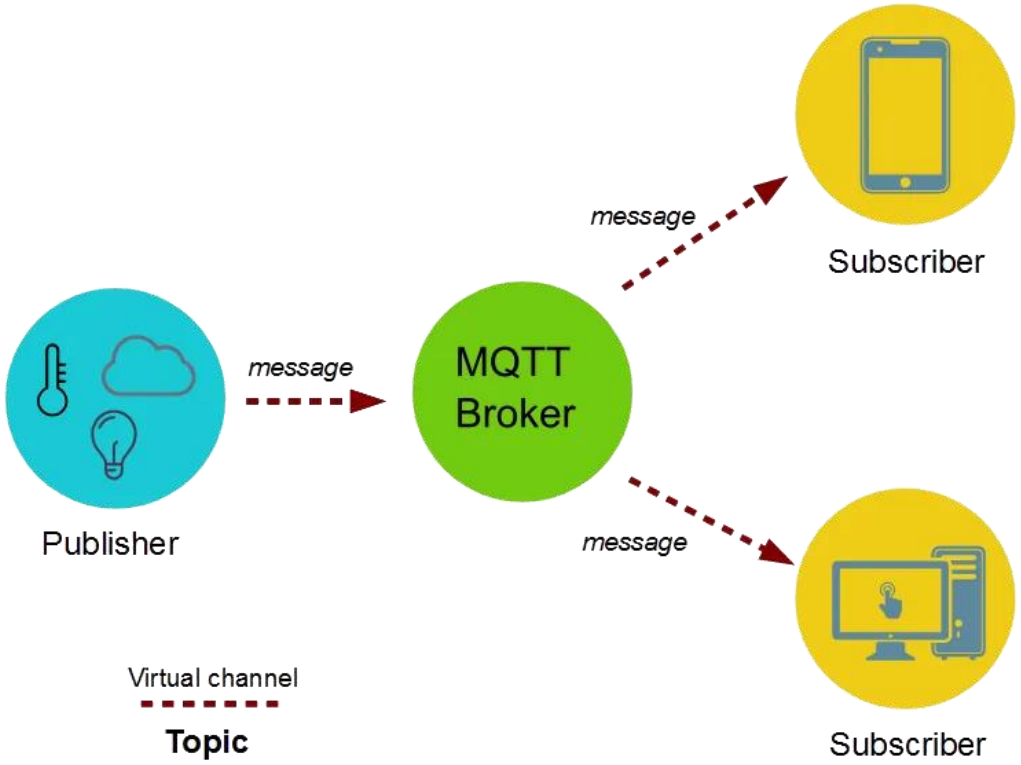
Exercise 2: Let's build a simple WoT project - with MQTT

Step 1: **Requirements**

Starting from the system developed during the last Hands-on

- Modify the architecture to use MQTT protocol instead of REST API

Excursus: MQTT



The Message Queuing Telemetry Transport (MQTT) is a lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP; however, any network protocol that provides ordered, lossless, bi-directional connections can support MQTT.

Source: https://en.wikipedia.org/wiki/MQTT

For more info: wait the theory lesson :D

Excursus: MQTT - topics

Topics

In MQTT, the word topic refers to an UTF-8 string that the broker uses to filter messages for each connected client. The topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator).

In comparison to a message queue, MQTT topics are very lightweight. The client does not need to create the desired topic before they publish or subscribe to it. The broker accepts each valid topic without any prior initialization.

Examples:

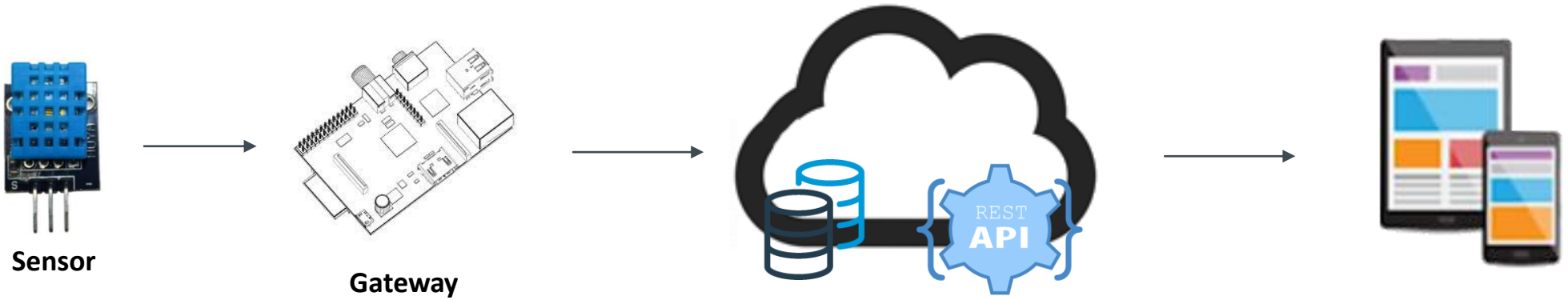
myhome/groundfloor/livingroom/temperature
USA/California/San Francisco/Silicon Valley
5ff4a2ce-e485-40f4-826c-b1a5d81be9b6/status
Germany/Bavaria/car/2382340923453/latitude

More info: https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/

EXERCISE 2

Step 2: Design the System Architecture

Old architecture



Sensor

Gateway
(get data + send data)

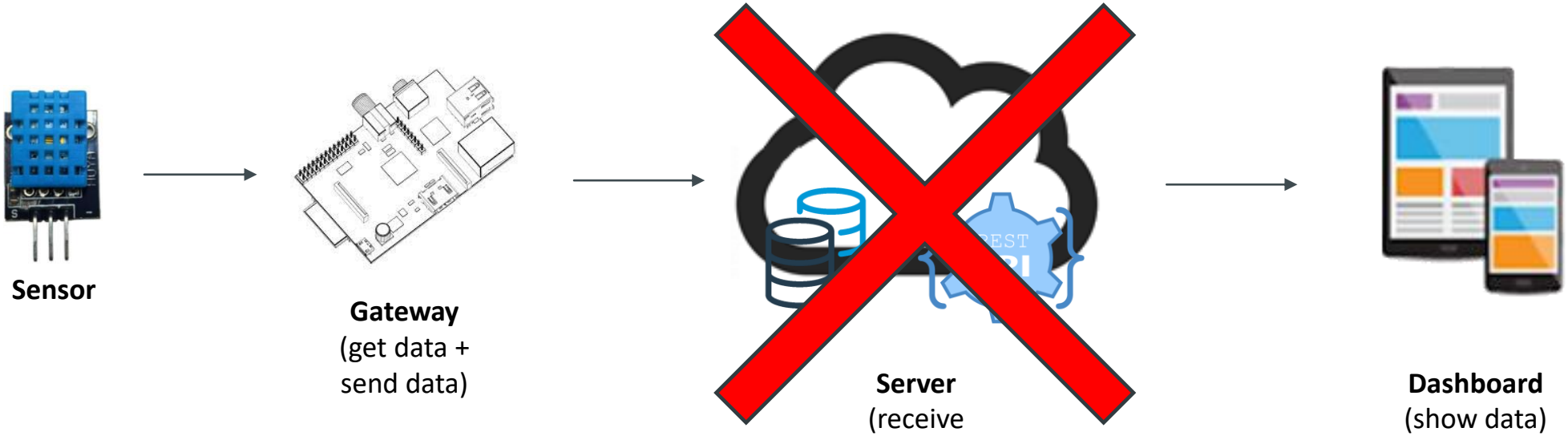
Server
(receive data + save data)

Dashboard
(show data)

EXERCISE 2

Step 2: Design the System Architecture

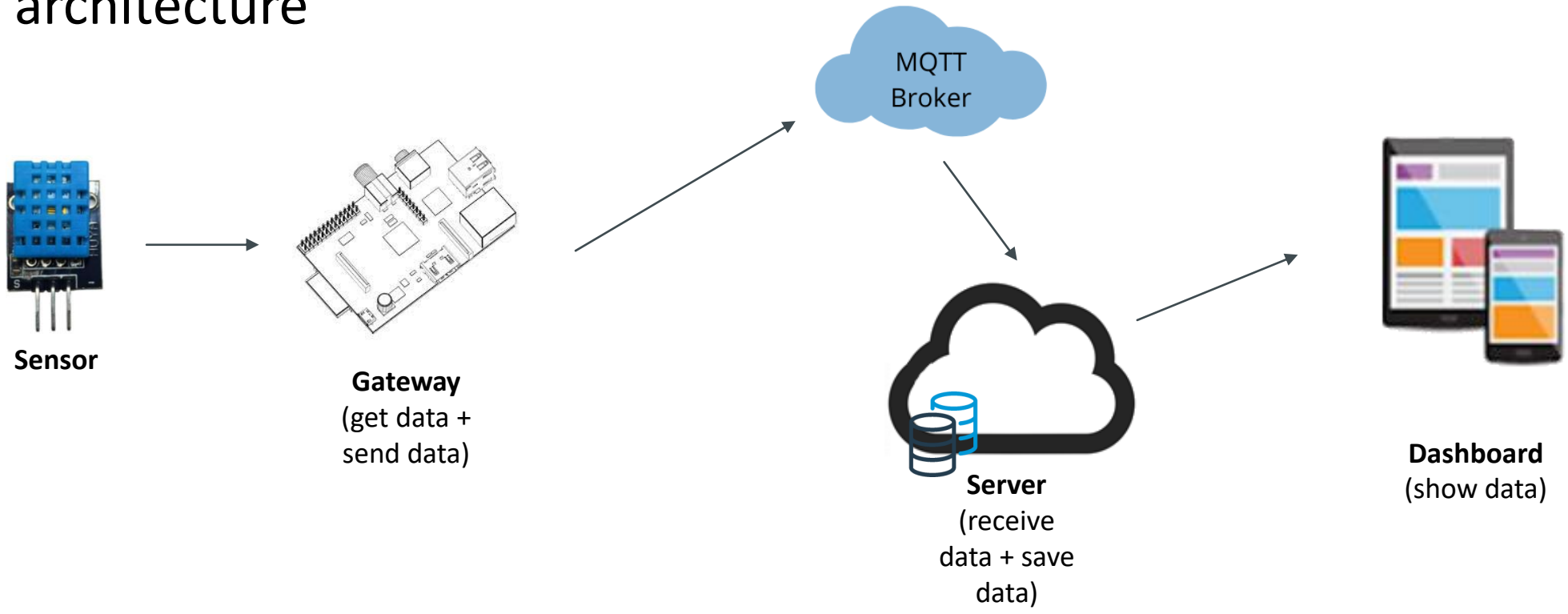
Old architecture



EXERCISE 2

Step 2: Design the System Architecture

New architecture



EXERCISE 2

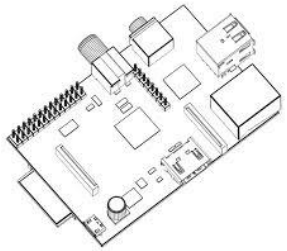
Step 3: choose technologies

Public (& free) MQTT Broker

1. https://www.emqx.io/mqtt/public-mqtt5-broker
2. https://www.hivemq.com/public-mqtt-broker/

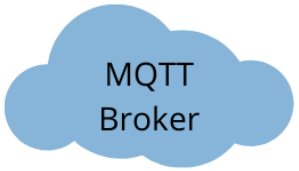
Sensor

DHT11



Gateway:

Raspberry
+
NodeJS



Server
Raspberry
+
NodeJS

Dashboard:

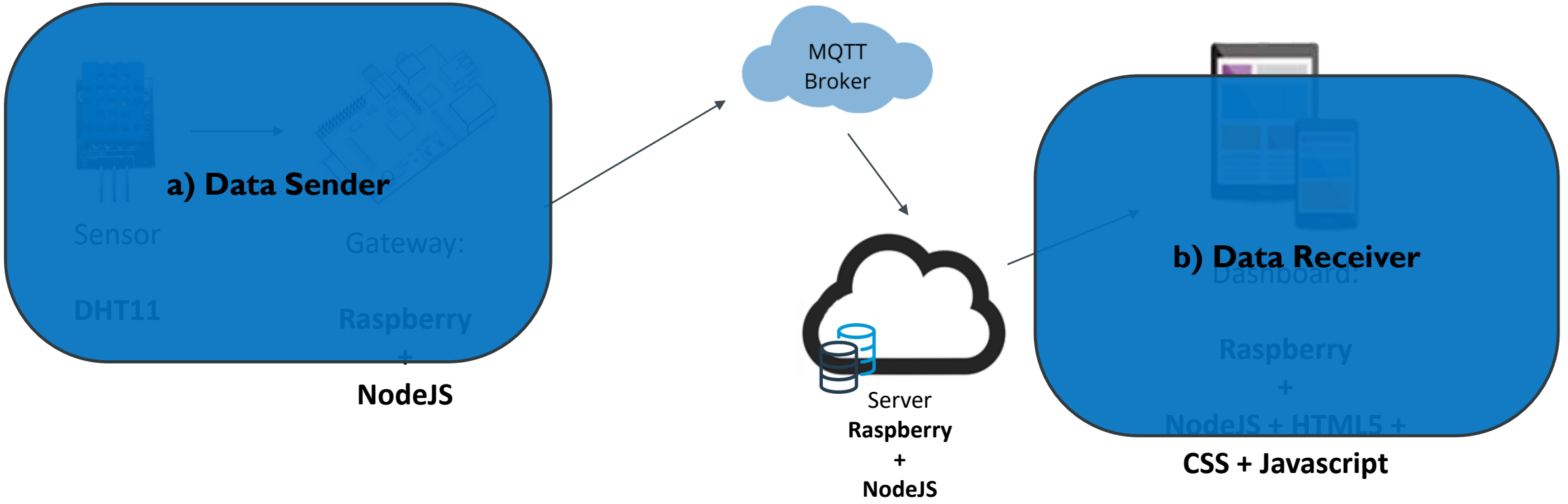
Raspberry
+
NodeJS + HTML5 +
CSS + Javascript

EXERCISE 2

Step 3: choose technologies

Public (& free) MQTT Broker

1. https://www.emqx.io/mqtt/public-mqtt5-broker
2. https://www.hivemq.com/public-mqtt-broker/



EXERCISE 2

Step 4: Developments

a) Data Sender

b) Data Receiver

EXERCISE 2

Step 4: Developments

a) Data Sender

b) Data Receiver

Preliminary exercises

- Fork an existing repository: https://github.com/UniSalento-IDALab-IoTCourse-2020-2021/handson-nodejs-exercise1-A.2
- Create a new repository but push existing code

EXERCISE 2 - **A) DATA SENDER**

Step 4: Developments

a) Data Sender

Assignment: https://classroom.github.com/a/3_tMK3Rx

Preparation

- **Accept the assignment** (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:

https://github.com/UniSalento-IDALab-IoTCourse-2020-2021/handson-nodejs-exercise1-A.2

EXERCISE 2 - **A) DATA SENDER**

Step 4: Developments

a) Data Sender

Assignment: https://classroom.github.com/a/3_tMK3Rx

Preparation

- Accept the assignment (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:
- **Clone the repository on your PC**

```
git clone https://github.com/IDALab-unisalento/exercise2a---datasender---wot-basic-app-with-mqtt-...
```

EXERCISE 2 - **A) DATA SENDER**

Step 4: Developments

a) Data Sender

Assignment: https://classroom.github.com/a/3_tMK3Rx

Preparation

- **Accept the assignment** (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:
- Clone the repository on your PC
- **Create a new branch**

```
git branch developments
git checkout developments
```

EXERCISE 2 - **A) DATA SENDER**

Step 4: Developments

a) Data Sender

Assignment: https://classroom.github.com/a/3_tMK3Rx

Preparation

- **Accept the assignment** (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:
- Clone the repository on your PC
- Create a new branch
- **Push the changes on github**

```
git push -u origin developments
```

EXERCISE 2 - **A) DATA SENDER**

Step 4: Developments

a) Data Sender

Assignment: https://classroom.github.com/a/3_tMK3Rx

Preparation

- **Accept the assignment** (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:
- Clone the repository on your PC
- Create a new branch
- Push the changes on github
- **Develop your code** and **commit+push it on github**

EXERCISE 2 - **A) DATA SENDER - DEVELOP YOUR CODE**

Action 1: Modify the app to send random data (so that you can test it also without a raspberry)

Action 2: Try it through the old server app (run the old server app and connect to http://localhost:3000/dashboard)

firstApp.js

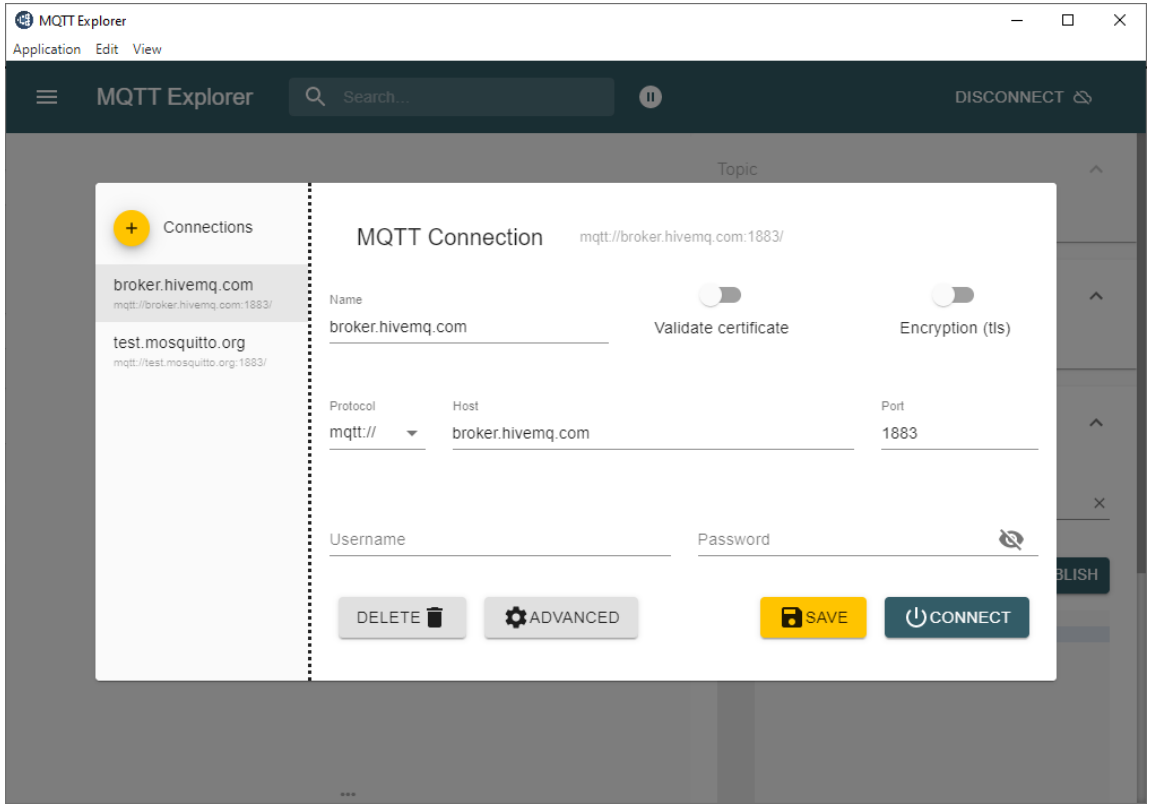
```
const http = require('http')

// Automatically update sensor value every 2 seconds
//we use a nested function (function inside another function)
setInterval(function() {

   const data = JSON.stringify({
      'sensor': 'ID1',
      'timestamp': 12345678,
      'temperature': Math.random()
   })

   const options = {
      hostname: 'localhost',
      port: 3000,
      path: '/temperature',
      method: 'POST',
      headers: {
         'Content-Type': 'application/json',
         'Content-Length': data.length
      }
   }

   const req = http.request(options, res => {
      //define the callback function that will print the result of the
request in case of success
      res.on('data', d => {
         process.stdout.write(d);
      })

      //define the callback function that will print the result of the
request in case of error
      req.on('error', error => {
         console.error(error);
      })
   })
   //send the request
   req.write(data);
   req.end();
}, 2000);
```

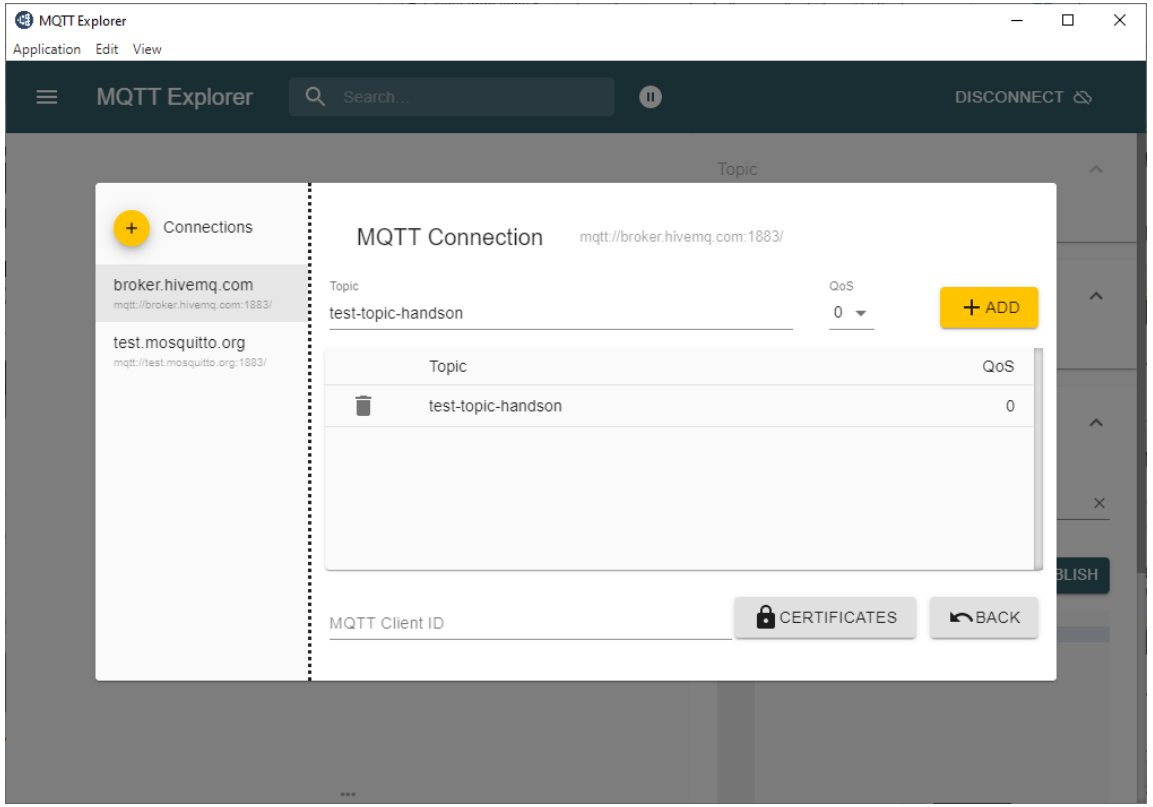
EXERCISE 2 - **A) DATA SENDER - DEVELOP YOUR CODE**

Action 3: Identify the best MQTT broker and download an MQTT client for trials:

MQTT broker: https://www.hivemq.com/public-mqtt-broker/

MQTT client: http://mqtt-explorer.com/





EXERCISE 2 - **A) DATA SENDER - DEVELOP YOUR CODE**

Action 4: Modify the app to send data to the broker

Helping link:
http://www.steves-internet-guide.com/using-node-mqtt-client/

firstApp.js

```
const mqtt=require('mqtt');

var client =
mqtt.connect("mqtt://broker.hivemq.com",{clientId:"mqttjs01"});
client.on("connect",function(){
   console.log("connected");
});
client.on("error",function(error){
   console.log("Can't connect"+error);
});

// Automatically update sensor value every 2 seconds
//we use a nested function (function inside another function)
setInterval(function() {

   const data = JSON.stringify({
      'sensor': 'ID1',
      'timestamp': 12345678,
      'temperature': Math.random()
   })

   const options = {

      hostname: '10.0.13.50',
      port: 3000,
      path: '/temperature',
      method: 'POST',
      headers: {
         'Content-Type': 'application/json',
         'Content-Length': data.length
      }
   }

client.publish("test-topic-handson/teo", data);
}, 2000);
```

EXERCISE 2 - **A) DATA SENDER**

Step 4: Developments

a) Data Sender

Assignment: https://classroom.github.com/a/3_tMK3Rx

Preparation

- **Accept the assignment** (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:
- Clone the repository on your PC
- Create a new branch
- Push the changes on github
- Develop your code and commit+push it on github
- Merge the branch with the master branch

```
git checkout master
git merge developments
```

EXERCISE 2 - **B) DATA RECEIVER**

Step 4: Developments

b) Data Receiver

Assignment: https://classroom.github.com/a/BXeyY8hn

Preparation

- **Accept the assignment** (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:

https://github.com/UniSalento-IDALab-IoTCourse-2020-2021/handson-nodejs-exercise1-C.3

EXERCISE 2 - **B) DATA RECEIVER**

Step 4: Developments

b) Data Receiver

Assignment: https://classroom.github.com/a/BXeyY8hn

Preparation

- Accept the assignment (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:
- **Clone the repository on your PC**

```
git clone https://github.com/IDALab-unisalento/exercise2b---datareceiver---wot-basic-app-with-mqtt-...
```

EXERCISE 2 - **B) DATA RECEIVER**

Step 4: Developments

b) Data Receiver

Assignment: https://classroom.github.com/a/BXeyY8hn

Preparation

- Accept the assignment (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:
- Clone the repository on your PC
- **Create a new branch**

```
git branch developments
git checkout developments
```

EXERCISE 2 - **B) DATA RECEIVER**

Step 4: Developments

b) Data Receiver

Assignment: https://classroom.github.com/a/BXeyY8hn

Preparation

- Accept the assignment (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:
- Clone the repository on your PC
- Create a new branch
- **Push the changes on github**

```
git push -u origin developments
```

EXERCISE 2 - **B) DATA RECEIVER**

Step 4: Developments

b) Data Receiver

Assignment: https://classroom.github.com/a/BXeyY8hn

Preparation

- Accept the assignment (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:
- Clone the repository on your PC
- Create a new branch
- Push the changes on github
- **Develop your code and commit+push it on github**

EXERCISE 2 - **B) DATA RECEIVER - DEVELOP YOUR CODE**

Action 1: Modify the app to:

- Remove the database interactions

serverApp.js

```
const express = require("express");
const WebSocket = require('ws');
const path = require('path');

const app = express();

const wss = new WebSocket.Server({ port: 3001 });
wss.on('connection', function connection(ws) {
   ws.on('message', function incoming(message) {
      console.log('received: %s', message);
   });
   ws.send('something');
});

app.listen(3000, () => {
   console.log("Server running on port 3000");
});
app.use(
   express.urlencoded({
      extended: true
   })
)
app.use(express.json());
app.post("/temperature", (req, res, next) => {
   console.log(req.body.temperature);
   var temperature = req.body.temperature;
   var timestamp = req.body.timestamp;
   var sensor = req.body.sensor;

   async function pushToClient(){
      wss.clients.forEach(function each(client) {
         if (client.readyState === WebSocket.OPEN) {
            client.send(temperature);
         }
      });
   }
   pushToClient().catch(console.dir);
   res.sendStatus(200)
});

app.get('/dashboard', async (req, res) => {
   res.sendFile(path.join(__dirname + '/index.html'));
})
```

EXERCISE 2 - **B) DATA RECEIVER - DEVELOP YOUR CODE**

Action 2: Modify the app to:

- Introduce MQTT: instead of waiting data from a POST request we will wait an MQTT message
- Subscribe to the topic "test-topic-handson"

Helping link:
http://www.steves-internet-guide.com/using-node-mqtt-client/

EXERCISE 2 - **B) DATA RECEIVER - DEVELOP YOUR CODE**

serverApp.js

```
const express = require("express");
const WebSocket = require('ws');
const path = require('path');
const mqtt=require('mqtt');

const app = express();

const wss = new WebSocket.Server({ port: 3001 });

wss.on('connection', function connection(ws) {
   ws.on('message', function incoming(message) {
      console.log('received: %s', message);
   });
   ws.send('something');
});

var mqttClient =
mqtt.connect("mqtt://broker.hivemq.com",{clie
ntId:"mqttjs041"});
mqttClient.on("connect",function(){
   console.log("connected");
});
mqttClient.on("error",function(error){
   console.log("Can't connect"+error);
});

mqttClient.on('message',function(topic, message,
packet){
   console.log("message is "+ message);
   console.log("topic is "+ topic);
   var messageJSON = JSON.parse(message);
   var temperature = messageJSON.temperature;
   var timestamp = messageJSON.timestamp;
   var sensor = messageJSON.sensor;

   async function pushToClient(){
      wss.clients.forEach(function each(client) {
         if (client.readyState === WebSocket.OPEN)
{
            client.send(temperature);
         }
      });
   }
   pushToClient().catch(console.dir);
});
app.listen(3000, () => {
   console.log("Server running on port 3000");
});
app.use(
   express.urlencoded({
      extended: true
   })
)

app.use(express.json());

app.get('/dashboard', async (req, res) => {
   res.sendFile(path.join(__dirname +
'/index.html'));
})


var topic="test-topic-handson/teo";
console.log("subscribing to topic"+topic);
mqttClient.subscribe(topic); //single topic
```

EXERCISE 2 - **B) DATA RECEIVER**

Step 4: Developments

b) Data Receiver

Assignment: https://classroom.github.com/a/BXeyY8hn

Preparation

- **Accept the assignment** (connect to the link, click accept and wait a while, then refresh the page)
- The new repo will be already a fork of:
- Clone the repository on your PC
- Create a new branch
- Push the changes on github
- Develop your code and commit+push it on github
- **Merge the branch with the master branch**

```
git checkout master
git merge developments
```

FURTHER IMPROVEMENTS

Further improvements -> Homework

- Enhance the Server to also save data into the database when new data are received through MQTT

USEFUL LINKS

Questions?

NO CODE IN EMAILS,

PLEASE

DEAR ENG. MONTANARO,

I HAVE SOME TROUBLES WITH THE CODE THAT YOU CAN FIND ON GITHUB AT THE FOLLOWING LINK:

HTTPS://GITHUB.COM/GITHUBTRAINING/HELLOGITWORLD

YOU CAN FIND THE DETAILS IN THE ISSUE N. 3.

THANK YOU IN ADVANCE

JAMES

COURSE: INTERNET OF THINGS

Prof. Luigi Patrono, Ph.D.
luigi.patrono@unisalento.it

Eng. Teodoro Montanaro, Ph.D.
teodoro.montanaro@unisalento.it

Eng. Ilaria Sergi, Ph.D.
ilaria.sergi@unisalento.it