

## Watson IoT Platform – T3 - NodeRed – Device – connection

### Käyttöönotto-ohje 3

#### 1. Johdanto

Tässä käyttöönotto-ohjeessä luomme yhteyden todellisen laitteen ja IBM Watson NodeRED – palvelimen välille.

Teemme myös web-selaimella toimivan käyttöliittymän.

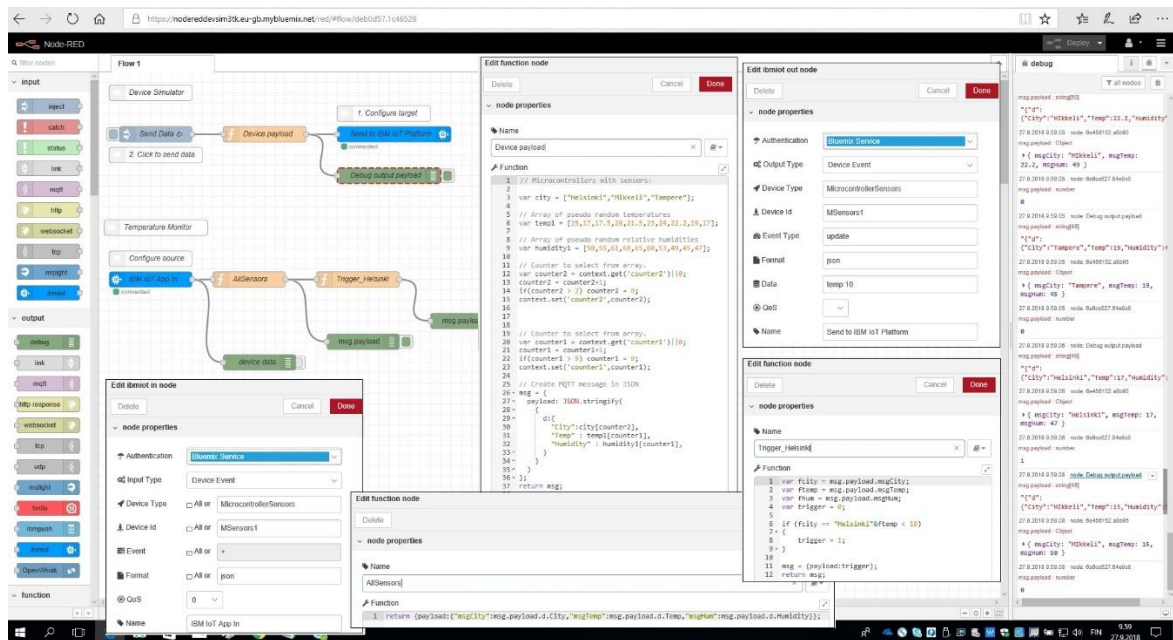
Käyttöönotto-ohjeessä T0 loit itsellesi käyttäjätunnuksen IBM Bluemix ympäristöön, perustit laitteen ja kirjoitit laitedataa joko MQTT-sovelluksella tai mikro-ohjainlaitteella.

Jos et jo tehnyt näitä asioita, katso ohjeet IBM Cloud –dokumentaatiosta. Tämän ohjeen uusimpaa versiota tallentaessani oli löydettävissä reitti Watson IoT -alustan käyttöönotto-ohjeeseen IBM Cloud -sivulla

<https://www.ibm.com/cloud/get-started/iot-platform>

Jos sinulla on noin tunti aikaa ja olet valmis rekisteröitymään käyttäjäksi, valitse "Connect a networked device to the IBM Watson IoT Platform". Valitse edelleen "Play with IBM Watson IoT Platform".

Käyttöönotto-ohjeessä T2 käsitelimme laitedataa Watson IoT-alustalla. Laitetietojen käsittelyä varten loimme NodeRED -editorilla sovelluksen. Sovellus tallentui IBM Cloud -palveluun CloudantNoSQLDB -tietokantaan.



Kuva 1.1 Käyttöönottotehtävän T2 NodeRED –flow ja kuhunkin node-lohkoon kirjatut sisällöt.

Kuvassa näkyvä laitedataa simulointina luova flow-kaavio ja laitedataa laitteelta lukeva flow-kaavio on nopea kirjoittaa, jos et niitä ole aiemmin tehnyt.

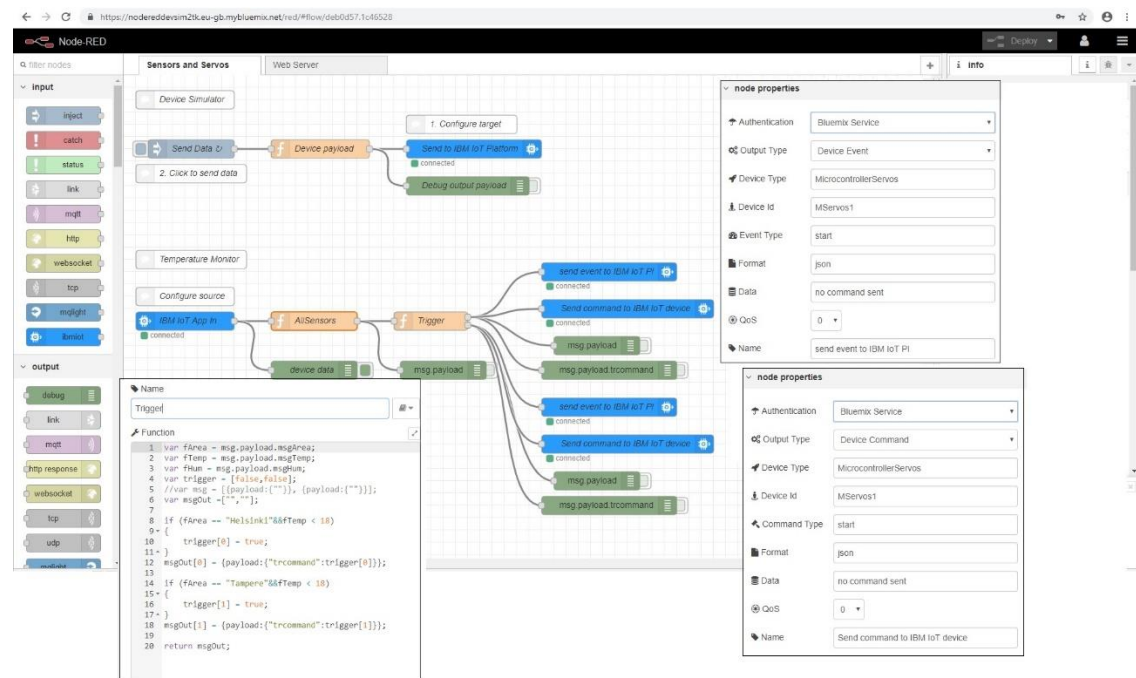
Juuri sama laitetyyppi ja laitteet on luotava Watson IoT –alustaan juuri samalle organisaatiolle.

The screenshot shows the 'Browse Devices' page in the IBM Watson IoT Platform. The page has a sidebar with navigation icons and a main content area. The 'Browse Devices' section is active, showing a table of devices. The table has columns for 'Device ID', 'Device Type', 'Class ID', and 'Date Added'. There are three devices listed: 'MSensors1', 'MServos1', and 'MServos2'. The 'Add Device' button is visible in the top right corner.

Device ID	Device Type	Class ID	Date Added
MSensors1	MicrocontrollerSensors	Device	27 Sep 2018 14:36
MServos1	MicrocontrollerServos	Device	27 Sep 2018 14:43
MServos2	MicrocontrollerServos	Device	28 Sep 2018 09:57

Kuva 1.2 Käyttöönottotehtävän T2 laitteet Devices – laitenaikymässä.

Laitteelle – oikealle tai simuloidulle – voidaan kirjoittaa seuraavan kuvan mukaisesti.



Kuva 1.3 Käyttöönottotehtävän T2 laitteelle kirjoitus.

Edellä kuvatun floun:n voit siirtää omaan Watson NodeRED:iin tuomalla sisällön Import Clipboard –toiminnolla. Kopioi seuraava ”koodi” leikepöydälle. Siirrä se windows notepad-editoriin poistaaksesi ylimääräiset näkymättömät merkit. Kopioi se siitä NodeRED:iin. Näin ei tarvitse kirjoittaa koodia.

```

{"id":"deb0d57.1c46528","type":"","tab":"","label":"Sensors and Servos","disabled":false,"info":"","","id":"3e77d543.c1882a","type":"","ibmiot
in","z":"deb0d57.1c46528","authentication":{"boundService":"","apiKey":"","inputType":"evt","logicalInterface":"","ruleId":"","deviceId":"MSensors1","ap
plicationId":"","deviceType":"MicrocontrollerSensors","eventType":"+","commandType":"","format":"json","name":"IBM IoT App
In","service":"","registered":"","allDevices":false,"allApplications":false,"allDeviceTypes":false,"allLogicalInterfaces":false,"allEvents":true,"allCommands":fal
se,"allFormats":false,"qos":"","o":"","x":"100","y":"400","wires":[["ae0082ac.51ffb","c0c482df.3f3b8"]]},{"id":"ae0082ac.51ffb","type":"function","z":"deb0d57.1c
46528","name":"AllSensors","func":"return
{payload:{\"msgArea\":\"msg.payload.d.Area\",\"msgTemp\":\"msg.payload.d.Temp\",\"msgHum\":\"msg.payload.d.Humidity\"}};\",\"outputs\":1,\"noerr\":0,\"x\":310
,\"y\":400,\"wires":[[\"54b28de8.4a9c34\",\"82c0704.645f29\"]]},{"id":"c0c482df.3f3b8","type":"debug","z":"deb0d57.1c46528","name":"device
data\",\"active\":true,\"console\":\"false\",\"complete\":\"true\",\"x\":310,\"y\":480,\"wires":[{}]},{"id":"cbe1a0b7.cd877","type":"ibmiot
out","z":"deb0d57.1c46528","authentication":{"boundService":"","apiKey":"","outputType":"","format":"","name":"Device payload","deviceType":"MicrocontrollerSe
nsors","eventCommandType":"update","format":"json","data":"temp:10","qos":"","name":"Send to IBM IoT
Platform","service":"","registered":"","x":570,\"y\":120,\"wires":[{}]},{"id":"5917a925.6a3c08","type":"inject","z":"deb0d57.1c46528","name":"Send
Data","topic":"","payload":"true","payloadType":"bool","repeat":"60","crontab":"","","once":false,"onceDelay":"","","x":110,\"y\":120,\"wires":[[\"6b4a591c.01
4b18\"]]},{"id":"6b4a591c.014b18","type":"function","z":"deb0d57.1c46528","name":"Microcontrollers with sensors:\\n\\nvar
area1 = [\\nGreenhouse1\\\",\\nGreenhouse2\\\",\\nGreenhouse3\\\"]\\n\\n// Array of pseudo random temperatures\\nvar temp1 =
[15,17,17.5,20,21,23,24,22.2,19,17];\\n\\n// Array of pseudo random relative humidities\\nvar humidity1 = [50,55,61,68,65,60,53,49,45,47];\\n\\n//
Counter to select from array.\\nvar counter1 = context.get('counter1') || 0;\\ncounter1 = counter1+1;\\nif(counter1 > 9) counter1 =
0;\\ncontext.set('counter1',counter1);\\n\\n// Counter to select from array.\\nvar counter2 = context.get('counter2') || 0;\\ncounter2 =
counter2+1;\\nif(counter2 > 2) counter2 = 0;\\ncontext.set('counter2',counter2);\\n\\n// Create MQTT message in JSON\\nmsg = {\\n payload:
JSON.stringify(\\n {\\n d:{\\n \\Area\\\":area1[counter2],\\n \\Temp\\\": temp1[counter1],\\n \\Humidity\\\": humidity1[counter1],\\n }\\n
}\\n )\\n);\\nreturn
msg;\\n\",\"outputs\":1,\"noerr\":0,\"x\":320,\"y\":120,\"wires":[[\"cbe1a0b7.cd877\",\"805c97ee.3ed9e8\"]]},{"id":"805c97ee.3ed9e8","type":"debug","z":"deb0d5
7.1c46528","name":"Debug output
payload\",\"active\":false,\"tosidebar\":true,\"console\":false,\"complete\":\"payload\",\"x\":560,\"y\":180,\"wires":[{}]},{"id":"86df0fb6c.af90c8","type":"comment","z
":"deb0d57.1c46528","name":"Device Simulator","info":"Sends simulated device sensor data to IBM Watson IoT Plaform.\\n\\nCan be configured to send
on click or on an automatic interval.\\n\\n#Prerequisite\\nOutput node device type and device ID need to match a device that it registered in a running
IBM Watson IoT Platform service.\\n\\n# Watson IoT Platform docs\\n[Connecting
devices](https://www.ibm.com/docs/services/iot/iotplatform_task.html)\"},{"id":"141b7c7c.ad42a84","type":"comment","z
":"deb0d57.1c46528","name":"1. Configure
target\",\"info":"","","x":550,\"y\":80,\"wires":[{}]},{"id":"c2dd8ed5.7dd7f","type":"comment","z":"deb0d57.1c46528","name":"2. Click to send data\",\"info\":\"To
automatically send data:\\n1. Change *Repeat* to interval.\\n2. Click Deploy
button.\\n\",\"x\":110,\"y\":160,\"wires":[{}]},{"id":"7926c7b2.86d938","type":"comment","z":"deb0d57.1c46528","name":"Temperature
Monitor\",\"info":"","","x\":110,\"y\":300,\"wires":[{}]},{"id":"188a5e87.e775a1","type":"comment","z":"deb0d57.1c46528","name":"Configure
source\",\"info":"","","x":100,\"y\":360,\"wires":[{}]},{"id":"54b28de8.4a9c34","type":"function","z":"deb0d57.1c46528","name":"Trigger","func":"var fArea =
msg.payload.msgArea;\\nvar fTemp = msg.payload.msgTemp;\\nvar fHum = msg.payload.msgHum;\\nvar trigger = [false,false];\\n\\nvar msg =
[{payload:{\"\\\"\\\"}, {payload:{\"\\\"\\\"}};\\nvar msgOut = [\"\\\"\\\"\\\"\\\"];\\n\\nif (fArea == \"\\\"\\\"\\\"\\\"&fTemp < 18)\\n{\\n trigger[0] = true;\\n}\\nmsgOut[0] =
{payload:{\"trcommand\":\"trigger[0]\"}};\\n\\nif (fArea == \"\\\"\\\"\\\"\\\"&fTemp < 18)\\n{\\n trigger[1] = true;\\n}\\nmsgOut[1] =

```

```
{payload:{\"trcommand\":trigger[1]}};\n\nreturn\nmsgOut,\"outputs\":2,\"noerr\":0,\"x\":500,\"y\":400,\"wires\":[[\"86eabffb.b92be\", \"8e9ae4a8.9edf88\", \"476d79d6.e80738\", \"a84195d8.b33de8\"],[\"9e624f2a.1e153\", \"21af5407.97aeac\", \"c4c0d1.71c2af3\", \"351079b3.3dfe66\"]]],{\"id\": \"82c0704.645f29\", \"type\": \"debug\", \"z\": \"deb0d57.1c46528\", \"name\": \"\", \"active\": false, \"tosidebar\": true, \"console\": false, \"tostatus\": false, \"complete\": \"false\", \"x\": 520, \"y\": 480, \"wires\": []}, {\"id\": \"86eabffb.b92be\", \"type\": \"debug\", \"z\": \"deb0d57.1c46528\", \"name\": \"\", \"active\": false, \"tosidebar\": true, \"console\": false, \"tostatus\": false, \"complete\": \"false\", \"x\": 770, \"y\": 440, \"wires\": []}, {\"id\": \"8e9ae4a8.9edf88\", \"type\": \"ibmiot\nout\", \"z\": \"deb0d57.1c46528\", \"authentication\": \"boundService\", \"apiKey\": \"\", \"outputType\": \"evt\", \"deviceId\": \"MServos1\", \"deviceType\": \"MicrocontrollerServos\", \"eventCommandType\": \"start\", \"format\": \"json\", \"data\": \"no command sent\", \"qos\": 0, \"name\": \"send event to IBM IoT\nPI\", \"service\": \"registered\", \"x\": 790, \"y\": 320, \"wires\": []}, {\"id\": \"476d79d6.e80738\", \"type\": \"debug\", \"z\": \"deb0d57.1c46528\", \"name\": \"\", \"active\": false, \"tosidebar\": false, \"console\": false, \"tostatus\": false, \"complete\": \"payload.trcommand\", \"x\": 790, \"y\": 480, \"wires\": []}, {\"id\": \"9e624f2a.1e153\", \"type\": \"ibmiot\nout\", \"z\": \"deb0d57.1c46528\", \"authentication\": \"boundService\", \"apiKey\": \"\", \"outputType\": \"evt\", \"deviceId\": \"MServos2\", \"deviceType\": \"MicrocontrollerServos\", \"eventCommandType\": \"start\", \"format\": \"json\", \"data\": \"no command sent\", \"qos\": 0, \"name\": \"send event to IBM IoT\nPI\", \"service\": \"registered\", \"x\": 790, \"y\": 540, \"wires\": []}, {\"id\": \"21af5407.97aeac\", \"type\": \"debug\", \"z\": \"deb0d57.1c46528\", \"name\": \"\", \"active\": false, \"tosidebar\": true, \"console\": false, \"tostatus\": false, \"complete\": \"false\", \"x\": 770, \"y\": 660, \"wires\": []}, {\"id\": \"c4c0d1.71c2af3\", \"type\": \"debug\", \"z\": \"deb0d57.1c46528\", \"name\": \"\", \"active\": false, \"tosidebar\": true, \"console\": false, \"tostatus\": false, \"complete\": \"payload.trcommand\", \"x\": 790, \"y\": 700, \"wires\": []}, {\"id\": \"a84195d8.b33de8\", \"type\": \"ibmiot\nout\", \"z\": \"deb0d57.1c46528\", \"authentication\": \"boundService\", \"apiKey\": \"\", \"outputType\": \"cmd\", \"deviceId\": \"MServos1\", \"deviceType\": \"MicrocontrollerServos\", \"eventCommandType\": \"start\", \"format\": \"json\", \"data\": \"no command sent\", \"qos\": 0, \"name\": \"Send command to IBM IoT\ndevice\", \"service\": \"registered\", \"x\": 820, \"y\": 380, \"wires\": []}, {\"id\": \"351079b3.3dfe66\", \"type\": \"ibmiot\nout\", \"z\": \"deb0d57.1c46528\", \"authentication\": \"boundService\", \"apiKey\": \"\", \"outputType\": \"cmd\", \"deviceId\": \"MServos2\", \"deviceType\": \"MicrocontrollerServos\", \"eventCommandType\": \"start\", \"format\": \"json\", \"data\": \"no command sent\", \"qos\": 0, \"name\": \"Send command to IBM IoT\ndevice\", \"service\": \"registered\", \"x\": 820, \"y\": 600, \"wires\": []}]}
```

Koodi on myös tiedostossa

IBMWatsonSignalsFunctionsServo12\_flow\_va.txt

IBM Cloud -tilisi ominaisuuksista ym. johtuen sinulla ei ehkä ole valittavassa NodeRED IBM IoT -nodessa autentikointimenetelmää ”Bluemix Service”. Voit tällöin valita autentikointimenetelmäksi ”API Key”. Käy luomassa Watson IoT Platform:n puolella API Key ja syötä tiedot vastaavasti tähän IBM IoT -nodeen.

## 2. Yhteys laitteen ja Watson IoT-alustan välillä.

Luo kuvassa 1.2 näkyvään laitenäkymään – tai itse asiassa kuvan kaltaiseen laitenäkymään omassa Watson IoT – Application Instanssissa – todellinen laite. Laitteelle luodaan ensin

Device Type – laitetyyppi, esim. M\_Sensors

Device ID – laite, esim. M\_Sensor\_01

ja annetaan järjestelmän luoda kirjautumistiedot. Tallenna organization ID .....Authentication Token ... kaikki syntyvät kirjautumistiedot myöhempää käyttöä varten tekstitiedostoon.

Laitteena voidaan käyttää lähes mitä hyvänsä laitetta, joka pystyy lähettämään Internet:iin http-liikenteenä MQTT-protokollan mukaisia sanomia.

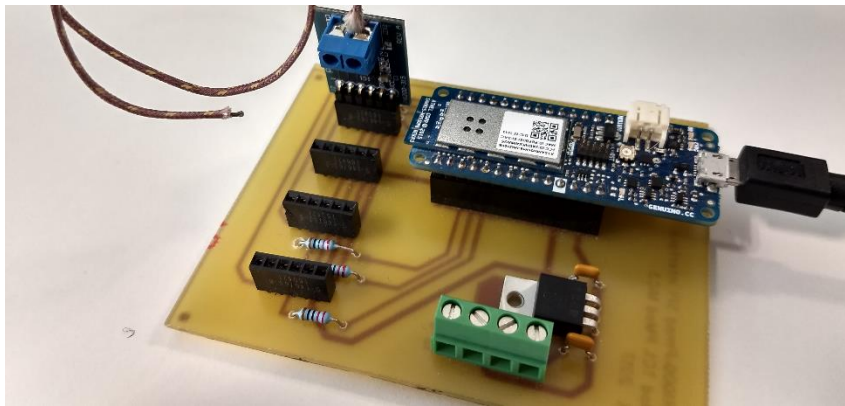
Ohjeessa IoT\_IBMWatson\_T0\_MKR1000\_Aloitus\_GettingStarted\_FI.pdf” kerrotaan yksityiskohtaisesti, miten laitteen kirjautumistiedot luodaan ja miten laite liitetään Watson IoT – alustaan.

Ohjeen ” IoT\_IBMWatson\_T1\_mqtt\_client\_FI.pdf” lopussa näytetään, miten Windows-tietokoneeseen asennettua MQTT Client –sovellusta MQTTBox käytetään ”laitteena”. Tällöin laitteelle voi luoda esim. seuraavanlaisen kirjautumisteidon

Device Type – laitetyyppi, esim. MQTTClient\_Sensors

Device ID – laite, esim. MQTTclient\_Sensor\_01

Tässä esimerkissä käytämme laitteena Arduino MKR1000 mikro-ohjainta ja jotakin siihen jo aiemmin liitettyä anturia.



Kuva 2.1. Arduino MKR1000 ja Digilent pmodTC1 –anturikortti.

Arduinossa tarvittavan ohjelman lähtökohtana voi käyttää kuvassa näkyvän anturin lämpötilatiedon siirtoon käytettyä ohjelmaa.

---

```
/*
MKR1000 connecting to IBM Watson IoT Platform

Based on documentation and "recipes" on IBM Bluemix
https://www.ibm.com/cloud-computing/bluemix/watson
Timo Karppinen 19.2.2017

Modified for testing SPI thermocouple board Digilent PmodTC1
Please connect
MKR1000 - PmodTC1
GND - 5 GND
Vcc - 6 Vcc
9 SCK - 4 SCK
10 MISO - 3 MISO
2 - 1 SS

Thermocouple data on the SPI
D31 - sign
D30 ....D18 - 13 bits of temperature data
D16 - normally FALSE. TRUE if thermocouple input is open or shorted to GND or VCC
D15 ... D0 - reference junction temperature

The reference junction compensation is calculated in the IC. no need to calculate here.

Timo Karppinen 25.1.2018
```

```

*/

#include <SPI.h>
#include <WiFi101.h>
#include <WiFiSSLClient.h>
#include <MQTTClient.h>

// WLAN
//char ssid[] = "Moto_Z2_TK"; // your network SSID (name)
//char pass[] = "xxxxxxxxxx"; // your network password (use for WPA)

char ssid[] = "HAMKVisitor"; // your network SSID (name)
char pass[] = "xxxxxxxxxx"; // your network password (use for WPA)

//char ssid[] = "Nelli";
//char pass[] = "xxxxxxxxxx";

// IBM Watson
// Your organization and device needs to be registered in IBM Watson IoT Platform.
// Instruction for registering on page
// https://internetofthings.ibmcloud.com/#

//char *client_id = "d:<your Organization ID>:<your Device Type>:<your Device ID>";
char *client_id = "d:yyyyyyy:A_MKR1000:DF48";
char *user_id = "use-token-auth"; // telling that authentication will be done with token
char *authToken = "xxxxxxxxxxxxxx"; // Your IBM Watson Authentication Token

//char *ibm_hostname = "your-org-id.messaging.internetofthings.ibmcloud.com";
char *ibm_hostname = "yyyyyy.messaging.internetofthings.ibmcloud.com";

// sensors and LEDS
const int LEDPin = LED_BUILTIN; // must be a pin that supports PWM. 0...8 on MKR1000
// PModTC1
const int thermoCS = 2; // chip select for MIC3 SPI communication
int thermoByte0 = 0; // 8 bit data from TC1 board
int thermoByte1 = 0; // 8 bit data from TC1 board
int thermoByte2 = 0; // 8 bit data from TC1 board
int thermoByte3 = 0; // 8 bit data from TC1 board
int temp14bit = 0; // 14 most significant bits on a 32 bit integer
int tempRaw = 0;
float tempScaledF = 0;

int blinkState = 0;

/*use this class if you connect using SSL
 * WiFiSSLClient net;
 */
WiFiClient net;
MQTTClient MQTTc;

unsigned long lastSampleMillis = 0;
unsigned long previousWiFiBeginMillis = 0;
unsigned long lastWatsonMillis = 0;
unsigned long lastPrintMillis = 0;

void setup()
{
  pinMode(thermoCS, OUTPUT);
  digitalWrite(thermoCS, HIGH); // for not communicating with MIC3 at the moment
  Serial.begin(9600);
  delay(2000); // Wait for wifi unit to power up
  WiFi.begin(ssid, pass);
  delay(5000); // Wait for WiFi to connect
  Serial.println("Connected to WLAN");
  printWiFiStatus();

  /*
   client.begin("<Address Watson IOT>", 1883, net);
   Address Watson IOT: <WatsonIOTOrganizationID>.messaging.internetofthings.ibmcloud.com
   Example:
   client.begin("iqwckl.messaging.internetofthings.ibmcloud.com", 1883, net);
  */
  MQTTc.begin(ibm_hostname, 1883, net); // Cut for testing without Watson

  connect();

  SPI.begin();
  // Set up the I/O pins

```

```
pinMode(thermoCS, OUTPUT);
pinMode(LEDPin, OUTPUT);

}

void loop() {
  MQTTC.loop(); // Cut for testing without Watson

  // opening and closing SPI communication for reading TC1
  if(millis() - lastSampleMillis > 500)
  {
    lastSampleMillis = millis();
    SPI.beginTransaction(SPISettings(14000000, MSBFIRST, SPI_MODE0));
    digitalWrite(thermoCS, LOW);

    thermoByte0 = SPI.transfer(0x00);
    thermoByte1 = SPI.transfer(0x00);
    thermoByte2 = SPI.transfer(0x00);
    thermoByte3 = SPI.transfer(0x00);

    digitalWrite(thermoCS, HIGH);
    SPI.endTransaction();

    thermoByte0 = thermoByte0 << 24;
    thermoByte1 = thermoByte1 << 16;

    templ4bit =( thermoByte0 | thermoByte1 );

    tempRaw = templ4bit/262144; // shifting 18 bits to right gives multiply of 0,25 degree C.
    tempScaledF = float(templ4bit/262144)/4;
  }

  // Print on serial monitor once in 1000 millisecond
  if(millis() - lastPrintMillis > 1000)
  {
    Serial.print("templ4bit  ");
    Serial.println(templ4bit, BIN);
    Serial.print("  tempScaled  ");
    Serial.println(tempRaw, BIN);
    Serial.print("  tempScaledF  ");
    Serial.println(tempScaledF);

    lastPrintMillis = millis();
  }

  // publish a message every 30 second.
  if(millis() - lastWatsonMillis > 30000)
  {
    Serial.println("Publishing to Watson...");
    if(!MQTTC.connected()) { // Cut for testing without Watson
      connect(); // Cut for testing without Watson
    } // Cut for testing without Watson
    lastWatsonMillis = millis();
    //Cut for testing without Watson

    String wpayload = "{\"d\":{\"TemperatureSensor\":\"TC1 \",\"TempScaledF3AC\":\"" +
String(tempScaledF)+ ", \"TempStreightDF48\":\"" + String(templ4bit)+"}\"}";

    MQTTC.publish("iot-2/evt/TemperatureTC1/fmt/json", wpayload);

  }

  delay(1);
}

// end of loop
}

void connect()
{
  Serial.print("checking WLAN...");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print("."); // printing a dot every half second
    if ( millis() - previousWiFiBeginMillis > 5000) // reconnecting
    {
      previousWiFiBeginMillis = millis();
    }
  }
}
```

```
    WiFi.begin(ssid, pass);
    delay(5000); // Wait for WiFi to connect
    Serial.println("Connected to WLAN");
    printWiFiStatus();
  }
  delay(500);
}
/*
Example:
MQTTc.connect("d:iqwckl:arduino:oxigenarbp", "use-token-auth", "90wT2?a*1WAMVJStb1")

Documentation:
https://console.ng.bluemix.net/docs/services/IoT/iotplatform_task.html#iotplatform_task
*/

Serial.print("\nconnecting Watson with MQTT...");
// Cut for testing without Watson
while (!MQTTc.connect(client_id, user_id, authToken))
{
  Serial.print(".");
  delay(3000);
}
Serial.println("\nconnected!");
}

// messageReceived subroutine needs to be here. MQTT client is calling it.
void messageReceived(String topic, String payload, char * bytes, unsigned int length) {
  Serial.print("incoming: ");
  Serial.print(topic);
  Serial.print(" - ");
  Serial.print(payload);
  Serial.println();
}

void printWiFiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```

---

**Koodi 1.** Arduino MKR1000 lukee pmodTC1 –moduulilta lämpötilan ja lähettää tiedon Watson IoT –alustalle.

Voit kopioida koodin tästä. Mutta siirrä kopiosi ensin esim. Windows Notepad:iin ja kopioi se siitä uudelleen. Näin koodista poistuu näkymättömät muotoilumerkit!!

Koodi on tiedostossa

Wat\_MKR\_SPI\_ThermoC\_H\_DF48.txt

### 3. Käyttöönotto-ohjeet

#### Tehtävä 1

Muuta esimerkin koodia niin, että se siirtää sinun omaan NodeRED-harjoitukseen payload-sanomasisällön, jossa on JSON-rakenne {d:{



```
"Area":Greenhouse1,  
"Temp":21,  
"Humidity":70,  
}  
}.
```

Voit NodeRED-esimerkkien mukaisissa flow – nodeissa muuttaa muuttujat "city" muuttujiksi "area". Tähän muuttujaan kirjoitat laitteeltasi arvon "Greenhouse1".

NodeRED-esimerkkien mukaiseen muuttujaan "temp" kirjoitat laitteeltasi todellisen mittausarvon.

**HUOM! muista muuttaa myös tähän IBM Watson IoT –alustaan:**

**Dashboard**

**Security**

**Connection Security:**

**TLS Optional**

**Tämän jälkeen sammuta laitteestasi sähkö ja anna sen käynnistyä ja ottaa yhteys uudelleen.**

IBM Cloud:ssa on ominaisuus, ettei sama laite saa yrittää liittymistä liian usein. Arduino-koodissa laitteemme yrittää uudelleen 30 sekunnin välein. Jo muutaman minuutin "yrittäminen" väärillä tunnistetiedoilla estää tältä laitteelta pääsyn IBM Cloudiin !!! Voit yrittää vasta jonkin ajan kuluttua uudelleen oikeilla tunnistautumistiedoilla.

Jos et käytä todellista laitetta, voit lähettää saman sisällön mqtt-sanomana valitsemaltasi helpokäyttöiseltä MQTT Client -sovellukselta. Tällainen on esim. MQTTBox.

<http://workswithweb.com/mqttbox.html#>

Katso aiempi käyttöönotto-ohje "IoT\_IBMWatson\_T1\_mqtt\_client\_FI.pdf".

**Raporttiin:** Muutaman rivin selostus. Pari ruutukaappauskuvaa, joissa näkyy anturidata Watson IoT-alustaan luodun anturilaitteen tapahtumana.

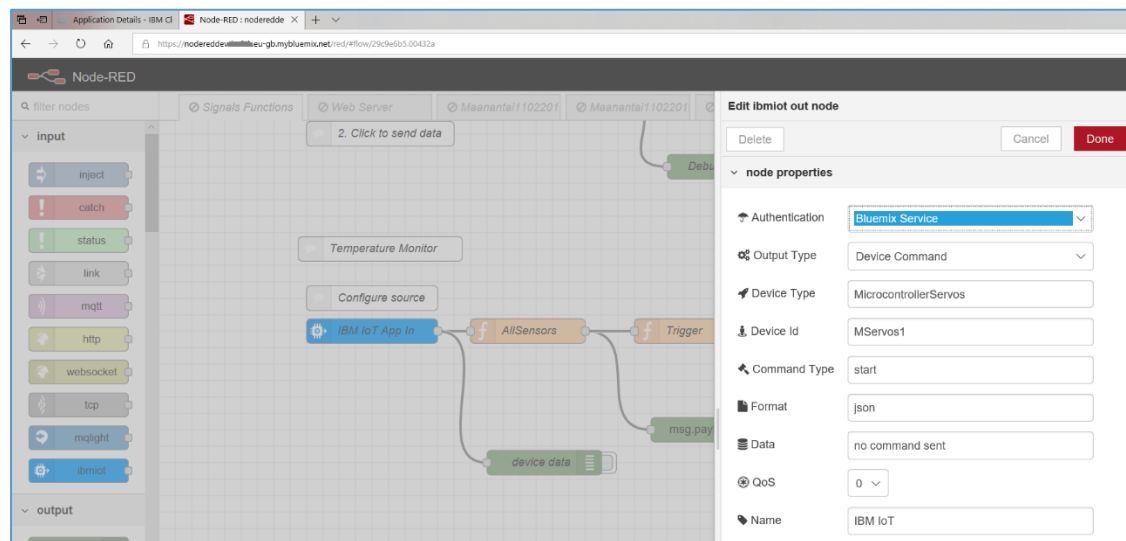
## **Tehtävä 2.**

Luo Watson IoT -alustaan toinenkin laite. Voit sen nimetä

Device Type – laitetyyppi, esim. MQTTClient\_Servos

Device ID – laite, esim. MQTTclient\_Servo\_01

**Huomaa, että IBM Watson IoT -alustaan voi laite kirjautua lukemaan vain command-tyyppisiä sanomia.**



Kuva 3.1

Testaa vastaanoton toimintaa. Kirjoita NodeRED:n puolelta muutama muukin sanoma kuin vain esimerkin käsky käynnistää servo.

**Raporttiin:** Pari ruutukaappauskuvaa, joissa NodeRED flow:n mukainen päätös näkyy command-tyyppisenä tapahtumana IoT-alustan laitenäkymässä.

#### 4. Web -sivu käyttöliittymän

Luomme nyt NodeRED -editoria käyttäen selaimella luettavan web-sivun.

WWW-sivun luontiin käytettävässä NodeRED -flow:ssa on kohtalaisen paljon kirjoitettavaa. Minimoimme tässä tehtävässä virheiden määrän kopiaamalla pohjaksi valmiin flow:n.

NodeRED web page flow:

```
{["id":"ea9fc0ca.2f179","type":"websocket
out","z":"91f97da0.d49da","name":"","server":"9ceb3a2f.eb53e8","client":"","x":700,"y":280,"w
ires":[]},{["id":"cf29a616.572df8","type":"http
response","z":"91f97da0.d49da","name":"","statusCode":"","headers":{"x":690,"y":220,"wires"
:[]},{["id":"31048db7.aa84b2","type":"http
in","z":"91f97da0.d49da","name":"","url":"/controlroom","method":"get","upload":false,"swagg
erDoc":"","x":200,"y":220,"wires":["ce6140f0.ee787"]},{["id":"ce6140f0.ee787","type":"templat
e","z":"91f97da0.d49da","name":"Control Room Web
Page","field":"payload","fieldType":"msg","format":"html","syntax":"mustache","template":"<
!DOCTYPE HTML>\n<html>\n  <head>\n    <title>Sensors and Servos</title>\n    <script
type=\"text/javascript\">\n      var ws;\n      var wsUri = \"ws:~\";\n      var loc =
window.location;\n      console.log(loc);\n      if (loc.protocol === \"https:\") { wsUri = \"wss:~\";
```

```

}\n    // This needs to point to the web socket in the Node-RED flow\n    // ... in this case it's
ws/simple\n    wsUri += \"//\" + loc.host +
loc.pathname.replace(\"controlroom\", \"ws/controlroom\");\n\n    function wsConnect() {\n
console.log(\"connect\", wsUri);\n    ws = new WebSocket(wsUri);\n    //var line = \"\";\n
// either uncomment this for a building list of messages\n    ws.onmessage = function(msg)\n{\n    var line = \"\"; // or uncomment this to overwrite the existing message\n    //
parse the incoming message as a JSON object\n    var data = msg.data;\n
//console.log(data);\n    // build the output from the topic and payload parts of the
object\n    line += \"<p>\"+data+\"</p>\";\n    // replace the messages div with the
new \"line\"\n    document.getElementById('messages').innerHTML = line;\n
//ws.send(JSON.stringify({data:data}));\n    }\n    ws.onopen = function() {\n    //
update the status div with the connection status\n
document.getElementById('status').innerHTML = \"connected\";\n    //ws.send(\"Open
for data\");\n    console.log(\"connected\");\n    }\n    ws.onclose = function() {\n
// update the status div with the connection status\n
document.getElementById('status').innerHTML = \"not connected\";\n    // in case of lost
connection tries to reconnect every 3 secs\n    setTimeout(wsConnect,3000);\n    }\n
}\n\n    function doit(m) {\n    if (ws) { ws.send(m); }\n    }\n    var ssmsg = new
SpeechSynthesisUtterance('Hi there. I am Timo\\'s Watson IoT servoce. Sensor values will be
updated for you once in a minute!');\n    window.speechSynthesis.speak(ssmsg);\n
</script>\n </head>\n <body onload=\"wsConnect();\" onunload=\"ws.disconnect();\">\n
<font face=\"Arial\">\n    <h1>Sensors and servos</h1>\n    <p>The sensor locations and
values - updated when a device sends new values:</p>\n    <div id=\"messages\"></div>\n
<button type=\"button\" onclick='doit(\"click\");'>Click to confirm you have got new
values!</button>\n    <br><br>\n    <button type=\"button\"
onclick='doit(\"masterOFF\");'>Turn all servos OFF !</button>\n    <br><br>\n    <button
type=\"button\" onclick='doit(\"masterON\");'>Turn all servos ON !</button>\n    <button
type=\"button\" onclick='doit(\"servo1ON\");'>Only servo 1 ON !</button>\n    <button
type=\"button\" onclick='doit(\"servo2ON\");'>Only servo 2 ON !</button>\n    <hr>\n
<div id=\"status\">unknown</div>\n    </font>\n
</body>\n</html>\n\", \"x\":449, \"y\":220, \"wires\": [[\"cf29a616.572df8\"]], {\"id\": \"ecfad89c.fef4a8\", \"ty
pe\": \"function\", \"z\": \"91f97da0.d49da\", \"name\": \"format to a string\", \"func\": \"msg.payload =
msg.payload.toString();\\nreturn
msg;\", \"outputs\": 1, \"noerr\": 0, \"x\": 450, \"y\": 280, \"wires\": [[\"ea9fc0ca.2f179\", \"5422f84e.bcc8a8\"]], {\"i
d\": \"d803ba69.8d6d58\", \"type\": \"websocket
in\", \"z\": \"91f97da0.d49da\", \"name\": \"\", \"server\": \"9ceb3a2f.eb53e8\", \"client\": \"\", \"x\": 470, \"y\": 360, \"wir
es\": [[\"b092a074.dd087\", \"6245ffe1.dd6af\"]], {\"id\": \"b092a074.dd087\", \"type\": \"debug\", \"z\": \"91f97
da0.d49da\", \"name\": \"\", \"active\": true, \"console\": \"false\", \"complete\": \"false\", \"x\": 690, \"y\": 360, \"wires\"
: []}, {\"id\": \"9569b80d.4440d8\", \"type\": \"ibmiot
in\", \"z\": \"91f97da0.d49da\", \"authentication\": \"boundService\", \"apiKey\": \"\", \"inputType\": \"evt\", \"logica
lInterface\": \"\", \"ruleId\": \"\", \"deviceId\": \"MSensors1\", \"applicationId\": \"\", \"deviceType\": \"Microcontroll
erSensors\", \"eventType\": \"+\", \"commandType\": \"\", \"format\": \"json\", \"name\": \"IBM IoT App
In\", \"service\": \"registered\", \"allDevices\": false, \"allApplications\": false, \"allDeviceTypes\": false, \"allLogi
calInterfaces\": false, \"allEvents\": true, \"allCommands\": false, \"allFormats\": false, \"qos\": \"0\", \"x\": 100, \"y
\": 380, \"wires\": [[\"fe3f970.6ec0168\", \"a4d4306d.4a7e2\"]], {\"id\": \"fe3f970.6ec0168\", \"type\": \"debug\",
\"z\": \"91f97da0.d49da\", \"name\": \"\", \"active\": false, \"tosidebar\": true, \"console\": false, \"tostatus\": false,
\"complete\": \"false\", \"x\": 310, \"y\": 520, \"wires\": []}, {\"id\": \"a4d4306d.4a7e2\", \"type\": \"json\", \"z\": \"91f97d
a0.d49da\", \"name\": \"\", \"property\": \"payload\", \"action\": \"\", \"pretty\": false, \"x\": 270, \"y\": 340, \"wires\": [[\"

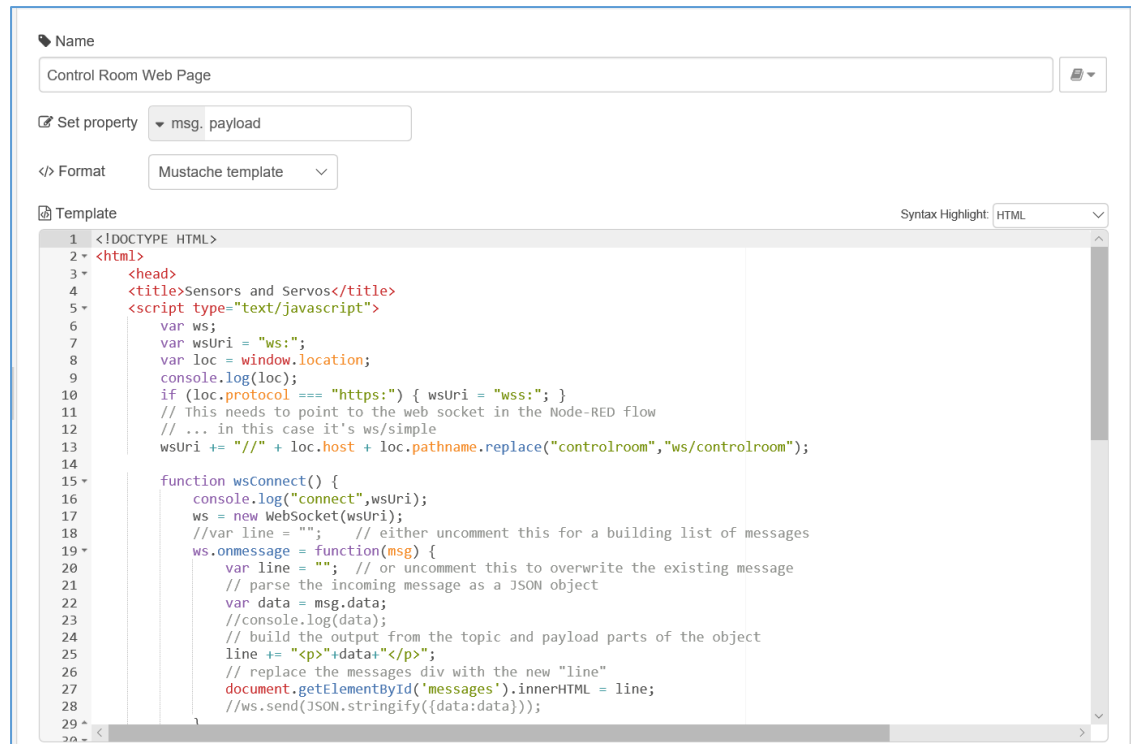
```

```

ecfad89c.fef4a8","66824dfb.1ce1e4"]],{"id":"66824dfb.1ce1e4","type":"debug","z":"91f97da0.d49da","name":"","active":false,"tosidebar":true,"console":false,"tostatus":false,"complete":false,"x":530,"y":520,"wires":[]},{"id":"5422f84e.bcc8a8","type":"debug","z":"91f97da0.d49da","name":"","active":false,"tosidebar":true,"console":false,"tostatus":false,"complete":false,"x":790,"y":520,"wires":[]},{"id":"6245ffe1.dd6af","type":"function","z":"91f97da0.d49da","name":"Trigger","func":"var fpayload = msg.payload;\nvar trigger = [false,false];\nvar msgOut = [\"\\\",\\\"\\\"];\n\nif (fpayload == \"masterOFF\")\n{\n  trigger[0] = false;\n  trigger[1] = false;\n  msgOut[0] = {payload:{\"mcommand\":trigger[0]}};\n  msgOut[1] = {payload:{\"mcommand\":trigger[1]}};\n}\n\nif (fpayload == \"masterON\")\n{\n  trigger[0] = true;\n  trigger[1] = true;\n  msgOut[0] = {payload:{\"mcommand\":trigger[0]}};\n  msgOut[1] = {payload:{\"mcommand\":trigger[1]}};\n}\n\nif (fpayload == \"servo1ON\")\n{\n  trigger[0] = true;\n  msgOut[0] = {payload:{\"mcommand\":trigger[0]}};\n  msgOut[1] = {payload:{\"mcommand\":trigger[1]}};\n}\n\nif (fpayload == \"servo2ON\")\n{\n  trigger[1] = true;\n  msgOut[0] = {payload:{\"mcommand\":trigger[0]}};\n  msgOut[1] = {payload:{\"mcommand\":trigger[1]}};\n}\n\nreturn msgOut;","outputs":2,"noerr":0,"x":440,"y":700,"wires":[["dfa2b3c3.4ee48","9e8c3344.3d508","b9b4a7fa.ad30c8","a30661ce.407b6"],["b01f3595.3f0dc8","c9723a15.212858","854b44d5.083cd8","4be5186a.a73f78"]],{"id":"dfa2b3c3.4ee48","type":"debug","z":"91f97da0.d49da","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":false,"x":710,"y":740,"wires":[]},{"id":"9e8c3344.3d508","type":"ibmiot out","z":"91f97da0.d49da","authentication":"boundService","apiKey":"","outputType":"evt","deviceId":"MServos1","deviceType":"MicrocontrollerServos","eventCommandType":"start","format":"json","data":"no command sent","qos":0,"name":"send event to IBM IoT PI","service":"registered","x":730,"y":620,"wires":[]},{"id":"b9b4a7fa.ad30c8","type":"debug","z":"91f97da0.d49da","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload.mcommand","x":730,"y":780,"wires":[]},{"id":"b01f3595.3f0dc8","type":"ibmiot out","z":"91f97da0.d49da","authentication":"boundService","apiKey":"","outputType":"evt","deviceId":"MServos2","deviceType":"MicrocontrollerServos","eventCommandType":"start","format":"json","data":"no command sent","qos":0,"name":"send event to IBM IoT PI","service":"registered","x":730,"y":840,"wires":[]},{"id":"c9723a15.212858","type":"debug","z":"91f97da0.d49da","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":false,"x":710,"y":960,"wires":[]},{"id":"854b44d5.083cd8","type":"debug","z":"91f97da0.d49da","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload.mcommand","x":730,"y":1000,"wires":[]},{"id":"a30661ce.407b6","type":"ibmiot out","z":"91f97da0.d49da","authentication":"boundService","apiKey":"","outputType":"cmd","deviceId":"MServos1","deviceType":"MicrocontrollerServos","eventCommandType":"start","format":"json","data":"no command sent","qos":0,"name":"Send command to IBM IoT device","service":"registered","x":760,"y":680,"wires":[]},{"id":"4be5186a.a73f78","type":"ibmiot out","z":"91f97da0.d49da","authentication":"boundService","apiKey":"","outputType":"cmd","deviceId":"MServos2","deviceType":"MicrocontrollerServos","eventCommandType":"start","format":"json","data":"no command sent","qos":0,"name":"Send command to IBM IoT device","service":"registered","x":760,"y":900,"wires":[]},{"id":"9ceb3a2f.eb53e8","type":"websocket-listener","z":"91f97da0.d49da","path":"/ws/controlroom","wholemsg":false}]

```





Kuva 4.3 Control Room Web page

Node:n java script -koodi ei mahdu kokonaisuudessaan kuvaan. Tämän saat myös tekstinä.

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Sensors and Servos</title>
    <script type="text/javascript">
      var ws;
      var wsUri = "ws:";
      var loc = window.location;
      console.log(loc);
      if (loc.protocol === "https:") { wsUri = "wss:"; }
      // This needs to point to the web socket in the Node-RED flow
      // ... in this case it's ws/simple
      wsUri += "://" + loc.host + loc.pathname.replace("controlroom", "ws/controlroom");

      function wsConnect() {
        console.log("connect", wsUri);
        ws = new WebSocket(wsUri);
        //var line = ""; // either uncomment this for a building list of messages
        ws.onmessage = function(msg) {
          var line = ""; // or uncomment this to overwrite the existing message
          // parse the incoming message as a JSON object
          var data = msg.data;
          //console.log(data);
          // build the output from the topic and payload parts of the object
          line += "<p>" + data + "</p>";
          // replace the messages div with the new "line"
          document.getElementById('messages').innerHTML = line;
          //ws.send(JSON.stringify({data:data}));
        }
      }

      ws.onopen = function() {
        // update the status div with the connection status
        document.getElementById('status').innerHTML = "connected";
      }
    </script>
  </head>
  <body>
    <div id="messages"></div>
    <div id="status"></div>
  </body>
</html>

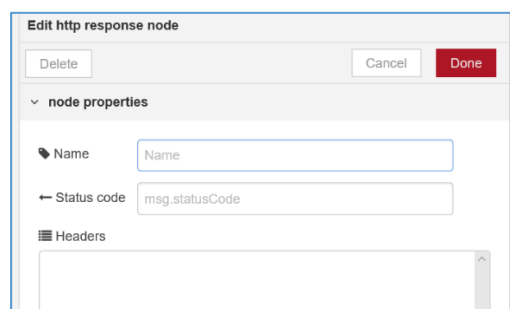
```

```
//ws.send("Open for data");
console.log("connected");
}
ws.onclose = function() {
  // update the status div with the connection status
  document.getElementById('status').innerHTML = "not connected";
  // in case of lost connection tries to reconnect every 3 secs
  setTimeout(wsConnect,3000);
}
}

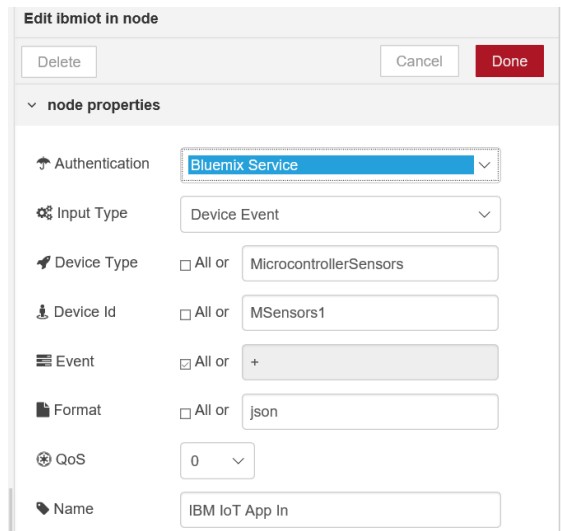
function doit(m) {
  if (ws) { ws.send(m); }
}

var ssmsg = new SpeechSynthesisUtterance('Hi there. I am Timo\'s Watson IoT servoce. Sensor values will be
updated for you once in a minute!');
window.speechSynthesis.speak(ssmsg);
</script>
</head>
<body onload="wsConnect();" onunload="ws.disconnect();">
  <font face="Arial">
    <h1>Sensors and servos</h1>
    <p>The sensor locations and values - updated when a device sends new values:</p>
    <div id="messages"></div>
    <button type="button" onclick='doit("click");'>Click to confirm you have got new values!</button>
    <br><br>
    <button type="button" onclick='doit("masterOFF");'>Turn all servos OFF !</button>
    <br><br>
    <button type="button" onclick='doit("masterON");'>Turn all servos ON !</button>
    <button type="button" onclick='doit("servo1ON");'>Only servo 1 ON !</button>
    <button type="button" onclick='doit("servo2ON");'>Only servo 2 ON !</button>
    <hr>
    <div id="status">unknown</div>
  </font>
</body>
</html>
```

Koodi 4.1 html-koodi selaimella avattavaan sivuun.



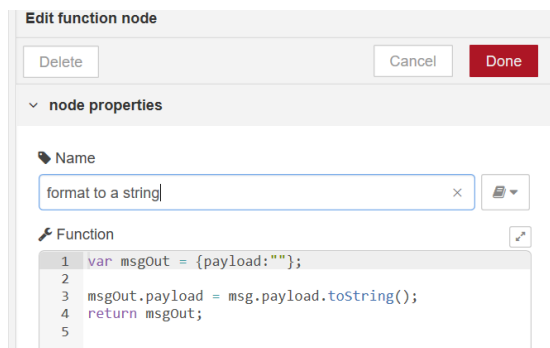
Kuva 4.4 http response node jätetään tyhjäksi.



Kuva 4.5 ibm iot node

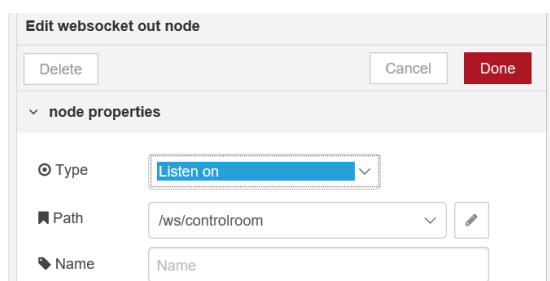
Input node tyyppiä ibm iot ottaa vastaan sanomat Watson IoT -alustalta. Siihen voidaan määritteellä tarkalleen, minkä laitteen sanomia luetaan. Voidaan vaihtoehtoisesti lukea kaikentyyppisten laitteiden kaikki sanomat – valitaan kaikkiin kohtiin ”all”. Mutta on hiukan vaarallista valita laitetypiksi ”all” ja laitteeksi ”all”! Sanomat saattavat jäädä kiertämään silmukkaa!

Flown JSON -node muuntaa JSON objekteja tyyppiin string ja päinvastoin. Siihen ei muuteta mitään.



Kuva 4.6 function node

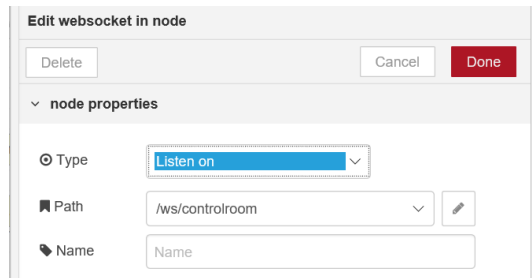
Function node poimii sanomasta joko koko sisällön tai haluttaessa se voidaan määrittää poimimaan JSON-muotoisesta saapuvasta payload:sta vain tietty sisältö.



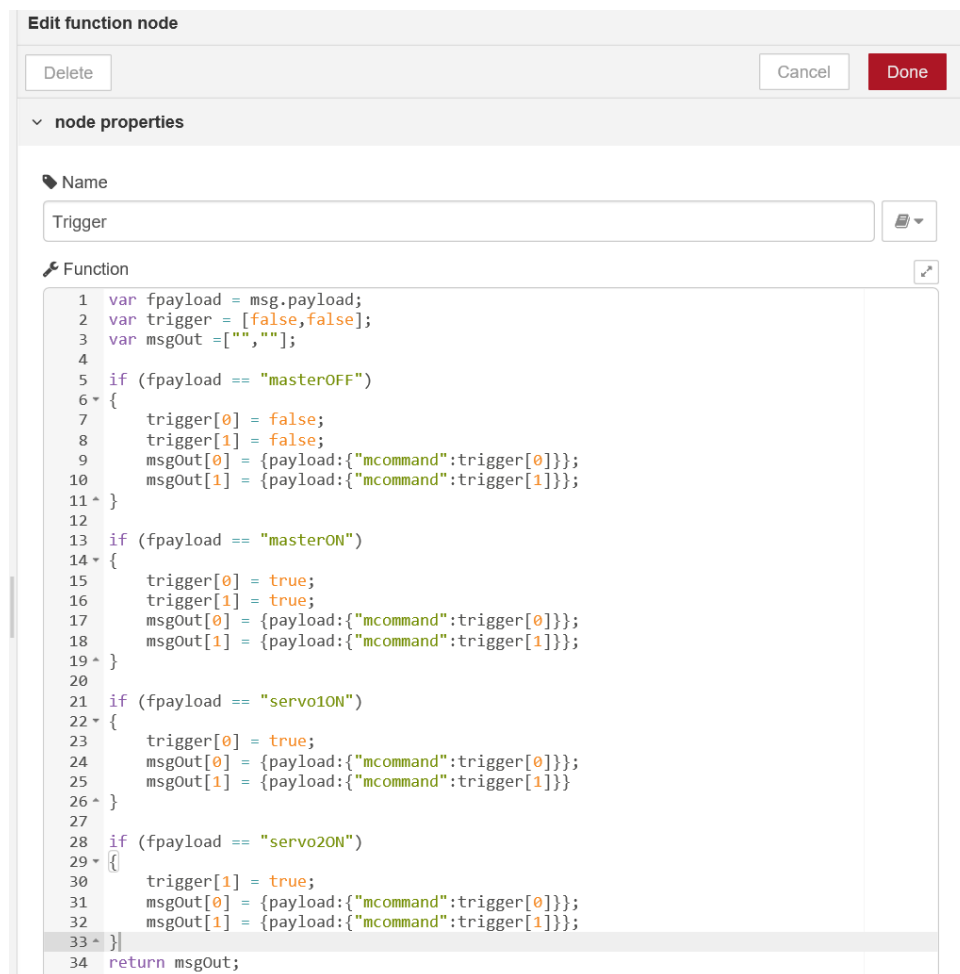
Kuva 4.7 Output node Websocket



Web socket -node:ssa määritellään web socket:lle nimi. Tämän on tietenkin vastattava javascript-koodissa käytettyä nimeä.



Kuva 4.8 Input node Web socket



Kuva 4.9 Function node Trigger

Java script -koodi on tässä myös tekstinä.

```
var fpayload = msg.payload;
var trigger = [false,false];
var msgOut =["", ""];

if (fpayload == "masterOFF")
{
    trigger[0] = false;
```

```
trigger[1] = false;
msgOut[0] = {payload:"mcommand":trigger[0]};
msgOut[1] = {payload:"mcommand":trigger[1]};
}

if (fpayload == "masterON")
{
    trigger[0] = true;
    trigger[1] = true;
    msgOut[0] = {payload:"mcommand":trigger[0]};
    msgOut[1] = {payload:"mcommand":trigger[1]};
}

if (fpayload == "servo1ON")
{
    trigger[0] = true;
    msgOut[0] = {payload:"mcommand":trigger[0]};
    msgOut[1] = {payload:"mcommand":trigger[1]};
}

if (fpayload == "servo2ON")
{
    trigger[1] = true;
    msgOut[0] = {payload:"mcommand":trigger[0]};
    msgOut[1] = {payload:"mcommand":trigger[1]};
}
return msgOut;
```

Koodi 4.2 Java script -koodi ohjauksen päätöksiä tekevässä funktiossa

Edit ibmiot out node

Delete Cancel Done

node properties

Authentication Bluemix Service

Output Type Device Event

Device Type MicrocontrollerServos

Device Id MServos1

Event Type start

Format json

Data no command sent

QoS 0

Name send event to IBM IoT PI

Kuva 4.10 Output node ibm iot – Device Event

Kuva 4.11 Output node ibm iot – Device Command

Huomaa, että Watson IoT -alustaan voidaan laitteelle lähettää joko event-tyyppinen sanoma tai command-tyyppinen sanoma.

Olemme nyt käyneet läpi kaikki flow-kaavio node:t.

## 5. Web Page -käyttöliittymän käyttöönotto tehtävä

### Tehtävä 3

Testaa web page -esimerkin toimintaa.

Java script koodiin web page -node:en on kommenttiriveille kirjattu, miten sivun saakin toimimaan niin, että tapahtumat kertyvät allekkaisiksi riveiksi selaimella luettavalle sivulle. Kokeile tätä.

**Raporttiin:** Toimiiko koodi kommentin vaihtoehtojen mukaisesti? Miksi toimii? Miksi ei toimi?