Watson IoT Platform – T3 - NodeRed – Device – connection

Käyttöönottotehtävä 3

1. Johdanto

Tässä käyttöönottotehtävässä luomme yhteyden todellisen laitteen ja IBM Watson NodeRED –palvelimen välille.

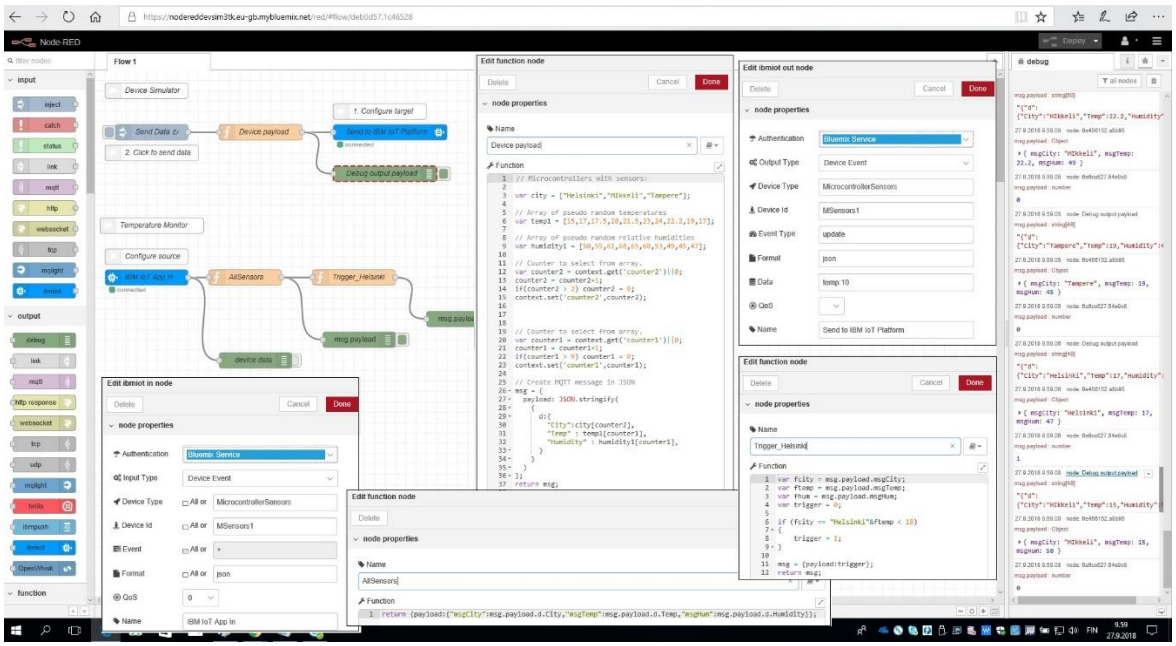
Teemme myös web-selaimella toimivan käyttöliittymän.

Käyttöönottotehtävässä 1 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

https://console.bluemix.net/docs/services/IoT/index.html#gettingstartedtemplate

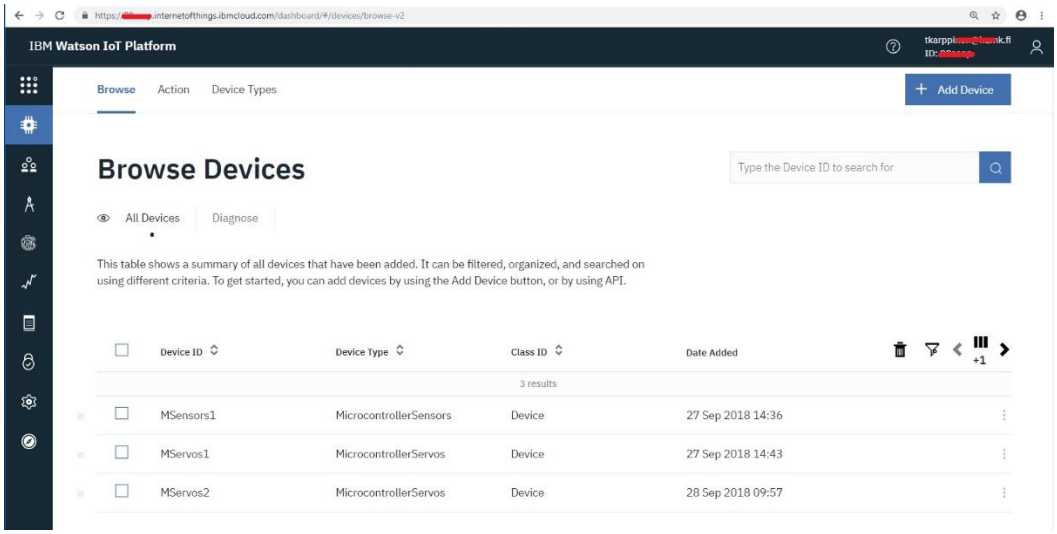
Käyttöönottotehtävässä 2 käsittelimme laitedataa Watson IoT-alustalla.



Kuva 1.1 Käyttöönottotehtävän 2 NodeRED –flow ja kuhunkin node-lohkoon kirjatut sisällöt.

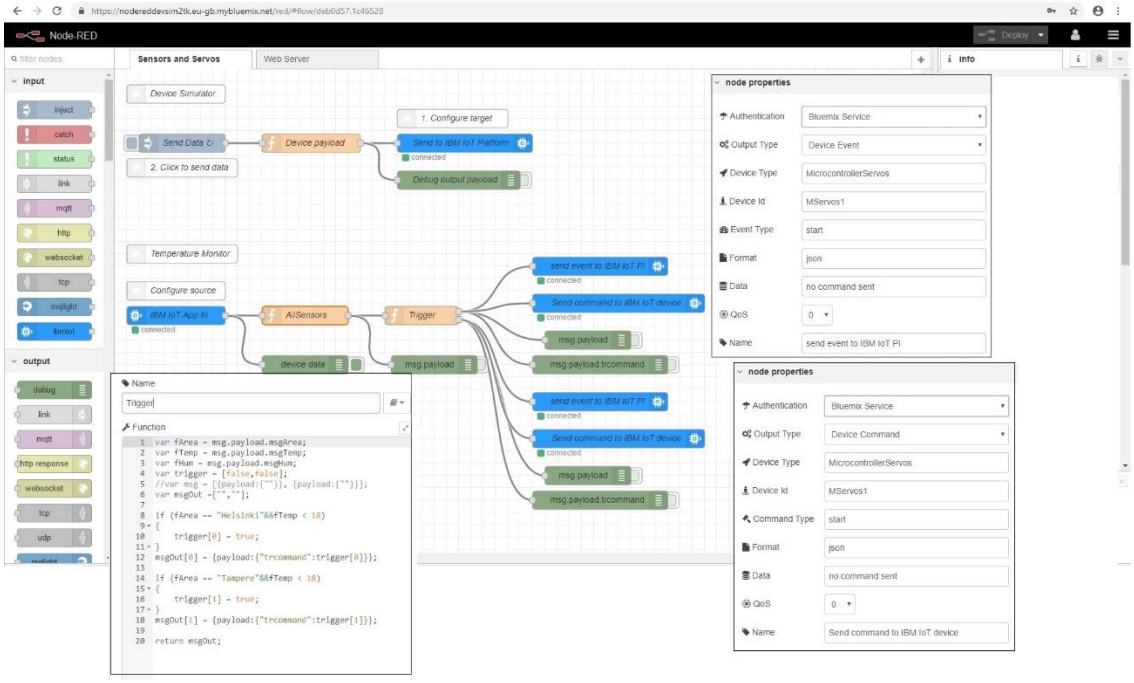
Kuvassa näkyvä laitedataa simulointina luova flow-kaavio ja laitedataa laitteelta lukeva flow-kaavio on nopea kirjoittaa, jos et niitä ole aiemmin tehnyt.

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



Kuva 1.2 Käyttöönottotehtävän 2 laitteet Devices – laitenäkymässä.

laitteelle – oikealle tai simuloidulle – voidaan kirjoitta seuraavan kuvan mukaisesti.



Kuva 1.3 Käyttöönottotehtävän 2 laitteelle kirjoitus.

Edellä kuvatun flown: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","applicationId":"","deviceType":"MicrocontrollerSensors","eventType":"+","commandType":"","format":"json","name":"IBM IoT App In","service":"registered","allDevices":false,"allApplications":false,"allDeviceTypes":false,"allLogicalInterfaces":false,"allEvents":true,"allCommands":false,"allFormats":false,"qos":"0","x":100,"y":400,"wires":[["ae0082ac.51ff8","c0c482df.3f3b8"]]},{"id":"ae0082ac.51ff8","type":"function","z":"deb0d57.1c46528","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":"evt","deviceId":"MSensors1","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":"Device payload","func":"// Microcontrollers with sensors:\n\nvar
area1 = [\"Greenhouse1\",\"Greenhouse2\",\"Greenhouse3\"];\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":"86df0b6c.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\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.bluemix.net/docs/services/IoT/iotplatform_task.html)","x":100,"y":40,"wires":[]},{"id":"141b7c7.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//var msg =
[{payload:{\"\"}}, {payload:{\"\"}}];\nvar msgOut =[\"\",\"\"];\n\nif (fArea == \"Helsinki\"&&fTemp < 18)\n{\n trigger[0] = true;\n}\nmsgOut[0] =
{payload:{\"trcommand\":trigger[0]}};\n\nif (fArea == \"Tampere\"&&fTemp < 18)\n{\n trigger[1] = true;\n}\nmsgOut[1] =
{payload:{\"trcommand\":trigger[1]}};\n\nreturn
msgOut;","outputs":2,"noerr":0,"x":500,"y":400,"wires":[["86eabffb.b92be","8e9ae4a8.9edf88","476d79d6.e80738","a84195d8.b33de8"],["9e624f2a.1
e153","21af5407.97aeac","c4c0d1.71c2af3","351079b3.3dfe66"]]},{"id":"82c0704.645f29","type":"debug","z":"deb0d57.1c46528","name":"","active":f
alse,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","x":520,"y":480,"wires":[]},{"id":"86eabffb.b92be","type":"debug","z":"deb0d5
7.1c46528","name":"","active":false,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","x":770,"y":440,"wires":[]},{"id":"8e9ae4a8.9ed
f88","type":"ibmiot
out","z":"deb0d57.1c46528","authentication":"boundService","apiKey":"","outputType":"evt","deviceId":"MServos1","deviceType":"MicrocontrollerSer
vos","eventCommandType":"start","format":"json","data":"no command sent ","qos":0,"name":"send event to IBM IoT
PI","service":"registered","x":790,"y":320,"wires":[]},{"id":"476d79d6.e80738","type":"debug","z":"deb0d57.1c46528","name":"","active":false,"tosideb
ar":false,"console":false,"tostatus":false,"complete":"payload.trcommand","x":790,"y":480,"wires":[]},{"id":"9e624f2a.1e153","type":"ibmiot
out","z":"deb0d57.1c46528","authentication":"boundService","apiKey":"","outputType":"evt","deviceId":"MServos2","deviceType":"MicrocontrollerSer
vos","eventCommandType":"start","format":"json","data":"no command sent","qos":0,"name":"send event to IBM IoT
PI","service":"registered","x":790,"y":540,"wires":[]},{"id":"21af5407.97aeac","type":"debug","z":"deb0d57.1c46528","name":"","active":false,"tosideb
ar":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
out","z":"deb0d57.1c46528","authentication":"boundService","apiKey":"","outputType":"cmd","deviceId":"MServos1","deviceType":"MicrocontrollerSe
rvos","eventCommandType":"start","format":"json","data":"no command sent","qos":0,"name":"Send command to IBM IoT
device","service":"registered","x":820,"y":380,"wires":[]},{"id":"351079b3.3dfe66","type":"ibmiot
out","z":"deb0d57.1c46528","authentication":"boundService","apiKey":"","outputType":"cmd","deviceId":"MServos2","deviceType":"MicrocontrollerSe
rvos","eventCommandType":"start","format":"json","data":"no command sent","qos":0,"name":"Send command to IBM IoT
device","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.

Tehtävä 1

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. A_MKR1000

Device ID – laite, esim. 1234

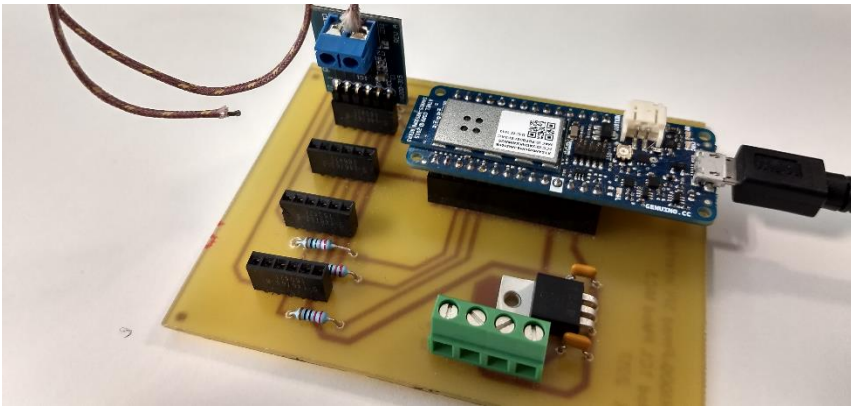
ja annetaan järjestelmän luoda kirjautumistiedot. **Tallenna organization ID …..Authentication Token … kaikki syntyvät kirjautmistiedot myöhempää käyttöä varten tekstitiedostoon.**

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

Ohjeessa ” IoT_IBMWatson_MKR1000_example1_v30012018.pdf” kerrotaan yksityiskohtaisesti, miten laitteen kirjautumistiedot luodaan ja miten laite liitetään Watson IoT –alustaan.

Ohjeen ”IoT_IBMWatson_mqtt_Harj1_v05092018.pdf” lopussa näytetään, miten Windows-tietokoneeseen asennettua MQTT Client –sovellusta MQTTBox käytetään ”laitteena”.

Tässä laboratorioharjoituksessa 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 temeprature

The reference junction compensation is calculted 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[] = "xxxxxxxxxxx";    // 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:xxxxxxxxxxx: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 = "v8nnas.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;


    temp14bit =( thermoByte0 | thermoByte1 );

    tempRaw = temp14bit/262144; // shifting 18 bits to right gives multiply of 0,25 degree C.
    tempScaledF = float(temp14bit/262144)/4;

  }

  // Print on serial monitor once in 1000 millisecond
  if(millis() - lastPrintMillis > 1000)
  {
    Serial.print("temp14bit   ");
    Serial.println(temp14bit, 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
         MQTTc.publish("iot-2/evt/TemperatureTC1/fmt/json", "{\"Temperature sensors\":\"TC1
\",\"TempScaledDF48\":" + String(tempScaledF)+", \"TempStreightDF48\":" + String(temp14bit)+"});
```

```
     }

    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:oxigenarbpm","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

Tehtävä 2

Muuta esimerkin koodia niin, että se siirtää sinun omaan NodeRED-harjoitukseen tilaan "Greenhouse1" lämpötilatiedon "Temp".

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! muita 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" estää tältä laitteelta pääsyn IBM Cloudiin !!! Voit yrittää jonkin ajan kuluttua uudelleen. Mutta, mikähän on tämä jokin aika?

Tehtävä 3

Tähän tehtävään voit käyttää eri Arduino MKR1000 korttia. Voi tämän "teknisesti" tehdä samallakin kortilla mutta näin ratkaistuna tämä ei ole niin tositilannetta vastaava.

Edellä esitetyssä Arduino –koodissa on näkyvissä myös sanoman vastaanotto. Siinä on valmiina rivit, joilla vastaanotettu sanoma kirjoitetaan "Serial Monitor" –ikkunaan.

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