Practica Cloud Week 2 Lennart van Koppen 11031859

Vraag 1:

De bewegingssensor is een aangepaste vorm van het voorbeeld van Azure IoTHub

'use strict';

var connectionString =

  "HostName=DelftIotHubPracticum2021.azure-devices.net;SharedAccessKeyName=service;DeviceId=device-student-22-device-1;SharedAccessKey=7c3SGtU21s8fUjcSGMMjXsO5sebBaPu2ix9Zg1Pj/I4=";

var Mqtt = require('azure-iot-device-mqtt').Mqtt;

var DeviceClient = require('azure-iot-device').Client

var Message = require('azure-iot-device').Message;

const readline = require('readline');

var client = DeviceClient.fromConnectionString(connectionString, Mqtt);

const rl = readline.createInterface({

  input: process.stdin,

  output: process.stdout

});

client.on('message', (msg) => {

  msg = JSON.parse(msg.getData());

      console.log(msg);

});

rl.on('line', (input) => {

  console.log("Er is beweging geconstateerd");

  var message = new Message(JSON.stringify({beweging: true}))

  client.sendEvent(message, function (err) {

    if (err) {

      console.error('send error: ' + err.toString());

    } else {

      console.log('message sent');

    }

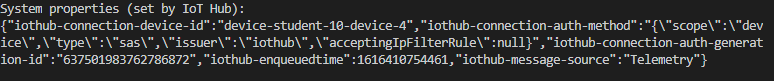
  });

});

Vraag 2:

IoTHub

Hieronder de metadata die IoTHub zelf toevoegd. Daaronder de code van de service die de berichten uitleest.



'use strict';

var Client = require('azure-iothub').Client;

var Message = require('azure-iot-device').Message;

const { EventHubConsumerClient } = require("@azure/event-hubs");

const connectionString = `Endpoint=sb://iothub-ns-delftiothu-8528485-c734b6e251.servicebus.windows.net/;EntityPath=delftiothubpracticum2021;SharedAccessKeyName=service;SharedAccessKey=7c3SGtU21s8fUjcSGMMjXsO5sebBaPu2ix9Zg1Pj/I4=`;

var printError = function (err) {

    console.log(err.message);

  };

const devices = ["device-student-22-device-1", "device-student-22-device-2", "device-student-22-device-3","device-student-22-device-4","device-student-22-device-5" ]

  var printMessages = function (messages) {

    for (const message of messages) {

      if(devices.includes(message.systemProperties["iothub-connection-device-id"])){

        console.log("Telemetry received: ");

        console.log(JSON.stringify(message.body));

        console.log("Properties (set by device): ");

        console.log(JSON.stringify(message.properties));

        console.log("System properties (set by IoT Hub): ");

        console.log(JSON.stringify(message.systemProperties));

        console.log("");

      }

    }

  };

  async function main() {

    console.log("IoT Hub Quickstarts - Read device to cloud messages.");

    const clientOptions = {

    };

    const consumerclient = new EventHubConsumerClient("$Default", connectionString, clientOptions);

    consumerclient.subscribe({

      processEvents: printMessages,

      processError: printError,

    });

  }

  main().catch((error) => {

    console.error("Error running sample:", error);

  });

Vraag 3:

De lamp is gebaseerd op de lamp zoals in week 1, code hiervan is hieronder te vinden:

'use strict';

const deviceId = "device-student-22-device-3";

const sas = "c79+me18yA0j2rqdBKNjNC+SEBZ8uV0CJdTe+pE6o7E=";

const connectionString = `HostName=DelftIotHubPracticum2021.azure-devices.net;DeviceId=${deviceId};SharedAccessKey=${sas};`

var Mqtt = require('azure-iot-device-mqtt').Mqtt;

var DeviceClient = require('azure-iot-device').Client

var Message = require('azure-iot-device').Message;

const chalk = require('chalk');

var client = DeviceClient.fromConnectionString(connectionString, Mqtt);

client.on('message', (msg) => {

    msg = JSON.parse(msg.getData());

    //updatesettings(msg);

});

let lampsettings = {

  "interval" : 2350,

  "status": 1,

  "alarm" : 0,

  "rgb":{

      "r":255,

      "g":255,

      "b":255

  }

}

console.log(chalk.rgb(lampsettings.rgb.r, lampsettings.rgb.g, lampsettings.rgb.b).underline('This is a simulated dimmable bulb'));

lampsettings.status = 0

lampsettings.rgb.r = 128;

lampsettings.rgb.g = 128;

lampsettings.rgb.b = 128;

function print() {

    //console.clear();

    if (lampsettings.status == 0) {

        console.log(chalk.rgb(0, 0, 0).underline('Light is switched off'));

    }

    else {

        console.log(chalk.rgb(lampsettings.rgb.r, lampsettings.rgb.g, lampsettings.rgb.b)('This is a simulated bulb'));

    }

}

function updatestatus(request, response){

    function directMethodResponse(err) {

    if(err) {

        console.error(chalk.red('An error ocurred when sending a method response:\n' + err.toString()));

        } else {

            console.log(chalk.green('Response to method \'' + request.methodName + '\' sent successfully.' ));

        }

    }

    response.send(200, 'bulb status set: ' + request.payload, directMethodResponse);

    lampsettings[request.methodName] = request.payload

    print()

}

client.onDeviceMethod('status', updatestatus)

Vraag 4:

Om de lamp aan te sturen wordt gebruik gemaakt van een direct method, deze is niet vanuit de bewegingsensor aan te roepen vandaar dat ik een service heb gemaakt. De service is herleidbaar van de eerder gemaakte service die de berichten uitleest. Deze filtert hij en bij de juiste device reageert het door een direct method aan te roepen die de lamp heeft. De code van de lamp en bewegingsensor zijn niet aangepast dus deze zijn gelijk aan de eerdere benoemde en heb ik daarom niet hieronder neergezet.

'use strict';

var Client = require('azure-iothub').Client;

var Message = require('azure-iot-device').Message;

const { EventHubConsumerClient } = require("@azure/event-hubs");

const connectionString = `Endpoint=sb://iothub-ns-delftiothu-8528485-c734b6e251.servicebus.windows.net/;EntityPath=delftiothubpracticum2021;SharedAccessKeyName=service;SharedAccessKey=7c3SGtU21s8fUjcSGMMjXsO5sebBaPu2ix9Zg1Pj/I4=`;

var printError = function (err) {

    console.log(err.message);

  };

const devices = ["device-student-22-device-1", "device-student-22-device-2", "device-student-22-device-3","device-student-22-device-4","device-student-22-device-5" ]

  var printMessages = function (messages) {

    for (const message of messages) {

      if(devices.includes(message.systemProperties["iothub-connection-device-id"])){

        if(message.systemProperties["iothub-connection-device-id"] === 'device-student-22-device-1'){

            lampbeweging()

        }

        console.log("Telemetry received: ");

        console.log(JSON.stringify(message.body));

        console.log("");

      }

    }

  };

  async function main() {

    console.log("IoT Hub Quickstarts - Read device to cloud messages.");

    const clientOptions = {

    };

    const consumerclient = new EventHubConsumerClient("$Default", connectionString, clientOptions);

    consumerclient.subscribe({

      processEvents: printMessages,

      processError: printError,

    });

  }

  main().catch((error) => {

    console.error("Error running sample:", error);

  });

const cconnectionstring = "HostName=DelftIotHubPracticum2021.azure-devices.net;SharedAccessKeyName=service;SharedAccessKey=7c3SGtU21s8fUjcSGMMjXsO5sebBaPu2ix9Zg1Pj/I4="

var client = Client.fromConnectionString(cconnectionstring);

function lampbeweging(){

    var deviceId = 'device-student-22-device-3';

    var methodParams = {

        methodName: 'status',

        payload: 1

      };

    client.invokeDeviceMethod(deviceId, methodParams, function (err, result) {

        if (err) {

            console.error('Failed to invoke method \'' + methodParams.methodName + '\': ' + err.message);

        } else {

            console.log('Response from ' + methodParams.methodName + ' on ' + deviceId + ':');

            console.log(JSON.stringify(result, null, 2));

        }

      });

}

Vraag 5:

Devices kunnen niet zelf ( tenzij je ze ook verbinding geeft tot het eventhub) de berichten inlezen. Vandaar dat een extra service nodig is om dit te doen. De service heeft de logica in handen en de devices hoeven dus niet na te denken.

Vraag 6

Ik lees in de lampcontroller eerst mijn waardes uit daarna stuur ik een bericht, deze wordt gelezen door de service waarna de service via een direct method de informatie naar de lamp stuurt. Nadat de lamp zijn response geeft van de direct method wordt een andere direct method aangeroepen die een bericht stuurt naar de lampcontroller.

**Lamp.js:**

'use strict';

const deviceId = "device-student-22-device-3";

const sas = "c79+me18yA0j2rqdBKNjNC+SEBZ8uV0CJdTe+pE6o7E=";

const connectionString = `HostName=DelftIotHubPracticum2021.azure-devices.net;DeviceId=${deviceId};SharedAccessKey=${sas};`

var Mqtt = require('azure-iot-device-mqtt').Mqtt;

var DeviceClient = require('azure-iot-device').Client

var Message = require('azure-iot-device').Message;

const chalk = require('chalk');

var client = DeviceClient.fromConnectionString(connectionString, Mqtt);

let lampsettings = {

  "interval" : 2350,

  "status": 1,

  "alarm" : 0,

  "rgb":{

      "r":255,

      "g":255,

      "b":255

  }

}

console.log(chalk.rgb(lampsettings.rgb.r, lampsettings.rgb.g, lampsettings.rgb.b).underline('This is a simulated dimmable bulb'));

lampsettings.status = 0

lampsettings.rgb.r = 128;

lampsettings.rgb.g = 128;

lampsettings.rgb.b = 128;

function print() {

    //console.clear();

    if (lampsettings.status == 0) {

        console.log(chalk.rgb(0, 0, 0).underline('Light is switched off'));

    }

    else {

        console.log(chalk.rgb(lampsettings.rgb.r, lampsettings.rgb.g, lampsettings.rgb.b)('This is a simulated bulb'));

    }

}

function updatestatus(request, response){

    function dmr (err) {

        if (err) {

            console.error(chalk.red('An error ocurred when sending a method response:\n' + err.toString()));

        } else {

            console.log(chalk.green('Response to method \'' + request.methodName + '\' sent successfully.'));

            print();

        }

    }

    if (typeof request.payload == 'undefined' && request.payload) {

        console.log(chalk.red('Invalid interval response received in payload'));

        // Report failure back to your hub.

        response.send(400, 'Invalid direct method parameter: ' + JSON.stringify(request.payload), dmr);

    } else {

        Object.assign(lampsettings, request.payload);

        response.send(200, JSON.stringify(request.methodName), dmr);

    }

}

client.onDeviceMethod('setlamp', updatestatus)

)

**lampcontrol:**

'use strict';

const deviceId = "device-student-22-device-1";

const sas = "ff2aanu0rv7UfoGB1b8NFT2Sp0qRlbJcS/t314PBtd0=";

const connectionString = `HostName=DelftIotHubPracticum2021.azure-devices.net;DeviceId=${deviceId};SharedAccessKey=${sas};`

var Mqtt = require('azure-iot-device-mqtt').Mqtt;

var DeviceClient = require('azure-iot-device').Client

var Message = require('azure-iot-device').Message;

const readline = require('readline');

var client = DeviceClient.fromConnectionString(connectionString, Mqtt);

const rl = readline.createInterface({

    input: process.stdin,

    output: process.stdout

});

let settings = {

    "interval": 2350,

    "status": 1,

    "alarm": 0,

    "rgb": {

        "r": 255,

        "g": 255,

        "b": 255

    }

}

function setsettingslamp() {

    var message = new Message(JSON.stringify(settings))

    client.sendEvent(message, function (err) {

        if (err) {

            console.error('send error: ' + err.toString());

        } else {

            console.log('message sent');

        }

    });

};

const question1 = () => {

    return new Promise((resolve, reject) => {

        rl.question('Zet de lamp aan (1) of uit (0): ', (answer) => {

            console.log(`Je hebt het volgende ingevuld: ${answer}`)

            settings.status = answer

            resolve()

        })

    })

}

const question2 = () => {

    return new Promise((resolve, reject) => {

        rl.question('Zet de lichtsterkte tussen 0 en 255: ', (answer) => {

            console.log(`Je hebt het volgende ingevuld: ${answer}`)

            settings.rgb.r = answer

            settings.rgb.g = answer

            settings.rgb.b = answer

            resolve()

        })

    })

}

client.onDeviceMethod('confirmation', confirmation)

const main = async () => {

    await question1()

    await question2()

    rl.close()

    setsettingslamp()

}

function confirmation(request, response) {

    console.log("received confirmation from:");

    console.log(request.payload);

    console.log("Sending confirmation to service");

    request.send(200);

}

main()

**service.js:**

'use strict';

var Client = require('azure-iothub').Client;

var Message = require('azure-iot-device').Message;

const fs = require('fs')

const { EventHubConsumerClient } = require("@azure/event-hubs");

const connectionString = `Endpoint=sb://iothub-ns-delftiothu-8528485-c734b6e251.servicebus.windows.net/;EntityPath=delftiothubpracticum2021;SharedAccessKeyName=service;SharedAccessKey=7c3SGtU21s8fUjcSGMMjXsO5sebBaPu2ix9Zg1Pj/I4=`;

const cconnectionstring = "HostName=DelftIotHubPracticum2021.azure-devices.net;SharedAccessKeyName=service;SharedAccessKey=7c3SGtU21s8fUjcSGMMjXsO5sebBaPu2ix9Zg1Pj/I4="

var client = Client.fromConnectionString(cconnectionstring);

client.open();

var printError = function (err) {

    console.log(err.message);

};

const devices = ["device-student-22-device-1", "device-student-22-device-2", "device-student-22-device-3","device-student-22-device-4","device-student-22-device-5" ]

  var printMessages = function (messages) {

    for (const message of messages) {

      if(devices.includes(message.systemProperties["iothub-connection-device-id"])){

        if(message.systemProperties["iothub-connection-device-id"] === 'device-student-22-device-1'){

            lampbeweging(message.body)

        }

      }

    }

  };

  async function main() {

    console.log("IoT Hub Quickstarts - Read device to cloud messages.");

    const clientOptions = {

    };

    const consumerclient = new EventHubConsumerClient("9", connectionString, clientOptions);

    consumerclient.subscribe({

      processEvents: printMessages,

      processError: printError,

    });

  }

  main().catch((error) => {

    console.error("Error running sample:", error);

  });

function lampbeweging(settings){

  var deviceId = 'device-student-22-device-3';

  var methodParams = {

      methodName: 'setlamp',

      payload: settings,

      responseTimeoutInSeconds: 30

  }

  sendMessage(deviceId, methodParams, () => messageToControl({Id:deviceId, calledFunction: methodParams.methodName}));

}

function messageToControl(deviceinfo) {

  var deviceId = 'device-student-22-device-1';

  var methodParams = {

    methodName: 'confirmation',

    payload: {"lampId": deviceinfo.Id, "calledFunction": deviceinfo.calledFunction},

    responseTimeoutInSeconds: 30

  };

  sendMessage(deviceId, methodParams)

}

function sendMessage(deviceId, methodParams, successMethod = () => {}) {

  client.invokeDeviceMethod(deviceId, methodParams, (err, result) => {

    if(err) {

      console.log("failed to send message to " + deviceId + " with function " + methodParams.methodName);

    } else {

      console.log("Succesfully send message to " + deviceId + " with function " + methodParams.methodName);

      console.log(methodParams.payload);

      successMethod();

    }

  });

}

Vraag 7:

Ik heb mijn code van de vorige opdracht klein beetje aangepast zodat berichten sturen dynamischer ging en ik niet afhankelijk was van hard coded variabelen. De twee lampen zijn vrijwel iedentiek aan elkaar alleen de device id’s zijn anders.

**Lamps.js:**

'use strict';

const deviceId = "device-student-22-device-3";

const sas = "c79+me18yA0j2rqdBKNjNC+SEBZ8uV0CJdTe+pE6o7E=";

const connectionString = `HostName=DelftIotHubPracticum2021.azure-devices.net;DeviceId=${deviceId};SharedAccessKey=${sas};`

var Mqtt = require('azure-iot-device-mqtt').Mqtt;

var DeviceClient = require('azure-iot-device').Client

var Message = require('azure-iot-device').Message;

const chalk = require('chalk');

var client = DeviceClient.fromConnectionString(connectionString, Mqtt);

let lampsettings = {

  "interval" : 2350,

  "status": 1,

  "alarm" : 0,

  "rgb":{

      "r":255,

      "g":255,

      "b":255

  }

}

console.log(chalk.rgb(lampsettings.rgb.r, lampsettings.rgb.g, lampsettings.rgb.b).underline('This is a simulated dimmable bulb'));

lampsettings.status = 0

lampsettings.rgb.r = 128;

lampsettings.rgb.g = 128;

lampsettings.rgb.b = 128;

function print() {

    //console.clear();

    if (lampsettings.status == 0) {

        console.log(chalk.rgb(0, 0, 0).underline('Light is switched off'));

    }

    else {

        console.log(chalk.rgb(lampsettings.rgb.r, lampsettings.rgb.g, lampsettings.rgb.b)('This is a simulated bulb'));

    }

}

function updatestatus(request, response){

    function dmr (err) {

        if (err) {

            console.error(chalk.red('An error ocurred when sending a method response:\n' + err.toString()));

        } else {

            console.log(chalk.green('Response to method \'' + request.methodName + '\' sent successfully.'));

            print();

        }

    }

    if (typeof request.payload == 'undefined' && request.payload) {

        console.log(chalk.red('Invalid interval response received in payload'));

        // Report failure back to your hub.

        response.send(400, 'Invalid direct method parameter: ' + JSON.stringify(request.payload), dmr);

    } else {

        Object.assign(lampsettings, request.payload);

        response.send(200, JSON.stringify(request.methodName), dmr);

    }

}

client.onDeviceMethod('setlamp', updatestatus)

**Lampcontrol.js:**

'use strict';

const deviceId = "device-student-22-device-1";

const sas = "ff2aanu0rv7UfoGB1b8NFT2Sp0qRlbJcS/t314PBtd0=";

const connectionString = `HostName=DelftIotHubPracticum2021.azure-devices.net;DeviceId=${deviceId};SharedAccessKey=${sas};`

var Mqtt = require('azure-iot-device-mqtt').Mqtt;

var DeviceClient = require('azure-iot-device').Client

var Message = require('azure-iot-device').Message;

const readline = require('readline');

var client = DeviceClient.fromConnectionString(connectionString, Mqtt);

const rl = readline.createInterface({

  input: process.stdin,

  output: process.stdout

});

let settings = {

  "interval" : 2350,

  "status": 1,

  "alarm" : 0,

  "rgb":{

      "r":255,

      "g":255,

      "b":255

  }

}

function setsettingslamp(){

  var message = new Message(JSON.stringify(settings))

  client.sendEvent(message, function (err) {

    if (err) {

      console.error('send error: ' + err.toString());

    } else {

      console.log('message sent');

    }

  });

};

const question1 = () => {

  return new Promise((resolve, reject) => {

    rl.question('Zet de lamp aan (1) of uit (0): ', (answer) => {

      console.log(`Je hebt het volgende ingevuld: ${answer}`)

      settings.status = answer

      resolve()

    })

  })

}

const question2 = () => {

  return new Promise((resolve, reject) => {

    rl.question('Zet de lichtsterkte tussen 0 en 255: ', (answer) => {

      console.log(`Je hebt het volgende ingevuld: ${answer}`)

      settings.rgb.r = answer

      settings.rgb.g = answer

      settings.rgb.b = answer

      resolve()

    })

  })

}

client.onDeviceMethod('confirmation', confirmation)

const main = async () => {

  // await question1()

  // await question2()

  // rl.close()

  setsettingslamp()

}

function confirmation(request, response) {

  console.log("received confirmation from:");

  console.log(request.payload);

  console.log("Sending confirmation to service");

  request.send(200);

}

main()

**Service.js:**

'use strict';

var Client = require('azure-iothub').Client;

var Message = require('azure-iot-device').Message;

const fs = require('fs')

const { EventHubConsumerClient } = require("@azure/event-hubs");

const connectionString = `Endpoint=sb://iothub-ns-delftiothu-8528485-c734b6e251.servicebus.windows.net/;EntityPath=delftiothubpracticum2021;SharedAccessKeyName=service;SharedAccessKey=7c3SGtU21s8fUjcSGMMjXsO5sebBaPu2ix9Zg1Pj/I4=`;

const cconnectionstring = "HostName=DelftIotHubPracticum2021.azure-devices.net;SharedAccessKeyName=service;SharedAccessKey=7c3SGtU21s8fUjcSGMMjXsO5sebBaPu2ix9Zg1Pj/I4="

var client = Client.fromConnectionString(cconnectionstring);

client.open();

var printError = function (err) {

    console.log(err.message);

};

const devices = ["device-student-22-device-1", "device-student-22-device-2", "device-student-22-device-3", "device-student-22-device-4", "device-student-22-device-5"]

const lamps = ["device-student-22-device-3", "device-student-22-device-4"];

const bewegingsensoren = ["device-student-22-device-1", "device-student-22-device-2"];

var printMessages = function (messages) {

    for (const message of messages) {

        let id = message.systemProperties["iothub-connection-device-id"];

        if (devices.includes(id)) {

            if (bewegingsensoren.includes(id)) {

                lampbeweging(id, message.body)

            }

        }

    }

};

async function main() {

    console.log("IoT Hub Quickstarts - Read device to cloud messages.");

    const clientOptions = {

    };

    const consumerclient = new EventHubConsumerClient("9", connectionString, clientOptions);

    consumerclient.subscribe({

        processEvents: printMessages,

        processError: printError,

    });

}

main().catch((error) => {

    console.error("Error running sample:", error);

});

function lampbeweging(id, payload) {

    var methodParams = {

        methodName: 'setlamp',

        payload: payload,

        responseTimeoutInSeconds: 30

    }

    lamps.forEach(lamp => {

        sendMessage(lamp, methodParams);

    });

}

function messageToControl(id, payload) {

    var methodParams = {

        methodName: 'confirmation',

        payload: { "lampId": payload.Id, "calledFunction": payload.calledFunction },

        responseTimeoutInSeconds: 30

    };

    sendMessage(id, methodParams);

}

function sendMessage(deviceId, methodParams) {

    return client.invokeDeviceMethod(deviceId, methodParams, (err, result) => {

        if (err) {

            console.log("failed to send message to " + deviceId + " with function " + methodParams.methodName);

            console.error(err.message);

        } else {

            console.log("Succesfully send message to " + deviceId + " with function " + methodParams.methodName);

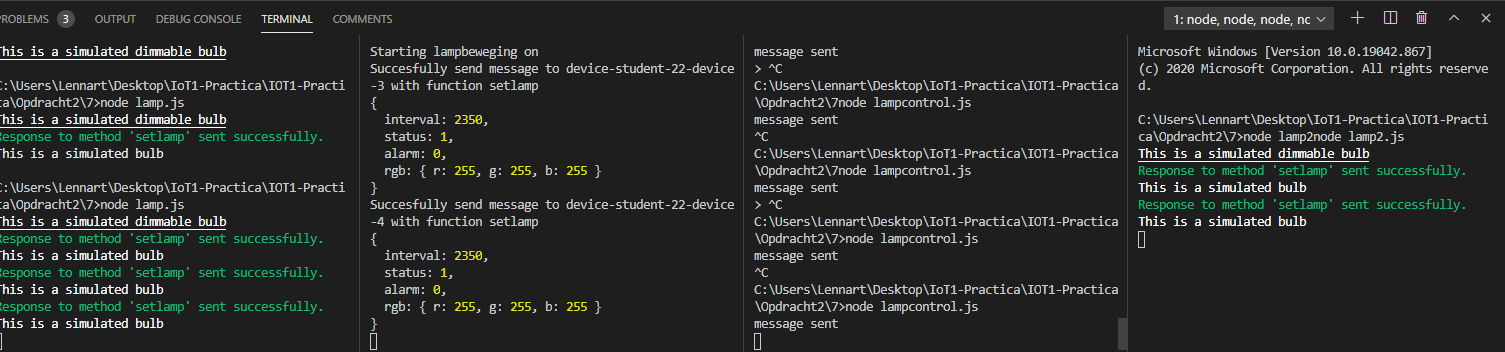
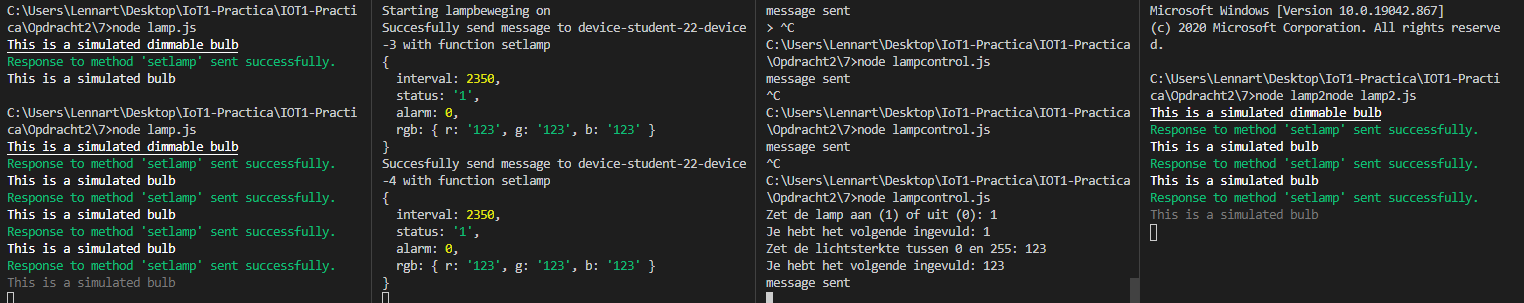
            console.log(methodParams.payload);

        }

    });

}

**Uitkomst in console**

** **

**Vraag 8:**

In lampcontrol wordt nu maar 1 vraag gesteld, waarna een andere functie een bericht stuurt naar iothub met de info welk apparaat uitgelezen moet worden. In service zijn enkele aanpassingen gemaakt om de nieuwe informatie uit de message body te kunnen verwerken en daarna met een direct method wordt informatie opgevraagd. Deze direct method is veranderd omdat na de sendmessage() direct een nieuwe direct method wordt aangeroepen die de informatie doorspeelt naar lampcontrol. Hiervoor moest de functie sendmessage worden aangepast om de result payload te kunnen gebruiken van de direct method naar de lamp.

**Lampcontrol:**

'use strict';

const deviceId = "device-student-22-device-1";

const sas = "ff2aanu0rv7UfoGB1b8NFT2Sp0qRlbJcS/t314PBtd0=";

const connectionString = `HostName=DelftIotHubPracticum2021.azure-devices.net;DeviceId=${deviceId};SharedAccessKey=${sas};`

var Mqtt = require('azure-iot-device-mqtt').Mqtt;

var DeviceClient = require('azure-iot-device').Client

var Message = require('azure-iot-device').Message;

const readline = require('readline');

var client = DeviceClient.fromConnectionString(connectionString, Mqtt);

const rl = readline.createInterface({

    input: process.stdin,

    output: process.stdout

});

let settings = {

    "interval": 2350,

    "status": 1,

    "alarm": 0,

    "rgb": {

        "r": 255,

        "g": 255,

        "b": 255

    }

}

//I ADDED THIS

function getLampSettings(id) {

  var message = new Message(JSON.stringify("device-student-22-device-" + id))

    console.log("opvragen gegevens van device: device-student-22-device-"+ id)

    send(message)

}

//ADDED THIS

function send(message) {

  client.sendEvent(message, function (err) {

        if (err) {

            console.error('send error: ' + err.toString());

        } else {

            console.log('message sent');

      }

    });

}

//MINIFIED THIS

function setLampSettings() {

    var message = new Message(JSON.stringify(settings))

    send(message);

};

//CHANGED THIS

const question1 = () => {

    return new Promise((resolve, reject) => {

        rl.question('Voer lamp ID in om status te checken: ', (answer) => {

            console.log(`Je hebt het volgende ingevuld: ${answer}`)

            getLampSettings(answer);

            resolve()

        })

    })

}

client.onDeviceMethod('confirmation', confirmation)

const main = async () => {

    await question1()

    rl.close()

}

function confirmation(request, response) {

    console.log("received confirmation from:");

    if(request.payload.status) {

        console.log("de lamp staat aan");

    } else {

        console.log("De lamp staat uit");

    }

    console.log("Sending confirmation to service");

    response.send(200);

}

main()

**service:**

'use strict';

var Client = require('azure-iothub').Client;

var Message = require('azure-iot-device').Message;

const fs = require('fs')

const { EventHubConsumerClient } = require("@azure/event-hubs");

const connectionString = `Endpoint=sb://iothub-ns-delftiothu-8528485-c734b6e251.servicebus.windows.net/;EntityPath=delftiothubpracticum2021;SharedAccessKeyName=service;SharedAccessKey=7c3SGtU21s8fUjcSGMMjXsO5sebBaPu2ix9Zg1Pj/I4=`;

const cconnectionstring = "HostName=DelftIotHubPracticum2021.azure-devices.net;SharedAccessKeyName=service;SharedAccessKey=7c3SGtU21s8fUjcSGMMjXsO5sebBaPu2ix9Zg1Pj/I4="

var client = Client.fromConnectionString(cconnectionstring);

client.open();

var printError = function (err) {

    console.log(err.message);

};

const devices = ["device-student-22-device-1", "device-student-22-device-2", "device-student-22-device-3", "device-student-22-device-4", "device-student-22-device-5"]

const lamps = ["device-student-22-device-3", "device-student-22-device-4"];

const bewegingsensoren = ["device-student-22-device-1", "device-student-22-device-2"];

var printMessages = function (messages) {

    for (const message of messages) {

        let id = message.systemProperties["iothub-connection-device-id"];

        if (devices.includes(id)) {

            if (bewegingsensoren.includes(id)) {

                lampbeweging(id, message.body)

            }

        }

    }

};

async function main() {

    console.log("IoT Hub Quickstarts - Read device to cloud messages.");

    const clientOptions = {

    };

    const consumerclient = new EventHubConsumerClient("9", connectionString, clientOptions);

    consumerclient.subscribe({

        processEvents: printMessages,

        processError: printError,

    });

}

main().catch((error) => {

    console.error("Error running sample:", error);

});

function lampbeweging(id, payload) {

    var methodParams = {

        methodName: 'getlamp',

        payload: payload,

        responseTimeoutInSeconds: 30

    }

    sendMessage(payload, methodParams, (result) => messageToControl(id, result));

}

function messageToControl(id, payload) {

    var methodParams = {

        methodName: 'confirmation',

        payload: payload,

        responseTimeoutInSeconds: 30

    };

    sendMessage(id, methodParams);

}

function sendMessage(deviceId, methodParams, func = (result) => {console.log(result)}) {

    return client.invokeDeviceMethod(deviceId, methodParams, (err, result) => {

        if (err) {

            console.log("failed to send message to " + deviceId + " with function " + methodParams.methodName);

            console.error(err.message);

        } else {

            console.log("Succesfully send message to " + deviceId + " with function " + methodParams.methodName);

            console.log(methodParams.payload);

            func(result.payload);

        }

    });

}

**Lamp 1 ( staat aan):**

'use strict';

const deviceId = "device-student-22-device-3";

const sas = "c79+me18yA0j2rqdBKNjNC+SEBZ8uV0CJdTe+pE6o7E=";

const connectionString = `HostName=DelftIotHubPracticum2021.azure-devices.net;DeviceId=${deviceId};SharedAccessKey=${sas};`

var Mqtt = require('azure-iot-device-mqtt').Mqtt;

var DeviceClient = require('azure-iot-device').Client

var Message = require('azure-iot-device').Message;

const chalk = require('chalk');

var client = DeviceClient.fromConnectionString(connectionString, Mqtt);

let lampsettings = {

  "interval" : 2350,

  "status": 1,

  "alarm" : 0,

  "rgb":{

      "r":255,

      "g":255,

      "b":255

  }

}

console.log(chalk.rgb(lampsettings.rgb.r, lampsettings.rgb.g, lampsettings.rgb.b).underline('This is a simulated dimmable bulb'));

function print() {

    //console.clear();

    if (lampsettings.status == 0) {

        console.log(chalk.rgb(0, 0, 0).underline('Light is switched off'));

    }

    else {

        console.log(chalk.rgb(lampsettings.rgb.r, lampsettings.rgb.g, lampsettings.rgb.b)('This is a simulated bulb'));

    }

}

function updatestatus(request, response){

    function dmr (err) {

        if (err) {

            console.error(chalk.red('An error ocurred when sending a method response:\n' + err.toString()));

        } else {

            console.log(chalk.green('Response to method \'' + request.methodName + '\' sent successfully.'));

            print();

        }

    }

    if (typeof request.payload == 'undefined' && request.payload) {

        console.log(chalk.red('Invalid interval response received in payload'));

        // Report failure back to your hub.

        response.send(400, 'Invalid direct method parameter: ' + JSON.stringify(request.payload), dmr);

    } else {

        response.send(200, lampsettings, dmr);

    }

}

// CHANGED THIS

client.onDeviceMethod('getlamp', updatestatus)

**lamp 2 ( staat uit)**

'use strict';

const deviceId = "device-student-22-device-4";

const sas = "a2Sh1qCEZ9mpPOxgrdzu4OiwcNptIPKKM9UyaL+UtOM=";

const connectionString = `HostName=DelftIotHubPracticum2021.azure-devices.net;DeviceId=${deviceId};SharedAccessKey=${sas};`

var Mqtt = require('azure-iot-device-mqtt').Mqtt;

var DeviceClient = require('azure-iot-device').Client

var Message = require('azure-iot-device').Message;

const chalk = require('chalk');

var client = DeviceClient.fromConnectionString(connectionString, Mqtt);

let lampsettings = {

  "interval" : 2350,

  "status": 0,

  "alarm" : 0,

  "rgb":{

      "r":255,

      "g":255,

      "b":255

  }

}

console.log(chalk.rgb(lampsettings.rgb.r, lampsettings.rgb.g, lampsettings.rgb.b).underline('This is a simulated dimmable bulb'));

lampsettings.status = 0

lampsettings.rgb.r = 128;

lampsettings.rgb.g = 128;

lampsettings.rgb.b = 128;

function print() {

    //console.clear();

    if (lampsettings.status == 0) {

        console.log(chalk.rgb(0, 0, 0).underline('Light is switched off'));

    }

    else {

        console.log(chalk.rgb(lampsettings.rgb.r, lampsettings.rgb.g, lampsettings.rgb.b)('This is a simulated bulb'));

    }

}

function updatestatus(request, response){

    function dmr (err) {

        if (err) {

            console.error(chalk.red('An error ocurred when sending a method response:\n' + err.toString()));

        } else {

            console.log(chalk.green('Response to method \'' + request.methodName + '\' sent successfully.'));

            print();

        }

    }

    if (typeof request.payload == 'undefined' && request.payload) {

        console.log(chalk.red('Invalid interval response received in payload'));

        // Report failure back to your hub.

        response.send(400, 'Invalid direct method parameter: ' + JSON.stringify(request.payload), dmr);

    } else {

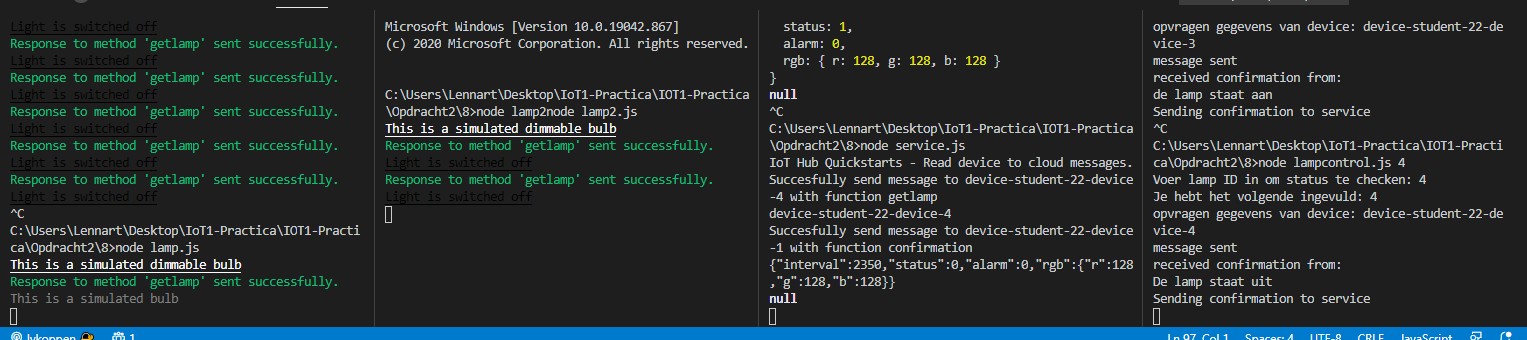
        response.send(200, JSON.stringify(lampsettings), dmr);

    }

}

client.onDeviceMethod('getlamp', updatestatus)

**console output:**

****