1. Registreer een device via de Azure dasboard of cli. Hierdoor krijg je een connection string

| Device ID | Type | Status | Last status update | Authentication type | C2D messages queued | Tags |
|---|---|---|---|---|---|---|
| first_device | IoT Device | Enabled | -- | Shared Access Signature | 0 | |

2. De Azure Iot Explorer hoef je niet te installeren. Alles kan je ook doen via portal.azure.com. Hierin zit ook een CLI voor de az commando's

   done

3. Maak een Iot device die telemetry verstuurt. Gebruik uit de voorbeeld code van git: device/samples/javascript/simple_sample_device.js. Deze code is simpeler dan: pnp_temperature_controller.js. Vergeet voor het runnen niet de npm install voor de packages

```
// Copyright (c) Microsoft. All rights reserved.
// Licensed under the MIT license. See LICENSE file in the project root for full
license information.

'use strict';

require('dotenv').config({path: './simple_env.env'});

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


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

const deviceConnectionString = process.env.IOTHUB_DEVICE_CONNECTION_STRING;
let sendInterval;

function disconnectHandler() {
    clearInterval(sendInterval);
    sendInterval = null;
    client.open().catch((err) => {
        console.error(err.message);
    });
}

function messageHandler(msg) {
    console.log('Id: ' + msg.messageId + ' Body: ' + msg.data);
    client.complete(msg, printResultFor('completed'));
}

function generateMessage() {
    const windSpeed = 10 + (Math.random() * 4); // range: [10, 14]
    const temperature = 20 + (Math.random() * 10); // range: [20, 30]
    const humidity = 60 + (Math.random() * 20); // range: [60, 80]
    const data = JSON.stringify({ deviceId: 'myFirstDevice', windSpeed: windSpeed,
temperature: temperature, humidity: humidity });
```

```
    const message = new Message(data);
    message.properties.add('temperatureAlert', (temperature > 28) ? 'true' :
'false');
    return message;
}

function errorHandler(err) {
    console.error(err.message);
}

function connectHandler() {
    console.log('Client connected');
    // Create a message and send it to the IoT Hub every two seconds
    if (!sendInterval) {
        sendInterval = setInterval(() => {
            const message = generateMessage();
            console.log('Sending message: ' + message.getData());
            client.sendEvent(message, printResultFor('send'));
        }, 2000);
    }
}

// fromConnectionString must specify a transport constructor, coming from any
transport package.
let client = Client.fromConnectionString(deviceConnectionString, Protocol);

client.on('connect', connectHandler);
client.on('error', errorHandler);
client.on('disconnect', disconnectHandler);
client.on('message', messageHandler);

client.open()
    .catch((err) => {
        console.error('Could not connect: ' + err.message);
    });

// Helper function to print results in the console
function printResultFor(op) {
    return function printResult(err, res) {
        if (err) console.log(op + ' error: ' + err.toString());
        if (res) console.log(op + ' status: ' + res.constructor.name);
    };
}
```

p.s. de npm package azure-iot-device-mqtt is deprecated en heeft memory leaks.

3. Controleer of er berichten aankomen op de Iot Hub: Gebruik hier voor het CLI commando. Noteer dit commando ook. Welke informatie ontvang je? En van welk device is deze afkomstig. Is deze informatie consistent?

```
PS C:\Users\iwanv\Documents\School\iot-cloud\opdracht3> node .\simple_sample_device.js
Client connected
Sending message: {"deviceId":"myFirstDevice","windSpeed":11.911307746267827,"temperature":28.225515439203896,"humidity":73.70340354418275}
send status: MessageEnqueued
```

```
Starting event monitor, use ctrl-c to stop...
{
    "event": {
        "origin": "first_device",
        "module": "",
        "interface": "",
        "component": "",
        "payload": "{\"deviceId\":\"myFirstDevice\",\"windSpeed\":11.911307746267827,\"temperature\":28.225515439203896,\"humidity\":73.70340354418275}"
    }
}
```

4. Welke 4 events kan je krijgen uit de client class? En welke zijn er gebruikt bij de test uit de vorige vraag?

```
connect
error
disconnect
message
```

Bij de vorige opdracht is connect gebruikt.

5. Pas het bericht dat je verstuurd aan zodat er ook het tijdstip van de meting in staat. En test of dit goed werkt.

```
function generateMessage() {
    const windSpeed = 10 + (Math.random() * 4); // range: [10, 14]
    const temperature = 20 + (Math.random() * 10); // range: [20, 30]
    const humidity = 60 + (Math.random() * 20); // range: [60, 80]
    const time = new Date().toLocaleTimeString(); // <---------------- Time added
here
    const data = JSON.stringify({ deviceId: 'myFirstDevice', windSpeed: windSpeed,
temperature: temperature, humidity: humidity, time : time });
    const message = new Message(data);
    message.properties.add('temperatureAlert', (temperature > 28) ? 'true' :
'false');
    return message;
}
```

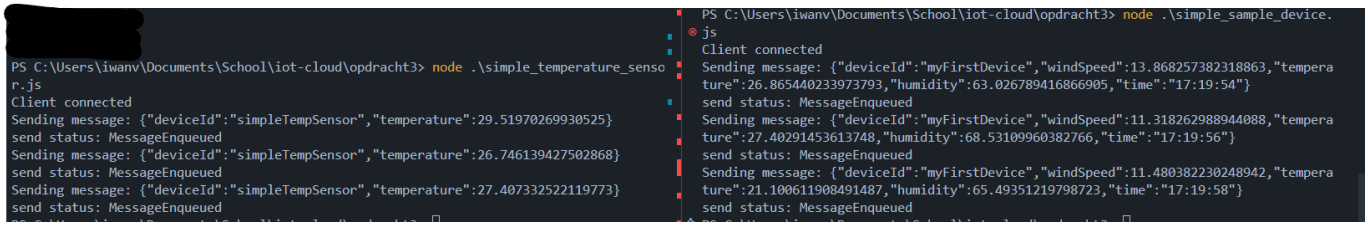
```
PS C:\Users\iwanv\Documents\School\iot-cloud\opdracht3> node .\simple_sample_device.js
Client connected
Sending message: {"deviceId":"myFirstDevice","windSpeed":13.467093103030805,"temperature":22.2784037084455,"humidity":78.83749744540646,"time":"17:06:48"}
send status: MessageEnqueued
PS C:\Users\iwanv\Documents\School\iot-cloud\opdracht3>
```

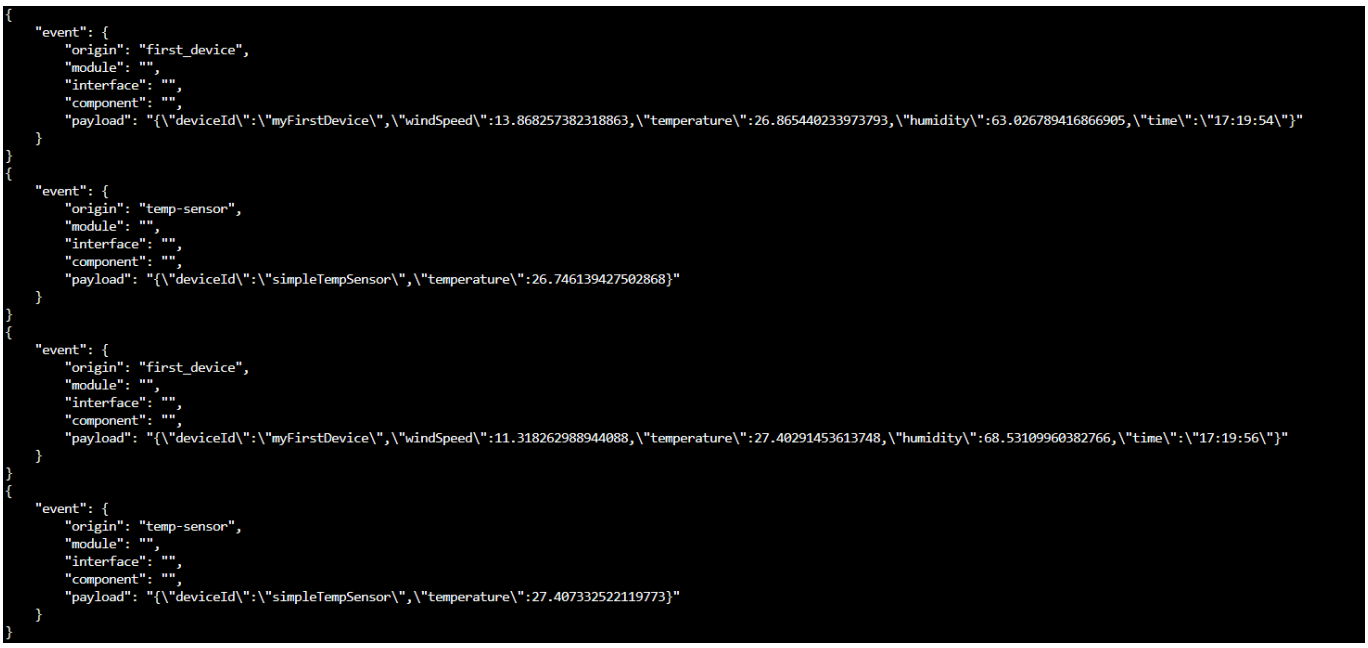
```
{
    "event": {
        "origin": "first_device",
        "module": "",
        "interface": "",
        "component": "",
        "payload": "{\"deviceId\":\"myFirstDevice\",\"windSpeed\":13.467093103030805,\"temperature\":22.2784037084455,\"humidity\":78.83749744540646,\"time\":\"17:06:48\"}"
    }
}
```

6. Maak een 2e sensor die alleen de temperatuur verstuurt in JSON. Haal alle code uit het oorspronkelijke sample weg die niet nodig is (dode code) . Test nu beide sensors.

nieuwe temperatuur sensor:

```
'use strict';

require('dotenv').config({ path: './simple_env.env' });

// Choose a protocol by uncommenting one of these transports.
const Protocol = require('azure-iot-device-mqtt').Mqtt;
const Client = require('azure-iot-device').Client;
const Message = require('azure-iot-device').Message;

const deviceConnectionString = process.env.IOTHUB_TEMPSENSOR_CONNECTION_STRING;
let sendInterval;

function generateMessage() {
    const temperature = 20 + (Math.random() * 10); // range: [20, 30]
    const data = JSON.stringify({ deviceId: 'simpleTempSensor', temperature:
temperature, });
    const message = new Message(data);
    message.properties.add('temperatureAlert', (temperature > 28) ? 'true' :
'false');
    return message;
}

function connectHandler() {
    console.log('Client connected');
    // Create a message and send it to the IoT Hub every two seconds
    if (!sendInterval) {
        sendInterval = setInterval(() => {
            const message = generateMessage();
            console.log('Sending message: ' + message.getData());
            client.sendEvent(message, printResultFor('send'));
        }, 2000);
    }
}

// fromConnectionString must specify a transport constructor, coming from any
transport package.
let client = Client.fromConnectionString(deviceConnectionString, Protocol);

client.on('connect', connectHandler);

client.open()
    .catch((err) => {
        console.error('Could not connect: ' + err.message);
    });

// Helper function to print results in the console
function printResultFor(op) {
    return function printResult(err, res) {
        if (err) console.log(op + ' error: ' + err.toString());
        if (res) console.log(op + ' status: ' + res.constructor.name);
    };
}
```




7. Als je kijkt in de reference bij de Message class dan zie je dat je ook kan aangeven dat er JSON wordt verstuurd. Pas de code van de 1e sensor aan en test het effect.

```
function generateMessage() {
    const windSpeed = 10 + (Math.random() * 4); // range: [10, 14]
    const temperature = 20 + (Math.random() * 10); // range: [20, 30]
    const humidity = 60 + (Math.random() * 20); // range: [60, 80]
    const time = new Date().toLocaleTimeString();    const data = JSON.stringify({
deviceId: 'myFirstDevice', windSpeed: windSpeed, temperature: temperature,
humidity: humidity, time : time });
    const message = new Message(data);
    message.contentType='application/json';
    message.properties.add('temperatureAlert', (temperature > 28) ? 'true' :
'false');
    return message;
}
```

8. Kopieer devices_methods.js. Dit is een simulated Iot Deur device

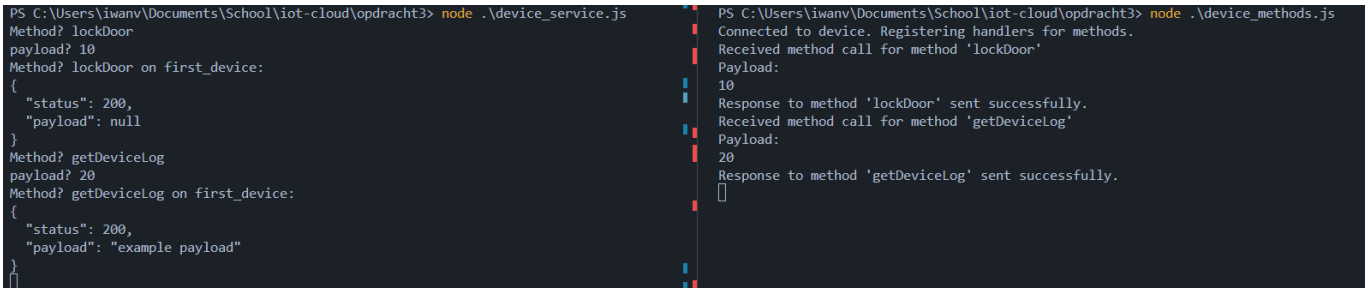
9. Welke 2 direct methodes biedt dit script aan?

   'lockDoor' en 'getDeviceLog'

10. Maak een service die deze direct methods aan roept (baseer je op (device_method.js uit de service sdk samples). Blijf voor de connection de environment variabele gebruiken (ivm security). Test dat je de Iot deur via deze service kan openen en sluiten

```
'use strict';


require('dotenv').config({ path: './simple_env.env' });

const readline = require('readline');

// Set up readline to listen for key presses
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

var answer1;
var answer2;

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

var connectionString = process.env.IOTHUB_CONNECTION_STRING;
var targetDevice = process.env.IOTHUB_DEVICE_ID;
var methodParams //= {
//     methodName: process.env.IOTHUB_METHOD_NAME,
//     payload: process.env.IOTHUB_METHOD_PAYLOAD,
//     responseTimeoutInSeconds: 15 // set response timeout as 15 seconds
// };


let index = 0;

const questions = ["Method? ", "payload? "];

// ask question and read in input
function askQuestion() {
    if (index >= questions.length) {
        // create methodParams
        methodParams = {
            methodName: answer1,
            payload: answer2,
            responseTimeoutInSeconds: 15 // set response timeout as 15 seconds
        };

        client.invokeDeviceMethod(targetDevice, methodParams, function (err,
result) {
            if (err) {
                console.error('Failed to invoke method \'' +
methodParams.methodName + '\': ' + err.message);
            } else {
                console.log(methodParams.methodName + ' on ' + targetDevice +
':');
                console.log(JSON.stringify(result, null, 2));

                methodParams = {
                    methodName: null,
```

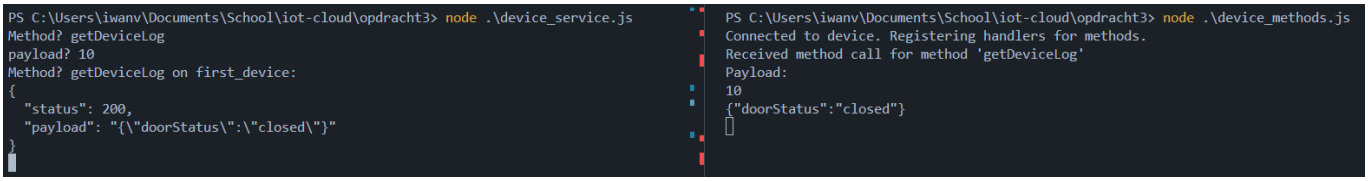
```
                    payload: null,
                    responseTimeoutInSeconds: 15 // set response timeout as 15
seconds
                };
            }
        });

        // set index back to 0 to keep repeating the questions
        index = 0;
    }

    rl.question(questions[index], handleInput);
    index++;
}

function handleInput(answer) {
    if (answer == "quit") {
        process.exit(0);
    }
    if (index == 1) {
        if (answer == 'getDeviceLog' || answer == 'lockDoor')
            answer1 = answer;
    }
    if (index == 2) {
        answer2 = answer;
    }
    askQuestion();
}

askQuestion();

var client = Client.fromConnectionString(connectionString);
```




11. Pas de code van de Iot deur aan zodat je kan opvragen of de deur open of dicht is. Deze response ontvang je als JSON

```
function onGetDeviceLog(request, response) {
    printDeviceMethodRequest(request);

    var responseMessage;

    if (open) {
        responseMessage = JSON.stringify({ doorStatus: 'open' });
```

```
    }
    else {
        responseMessage = JSON.stringify({ doorStatus: 'closed' });
    }

    // complete the response
    response.send(200, responseMessage, function (err) {
        if (err) {
            console.error('An error ocurred when sending a method response:\n' +
                err.toString());
        } else {
            console.log(responseMessage);
        }
```



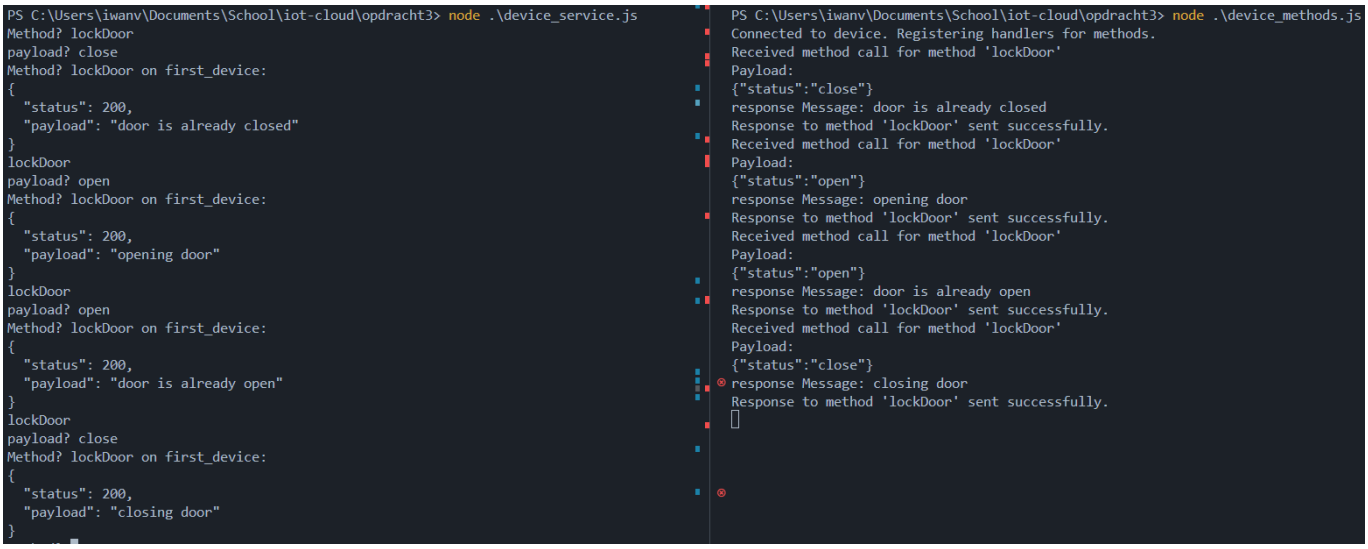

12. Pas de code zo aan dat je via 1 direct method de deur kan openen of sluiten. De parameter geeft aan of de deur gesloten moet worden of geopend. Ook deze parameter is JSON. Ook geeft de deur een antwoord: deur geopend, of ik was al geopend, deur gesloten enz.

Service kant:

```
function handleInput(answer) {
    if (answer == "quit") {
        process.exit(0);
    }
    if (index == 1) {
        if (answer == 'getDeviceLog' || answer == 'lockDoor')
            answer1 = answer;
    }
    if (index == 2) {
        if (answer1 == 'lockDoor') {
            if (answer == 'open' || answer == 'close') {
                answer2 = JSON.stringify( {'status' : answer} );
            }
            else {
                console.log("enter either open or close");
                index--;
            }
        }
        else
            answer2 = answer;
    }
    askQuestion();
}
```

device kant:

```
function onLockDoor(request, response) {
    printDeviceMethodRequest(request);

    var responseMessage = "";

    var parsedMessage = JSON.parse(request.payload);

    // if (request.payload.includes('open')) {
    if(parsedMessage.status == 'open') {
        if (open) {
            responseMessage = "door is already open";
        }
        else {
            responseMessage = "opening door";
            open = true;
        }
    }
    else if(parsedMessage.status == 'close') {
        if (!open) {
            responseMessage = "door is already closed";
        }
        else {
            responseMessage = "closing door";
            open = false;
        }
    }

    console.log("response Message: " + responseMessage)


    // complete the response
    response.send(200, responseMessage, function (err) {
        if (err) {
            console.error('An error ocurred when sending a method response:\n' +
                err.toString());
        } else {
            console.log('Response to method \'' + request.methodName +
                '\' sent successfully.');
        }
    });
}
```



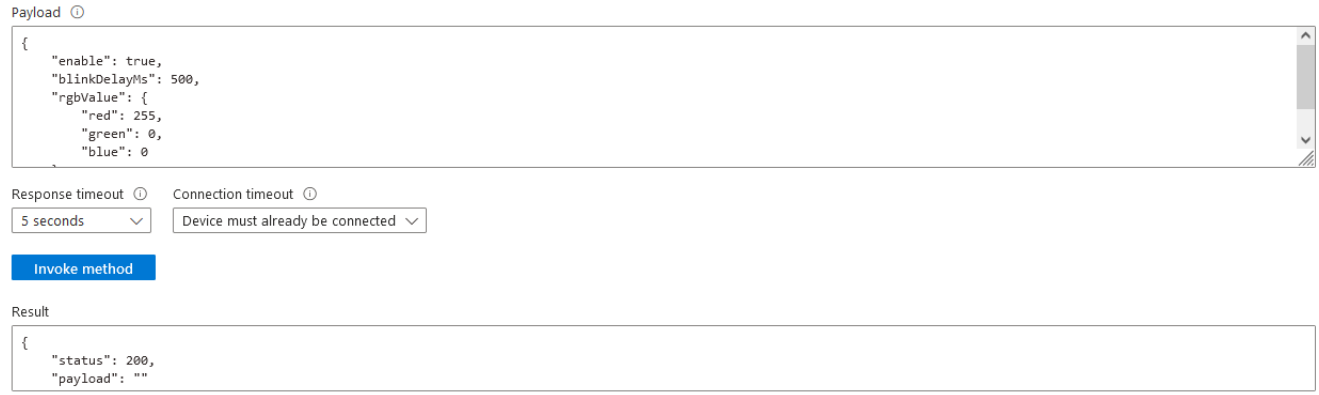

13. Maak een simulated alarmlicht dat je kan aanzetten met een bepaalde knipperfrequentie (in msec) en kleur (RGB waarde) die verbonden is met de Azure Iot Hub. Functioneel doet deze hetzelfde als het alarmlicht uit practicum 2. Ook mag in dit simulated device geen dode code zitten

```
'use strict';
require('dotenv').config({ path: './simple_env.env' });

const Protocol = require('azure-iot-device-mqtt').Mqtt;
const Client = require('azure-iot-device').Client;
let client = null;
let intervalId;
const chalk = require('chalk');

let blinkDelay = 1000;
let rgbValue = [0, 255, 0]
var toggle = false;

function main() {
    // open a connection to the device
    const deviceConnectionString = process.env.IOTHUB_DEVICE_CONNECTION_STRING;
    client = Client.fromConnectionString(deviceConnectionString, Protocol);
    // eslint-disable-next-line security/detect-non-literal-fs-filename
    client.open(onConnect);
}

function onConnect(err) {
    if (err) {
        console.error('Could not connect: ' + err.message);
    } else {
        console.log('Connected to device. Registering handlers for methods.');

        // register handlers for all the method names we are interested in
        client.onDeviceMethod('toggleEffect', onToggleEffect);
    }
}

function print() {
```

```
    process.stdout.cursorTo(0);
    if (toggle) {
        process.stdout.write("   ");
    }
    else {
        process.stdout.write(chalk.rgb(rgbValue[0], rgbValue[1], rgbValue[2])
("aan"));
    }
    toggle = (!toggle);
}

function printDeviceMethodRequest(request) {
    // print method name
    console.log('Received method call for method \'' + request.methodName + '\'');

    // if there's a payload just do a default console log on it
    if (request.payload) {
        console.log('Payload:\n' + request.payload);
    }
}


function onToggleEffect(request, response) {
    printDeviceMethodRequest(request);

    var responseMessage = "";
    clearInterval(intervalId);
    intervalId = null;

    if (request.payload.enable) {
        const enable = request.payload.enable;
        console.log(`Enable: ${enable}`);

        if (!enable)
            return;
    }
    else {
        console.log("error: enabled not found.");
        return;
    }

    if (request.payload.blinkDelayMs) {
        blinkDelay = request.payload.blinkDelayMs;
        console.log(`Blink Delay (ms): ${blinkDelay}`);
    }
    else {
        console.log("error: blinkDelayMs not found.")
        return;
    }

    if (request.payload.rgbValue) {
        const rgb = request.payload.rgbValue;
        console.log(`RGB Value ${rgb}`);
        rgbValue[0] = rgb['red'];
```

```
        rgbValue[1] = rgb['green'];
        rgbValue[2] = rgb['blue'];
        console.log(`RGB Value - Red: ${rgbValue[0]}, Green: ${rgbValue[1]}, Blue:
${rgbValue[2]}`);
    }
    else {
        console.log("error: rgbValue not found.")
        return;
    }

    intervalId = setInterval(print, blinkDelay);

    // complete the response
    response.send(200, responseMessage, function (err) {
        if (err) {
            console.error('An error ocurred when sending a method response:\n' +
                err.toString());
        } else {
            console.log(responseMessage);
        }
    });

}

// program

main();

process.stdout.write("aan");
toggle = false;
intervalId = setInterval(print, 1000);
```

```
PS C:\Users\iwanv\Documents\School\iot-cloud\opdracht3> node .\alarmLicht.js
aanConnected to device. Registering handlers for methods.
aanReceived method call for method 'toggleEffect'
Payload:
[object Object]
Enable: true
Blink Delay (ms): 500
RGB Value [object Object]
RGB Value - Red: 255, Green: 0, Blue: 0

aan
```

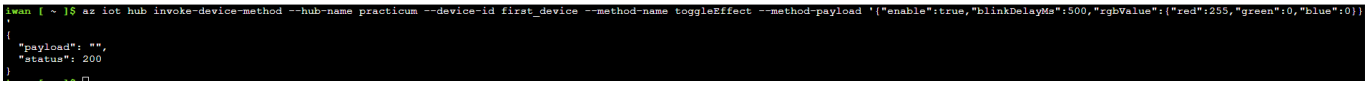



14. Je kan een Iot Device ook via de CLI testen. Je hoeft dan geen backend te hebben. Zie CLI reference: https://docs.microsoft.com/en-us/cli/azure/iot/hub?view=azure-cli-latest#az-iot-hub-invoke-device-method Test zowel de deur als het alarmlicht

```
PS C:\Users\iwanv\Documents\School\iot-cloud\opdracht3> node .\device_methods.js
Connected to device. Registering handlers for methods.
Received method call for method 'lockDoor'
Payload:
[object Object]
response Message: opening door
Response to method 'lockDoor' sent successfully.
```

```
iwan [ ~ ]$ az iot hub invoke-device-method --hub-name practicum  --device-id first_device --method-name lockDoor --method-payload '{"status":"open"}'
{
  "payload": "opening door",
  "status": 200
}
```

```
PS C:\Users\iwanv\Documents\School\iot-cloud\opdracht3> node .\alarmLicht.js
aanConnected to device. Registering handlers for methods.
   Received method call for method 'toggleEffect'
Payload:
[object Object]
Enable: true
Blink Delay (ms): 500
RGB Value [object Object]
RGB Value - Red: 255, Green: 0, Blue: 0

aan
```




15. Laat jouw de Iot deur ook testen door een medestudent. Hij moet nu dus verbinding kunnen maken met jouw Iot Hub door het runnen van de service die eerder gemaakt hebt. Bedenk welke gegevens je nog meer moet delen.

```
Method? lockDoor
payload? open
Method? lockDoor on first_device:
{
  "status": 200,
  "payload": ""
}
```

```
Received method call for method 'lockDoor'
Payload:
{"status":"open"}
response Message:
Response to method 'lockDoor' sent successfully.
```

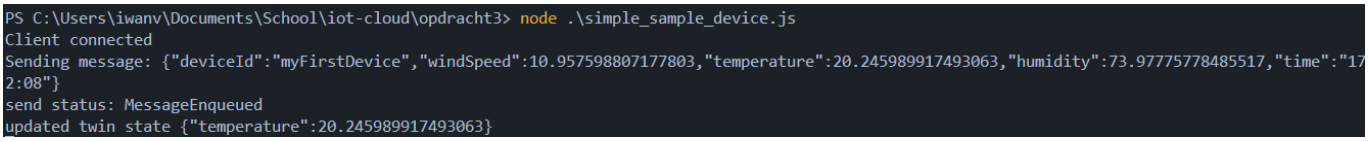
16. Open in de Azure console de twin properties. Welke info kan je hierin zien?

    informatie over het device, zoals het id, metadata zoals wanneer het voor het laatst is gepdate of het device aan staat. Ook laat het connectie statys zien en wanneer het voor het laatst actief was. Hieronder de twin properties van first_device.


```
{
"etag": "AAAAAAAAAAE=",
"deviceId": "first_device",
"deviceEtag": "MjE2Mjc5MzU=",
"version": 2,
"properties": {
    "desired": {
        "$metadata": {
            "$lastUpdated": "2024-09-16T13:13:13.2672548Z"
```

```
            },
            "$version": 1
        },
        "reported": {
            "$metadata": {
                "$lastUpdated": "2024-09-16T13:13:13.2672548Z"
            },
            "$version": 1
        }
    },
    "capabilities": {
        "iotEdge": false
    },
    "modelId": "",
    "status": "enabled",
    "statusUpdateTime": "0001-01-01T00:00:00.0000000Z",
    "lastActivityTime": "2024-09-16T15:19:40.6565584Z",
    "connectionState": "Connected",
    "cloudToDeviceMessageCount": 0,
    "authenticationType": "sas",
    "x509Thumbprint": {}
}
```


17. Demonstreer de reported of desired sectie van de twin door een Iot device te maken die deze gebruikt. Bedenk hiervoor een zinvolle twin property in een eerder door jou gemaakt Iot Device.





```
{
    "etag": "AAAAAAAAAAE=",
    "deviceId": "first_device",
    "deviceEtag": "MjE2Mjc5MzU=",
    "version": 8,
    "properties": {
        "desired": {
            "$metadata": {
                "$lastUpdated": "2024-09-16T13:13:13.2672548Z"
            },
            "$version": 1
        },
        "reported": {
            "$metadata": {
                "$lastUpdated": "2024-09-17T15:01:57.4918101Z",
                "temperature": {
                    "$lastUpdated": "2024-09-17T15:01:57.4918101Z"
                }
            },
            "$version": 7,
            "temperature": 26.023898937638855
```

```
        }
    },
    "capabilities": {
        "iotEdge": false
    },
    "modelId": "",
    "status": "enabled",
    "statusUpdateTime": "0001-01-01T00:00:00.0000000Z",
    "lastActivityTime": "2024-09-17T15:01:14.3841694Z",
    "connectionState": "Disconnected",
    "cloudToDeviceMessageCount": 0,
    "authenticationType": "sas",
    "x509Thumbprint": {}
}
```

code:

```
function generateTwinProperty(temperature) {
    client.getTwin(function (err, twin) {

        const twinProperty = { "temperature": temperature };
        twin.properties.reported.update(twinProperty, function (err) {
            if (err) throw err;
            console.log('updated twin state ' + JSON.stringify(twinProperty));
        });
    });
}


function generateMessage() {
    const windSpeed = 10 + (Math.random() * 4); // range: [10, 14]
    const temperature = 20 + (Math.random() * 10); // range: [20, 30]
    const humidity = 60 + (Math.random() * 20); // range: [60, 80]
    const time = new Date().toLocaleTimeString(); const data = JSON.stringify({
deviceId: 'myFirstDevice', windSpeed: windSpeed, temperature: temperature,
humidity: humidity, time: time });
    generateTwinProperty(temperature);
    const message = new Message(data);
    message.contentType = 'application/json';
    message.properties.add('temperatureAlert', (temperature > 28) ? 'true' :
'false');
    return message;
}
```