

Nishava, Inc.
Handed out: 01/02/2018

Deep Azure@McKesson
Due by 11:00 PM CST on Saturday, 2/03/2018

Final Project, Assignment 15

Azure DataBox IOT Emulator
(Node.JS / Native-React)

Student: Francois TURI

This is the cookbook to build the DataBox emulator written in Node.js and also for the raspberry part in Native-React on top of a Node.js host.

Table of Contents

Nishava, Inc. Deep Azure@McKesson	1
Handed out: 01/02/2018 Due by 11:00 PM CST on Saturday, 2/03/2018	1
Final Project, Assignment 15.....	1
Azure DataBox IOT Emulator	1
(Node.JS / Native-React).....	1
Student: Francois TURI	1
Lets check the Raspbery PI	12
pseudo raspberry interpreter src>data>sample.js.....	28
src>lib>sim908.js.....	32
src>lib>sample.js.....	34
Lets go back to the Node.js installation guide:	39
Debugging not working	42
Implementing the simulator	43
Limitation.....	43
Well, lets go develop on that emulator framework.....	44
Geopositioning and geofencing is working.....	45
Azure service: Location Based Services (preview)	46
Install the code.....	48
Configure a new webserver.....	49
Simple code to connect to MQTT.....	55
Node.js framework for MQTT (express)	56
Node.js web framework	57

Build the console window.....	57
On the emulator side, we added an SdCard to store the image:.....	58
SIM908: Geolocalisation.....	60
The Simulator page: sample.js	63
The library loader sample.js.....	67
Console Part.....	74
Server.js	74
Route.js.....	76
Console.ejs.....	80
Index.ejs.....	81
The good, the bad & the ugly	82
The bad: Node.js	82
The ugly: native-react.....	85

Instructions

You must select a technical topic related to Azure's Cloud Computing Services from the list of proposed topics or propose your own topic and meet requirements as outlined below to demonstrate main features of the selected technology. This final project is an individual effort. Topics are selected in an on-line auction. The auction allows 4 individuals to sign up for each topic. The auction is on the first come, first served basis. For this auction, we are using an online tool called Signup Genius. You will receive an email with login to this web site where you will be able to select your topic. Expect to get an email from Signup Genius this coming Tuesday (January 02, 2018) before 12 noon CST with an invitation to sign up for a topic. The invitation email will be sent to your email address you have registered for this class. If you do not get an email by 2PM CST – please contact us through Canvas or a private message to instructors Piazza.

If you are proposing a topic on your own, please send a private message through Piazza. Your topic if approved will be added to the list of public topics. The teaching staff does not provide guidance on any topic. You must do the research yourself. Once you sign up for a topic, the topic is yours.

The AUCTION WILL START on Wednesday, January 3rd, 2018 at 12:00 noon CST.

The purpose of the final project is for each one of us to learn about a Microsoft Azure's Service offering, framework or API that was not covered in class and then help your colleagues gain the same knowledge. Write your final project report as a tutorial or a learning tool. All project materials including all code and the data will be made available to the entire class. Please do not include any proprietary code or code you do not want to share.

You may propose your own topic but this must be approved by the professor. Do not submit a project from your work. Topics may be related to a domain of interest (Life Sciences, Medical, Financial, Aerospace, etc.) rather than a specific technology but its implementation must in some critical way utilize Azure features. Again, please do not include sensitive or proprietary material (IP, SW code /Non-Disclosure Agreements or owned by your company).

Your project must demonstrate a Microsoft Azure related service or services and include a software demonstration using a data source. Your project must include a visualization component to present a graphical representation of results. This project is not intended to result in a research paper but rather a technology demonstration and a tutorial.

We will select a representative number of final projects (30-40) and present their short YouTube videos to the whole class. Due to the large number of students we are not able to organize public presentation of all student projects. **One or two presentations might**

take place during the class on February 06 and the rest on February 13th during the class time. All students whose videos are presented are expected to participate in person through ZOOM and answer questions from the audience. **Your short YouTube presentation will be run on a machine in the classroom.** Every short YouTube video is expected to last not (much) more than 2 minutes. Q&As should not last more than a minute. **Presentations will take place on February 13th 2018, between 6 PM and 9 PM CST. Please note the extended hours.**

Final Project Requirements:

1. Select a technology topic from the auction list via Signup Genius starting on January 3rd, 12:00 PM (noon) CST or propose your own topic.
2. Define a problem statement as to what you will solve using selected technology.
3. Select a data source (there are many data sets available online).
4. Develop/implement/code a solution for the specified problem using the data source and selected technology using a programming language of your choice: Java, Python, C, .NET, other.
5. Produce a visualization of your data/results using D3, R, Python or other JS Framework.
6. Document your work as described below.

Documentation Requirements:

- a) **Project Summary (Abstract), 1 (one) page Word document in a separate file. Project summary should start with your name, name of the course and the title of your project. Please follow the attached Template for One Page Summary.**

Produce a one page summary to describe the problem you are trying to solve and demonstrate, description and location of your data set, the particular technology or feature you are demonstrating, its uses, benefits, drawbacks, challenges and your results. Describe briefly the working example you prepared. This first page is very important. Based on that page, your colleagues will judge whether to download your project and spend time reading your documentation and code. Please add URLs of both of your YouTube videos to the bottom of the one page summary. Please provide the URL of the public GitHub repository where you will upload all of your final project's artifacts.

Note: The filename should start with the topic name followed by your name.

- b) **Demo/solution implementation and working code:**

Produce a single working demo that meets your problem statement and provides a full implementation of your solution. You may not directly copy examples (demonstrations) of the technology you found on-line. (Grade will be 0). You can use API skeletons that are provided as we have done in class and show your extended

programming use of them. **Provide neatly organized and complete working code with comments.** Please note that a project will not be given any points if a working implementation and code are missing. Please provide the URL to your data source. If your data set is larger than 10MB, PLEASE DO NOT UPLOAD your data set. In such a case, please provide only a sample of your data with the rest of your submission. Our site has a limited capacity and a few large submissions could block it.

c) **Slides:**

Produce a set of Power Point slides (10-30) with a few snapshots of your demo which captures the key points, i.e. your problem statement, what the technology does, your demonstration and pros/cons. The first page of your Power Point slide must have a standard format that we provide. Please use white background for all your Power Point slides. This makes slides readable, printable and presentable on YouTube. Place URLs of your YouTube videos on the last slide. Note: The filename should start with the topic name followed by your name

d) **Report:**

Produce a detailed document with a complete description of analyzed technology including all installation and configuration steps. Your report will start with your name and project title. **Detailed description of installation and configuration is required when appropriate. Your colleagues must be able to reproduce your work based solely on the steps you have documented.** Describe your problem statement, data set, installation/configuration, results, what worked, what did not and why not, and any lessons learned. Report must show all steps to reproduce examples that you developed. This report is similar to your homework solutions where all steps are described along with the results.

- a) The first page of your report should be the same as the one page summary (abstract). The first page should contain your name and the project name.
- b) Please include your name and your project topic in the header or the footer of your MS Word Document.
- c) Name your file like you named 1 page summary appended with _report.
- d) Please include page numbers.
- e) You are welcome to upload a PDF version of your documents. However, you must always submit an MS Word version of your report.
- f) Please include URLs for both of your YouTube videos at the end of your report
- g) Please provide the URL of the public GitHub repository where you will upload all of your final project's artifacts.

e) **15 minute YouTube presentation:**

Produce a 15 minute YouTube presentation. This video will contain a summary of the technology and details of your project and results. If you insist, you can record two or even three 15 minute presentations. Please do not record four.

f) 2 minute YouTube presentation:

Produce a 2 minute YouTube presentation. This video will contain a summary of the technology and a quick overview of your demonstration. This video might get presented to the entire class so please make sure it is 2 minutes and not more.

g) Public GitHub repository where you will upload all of your final project artifacts

Submission Instructions:

Submit the Final Project Report in PDF format and One Page Summary in MS Word format to the course site. Upload your report and slides separately to your GitHub repository. Upload a zip file that contains your software and data files to the GitHub repository. Include your visualization files: html, JavaScript, Python, R code and data file within a separate folder in the zip file. Ensure your visualization files and all dependent files run! TAs will deduct points if they have to edit your code to run it. Name your zip file: **YourLastNameYourFirstName_Final.zip**

Grading criteria:

Project Report and practical software code example	50%
PowerPoint Slides	20%
15 minute YouTube video	15%
2 minute YouTube video	10%
One page summary	5%

If you fail to provide practical software code example, you will lose at least 30% of the final project grade.

Azure Data box: approach & challenge

Obviously I cannot get an actual Data box device to use that service. But it was kind of the point when I choose the project.

What If I want to manage a physical device without having the device. Like an Xray table.

We know that these devices will probably talk IoT/mqtt, and there is IoT simulator.

So the point of this project is to be able to manage a device as close as possible than the actual data box following data box specifications and exploring this way the world of things, shadows and simulators on Azure.

Research & Deployment traces

The general idea is to create an IoT simulator of the Azure datastore device. We are more interested of the management of the device (deployment, GPS tracking) than its hability to backup effectively PB of data which has little interest in term of simulation (would be just a storage)

So the angle would be to use IoT simulators and in particular RasperyPi.

Ideally if we could simulate a device than can:

- upload a photo
- download a photo
- Geo tracking
- Status of device: At Azure, in transit, At client

Communicating with MQTtp to a server on Azure, the job will be done.

Raspberry Pi emulator: installing IDE

Lets check the Raspberry PI

We will follow the wizard on <https://azure-samples.github.io/raspberry-pi-web-simulator/#GetStarted> and see what we can do with it.

Step 1

Step 2

Step 3

Overview of Raspberry Pi Simulator

3. **Integrated console window.** You can see the output of your app.

[Next](#)

[See doc](#) [View source](#)

```

1 //
2 // IoT Hub Raspberry Pi NodeJS - Microsoft Sample Code - Copyright
3 //
4 const mqtt = require("mqtt");
5 const client = require("azure-iot-device").Client;
6 const Message = require("azure-iot-device").Message;
7 const Protocol = require("azure-iot-device-mqtt").Mqtt;
8 const BME280 = require("bme280-sensor");
9
10 const BME280_OPTIONS = {
11   i2cAddress: 1, // defaults to 1, i2cAddress // defaults
12   i2cAddress: BME280_I2C_ADDR_PRIMARY
13 };
14
15 const connectionString = "Hub device connection string";
16 const deviceId = "1";
17
18 var sendingMessage = false;
19 var messageId = 0;
20 var client, sensor;
21
22 - Function getMessage() {
23   messageId++;
24   sensor.readSensorData();
25   .then(function (data) {
26     const msg = new Message(JSON.stringify(data));

```

OK time to create an HuB

○ Step 1

● Step 2

○ Step 3

×

Prepare an Azure IoT hub and get the device connection string

add a device to your IoT hub.

3. Select the device you just created and copy the **primary key of the connection string**.

Back

Next

[See doc](#) [View source](#)

refresh

Delete

to view, create, update, and delete devices on yo

Query

SELECT * FROM devices

WHERE

optional (e.g. tag:location='US')

Execute

Filter by Device Id

DEVICE ID	STATUS
✓ myFirstDevice	enabled

Device Details

myFirstDevice

Device Twin

Message To

Device Id

myFirstDevice

Primary key

This is Primary key

Secondary key

This is Secondary key

Connection string—primary key

This is Connection string—prime

Connection string—secondary key

This is Connection string—second

Connect device to IoT Hub

Enable

Disable

On Azure

IoT hub

Microsoft

*

Name

DataBoxHub

✓

*

Pricing and scale tier

S1 - Standard

>

*

IoT Hub units ⓘ

1

*

Device-to-cloud partitions ⓘ

4 partitions

▼

*

Subscription

McKesson Deep Dive Training (7)

▼

*

Resource group ⓘ

☐ Create new

☐ Use existing

smj56gk

▼

☒ Pin to dashboard

Create

Automation options

Connection string

OK, the screenshot of the wizard is a bit old. Connection string is now inside Shared Access Policies (the Device Explorer was probably replaced by that Shared Access tab)

Shared access policies

iothubowner
DataBoxHub

Save Discard Regen key Delete

Access policy name
iothubowner

Permissions

- ☒ Registry read
- ☒ Registry write
- ☒ Service connect
- ☒ Device connect

Shared access keys

Primary key
RAx6yZ/8axyQyTo03MSzjFvxPNYRZkXUkzAC+Mh9I0=

Secondary key
tc+cA3LPglhDsnF10bxX+Qg2b4A+BkQcD8hpr9Vrxv4=

Connection string—primary key
HostName=DataBoxHub.azure-devices.net;SharedAccessK...

Connection string—secondary key
HostName=DataBoxHub.azure-devices.net;SharedAccessK...

Primary key connection string:

HostName=DataBoxHub.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=*****+Mh9I0=

Project

Step 1

Step 2

Step 3

Run the sample app on Pi web simulator

the Azure IoT hub **device connection string**.

2. Click **Run** button or type "npm start" in the console window to run the application.

Back

Got it

[See doc](#)
[View source](#)

```

_OPTION = {
1, // defaults to 1
:: BME280.BME280_DEFAULT_I2C_ADDRESS() // defaults to 0x77

tionString = '[Your IoT hub device connection string]';
l = 4;

message = false;
'd = 0;
sensor;

```

Run

Reset

Type 'npm start' t

We don't support s

We keep your chang

> []

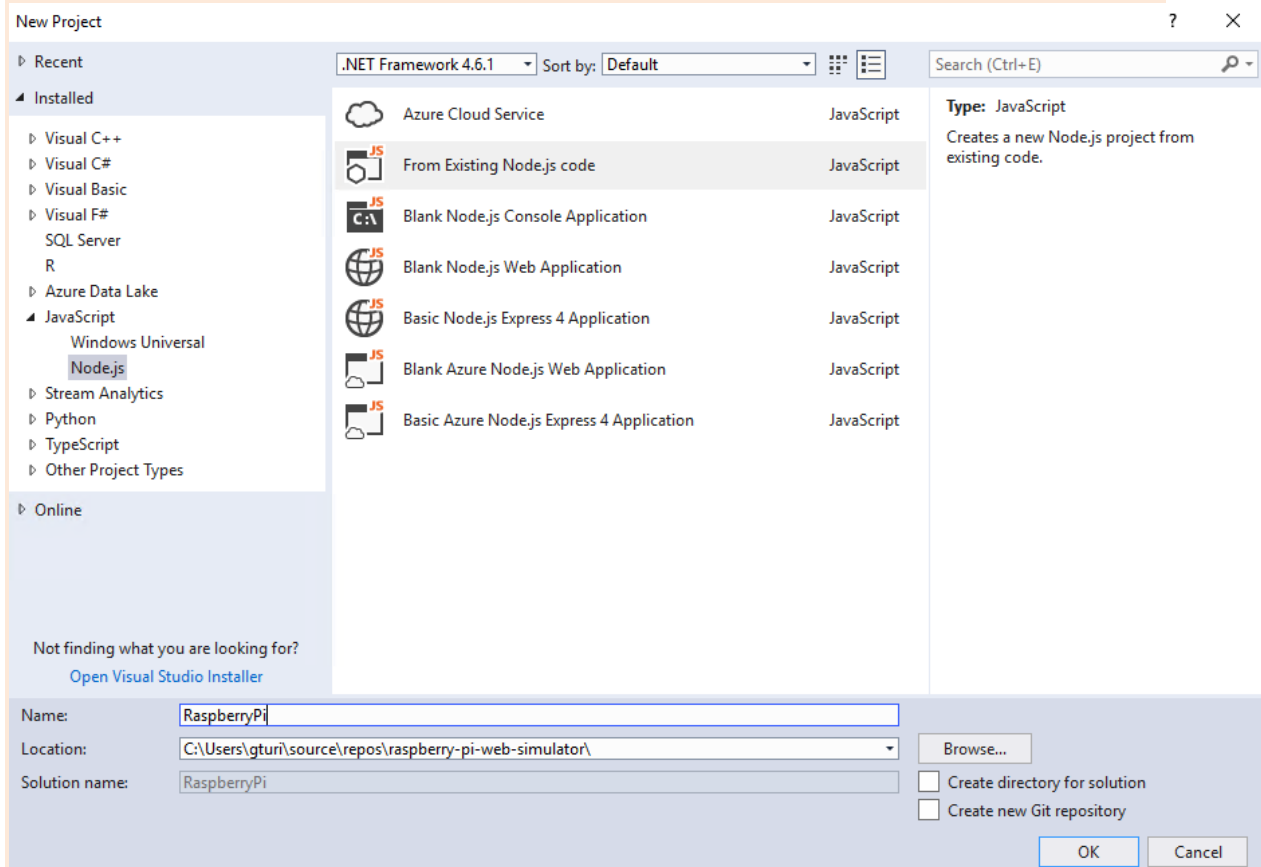
Its time to create a C# project on my windows10 vm.

```

gturi@emj56gkF1S MINGW64 ~
$ cd source/repos/GIT
gturi@emj56gkF1S MINGW64 ~/source/repos/GIT
$ git clone https://github.com/Azure-Samples/raspberry-pi-web-simulator

```


Another Node.js application. Lets hope it works better than last time



Lets try with java jre 1.8

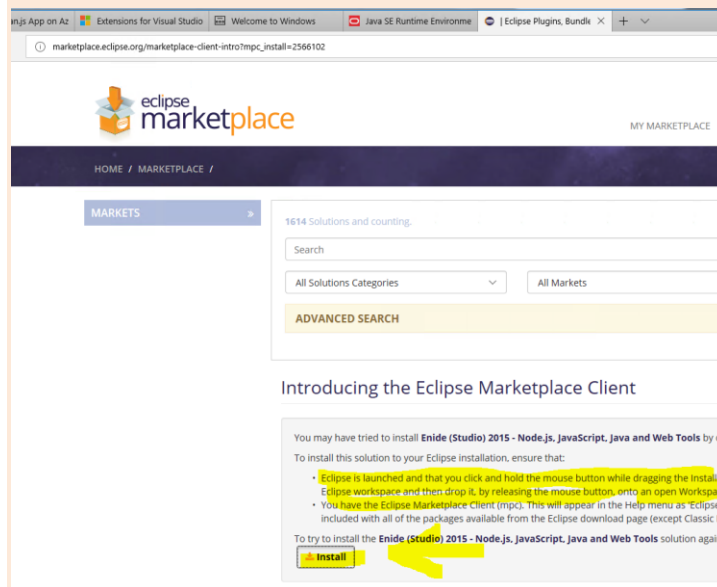
<http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>

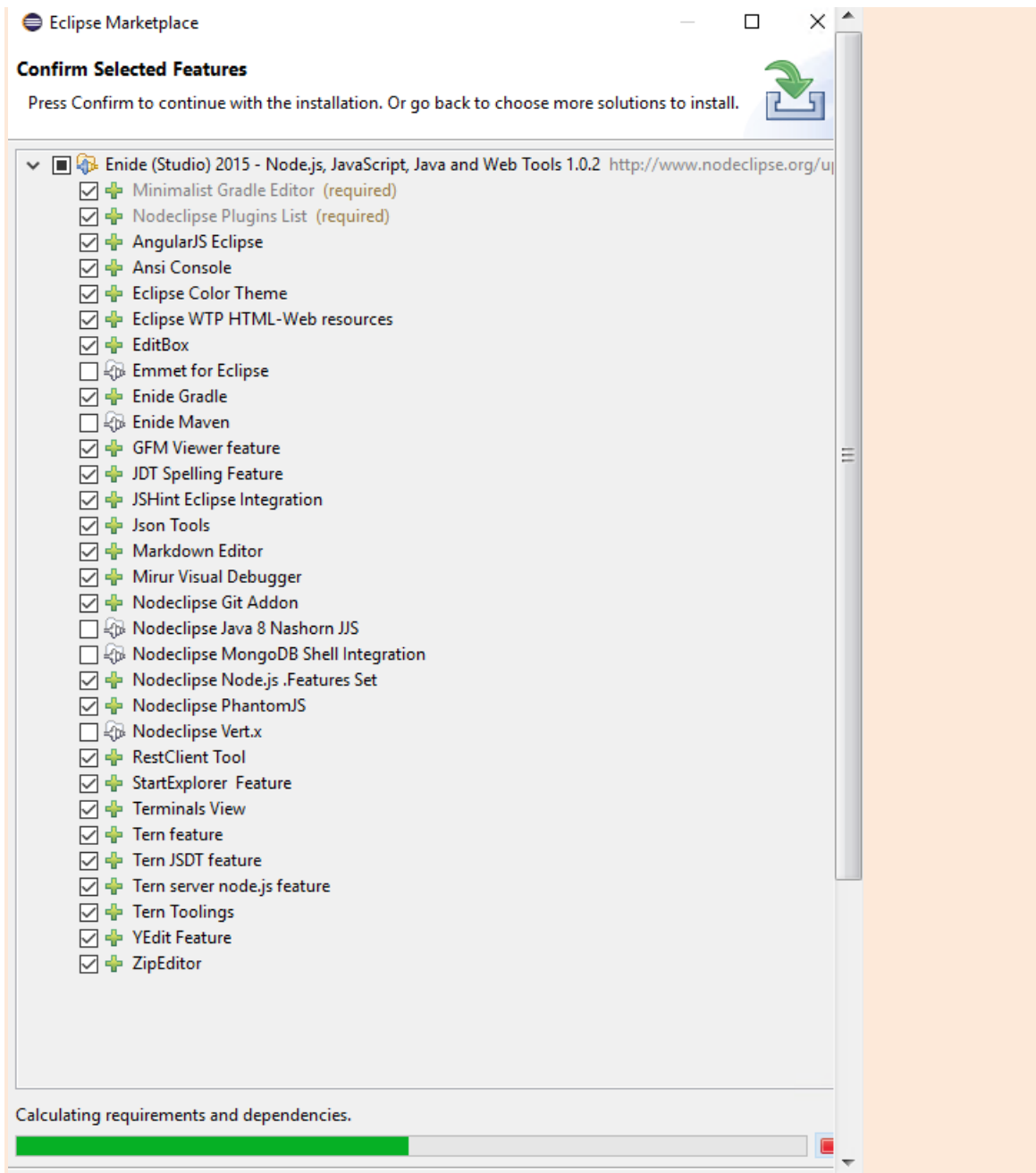
eclipse

<https://www.eclipse.org/downloads/download.php?file=/oomph/epp/oxygen/R2/eclipse-inst-win64.exe>

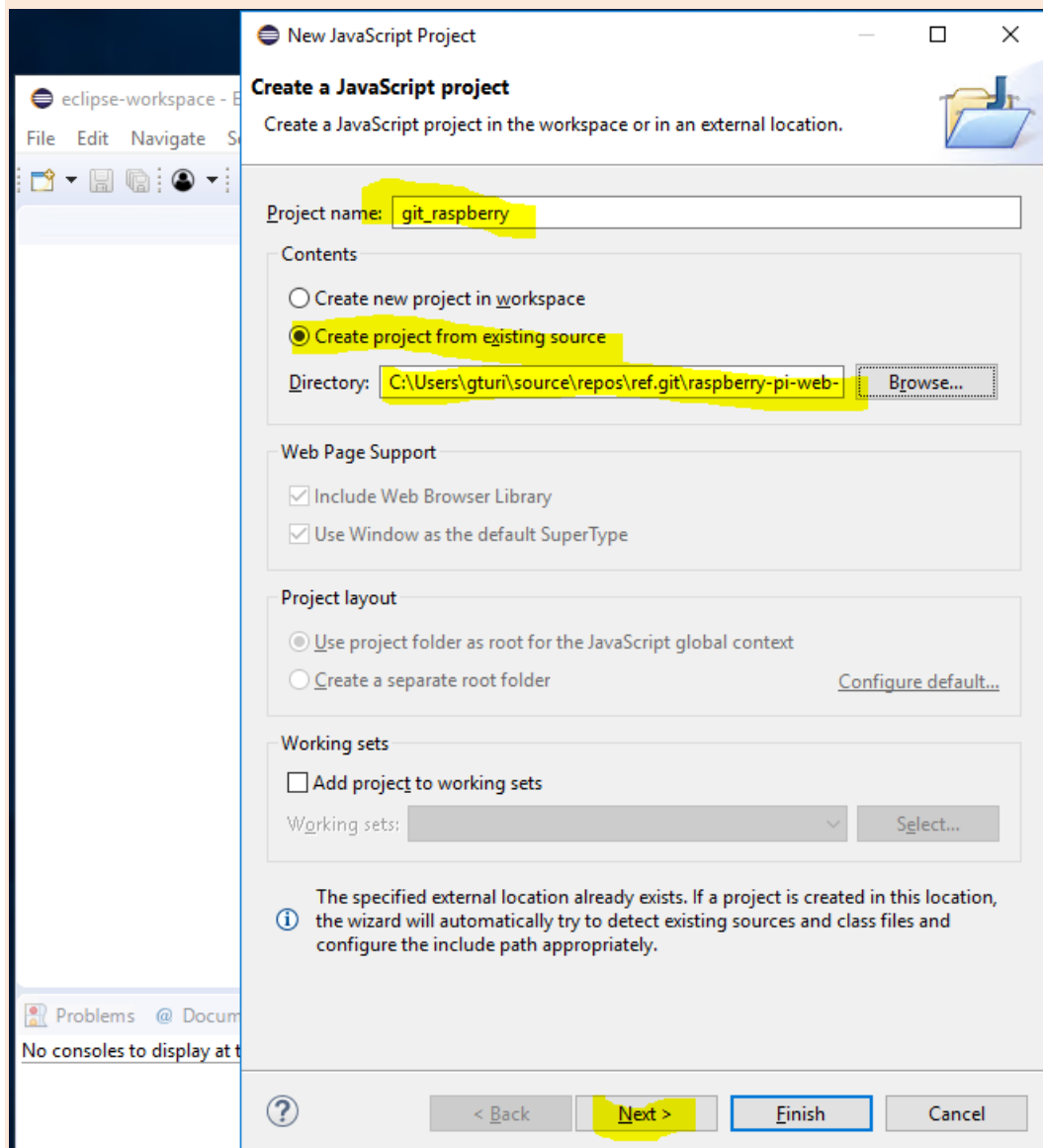
and nodeclipse

<http://www.nodeclipse.org/> We just drag this to the opened eclipse workspace

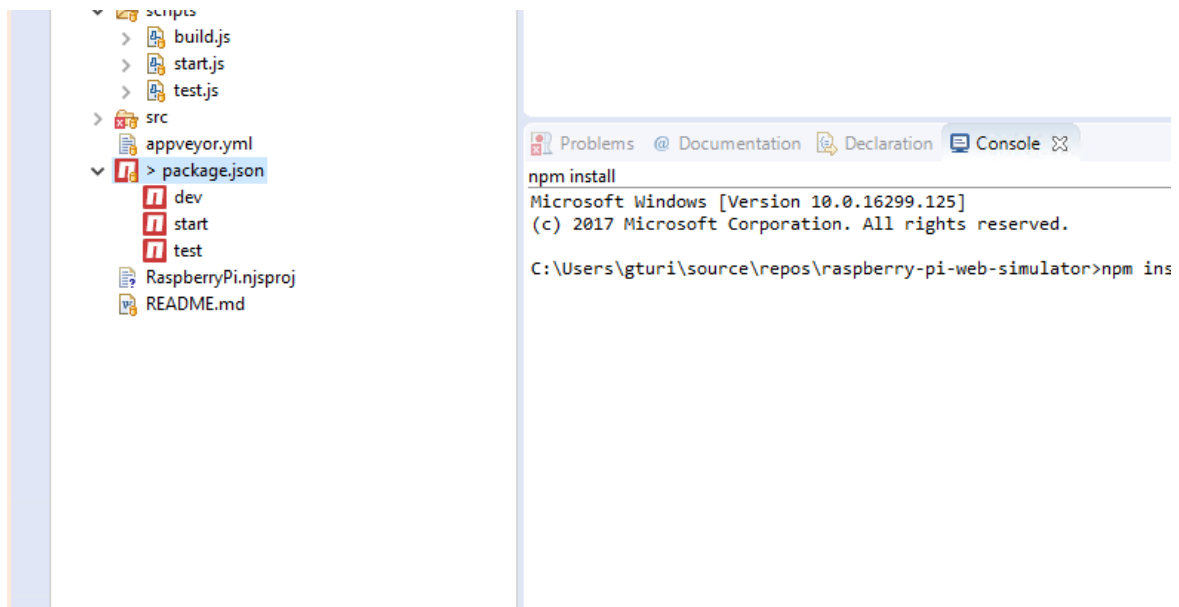




New JavaScript project

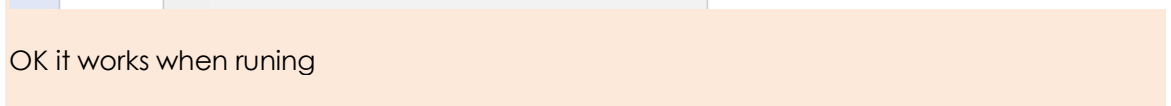


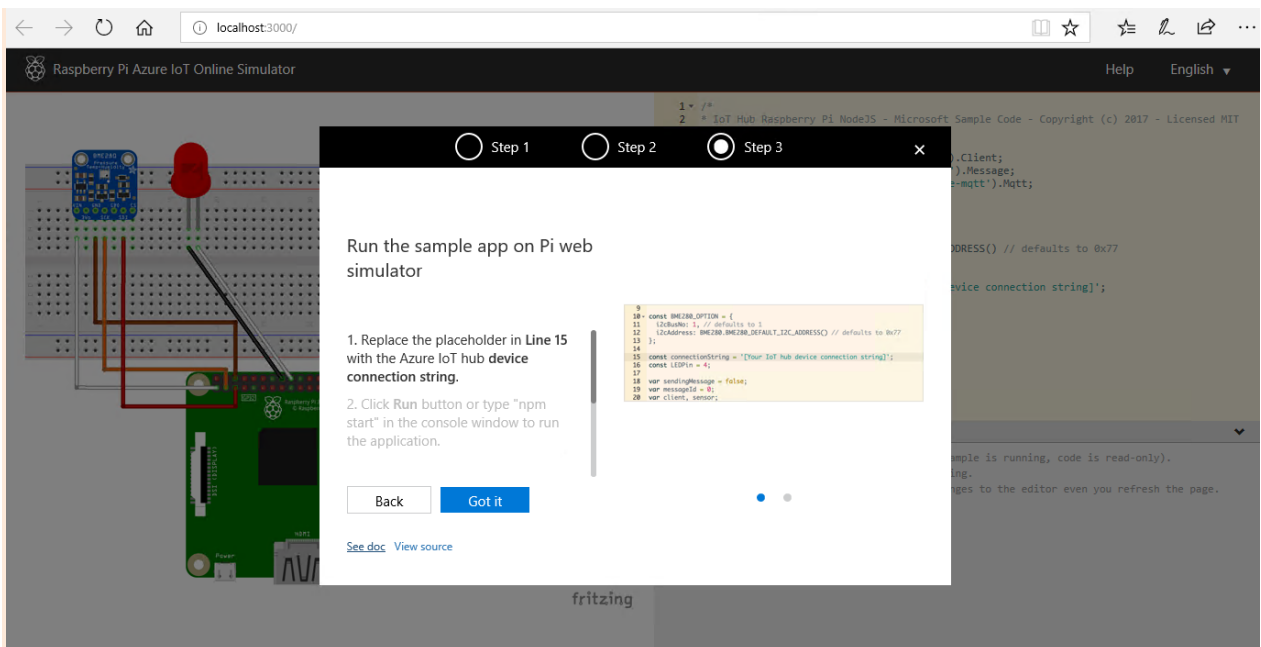
Open project and go to package.json, Right click and select
run as 4) npm install



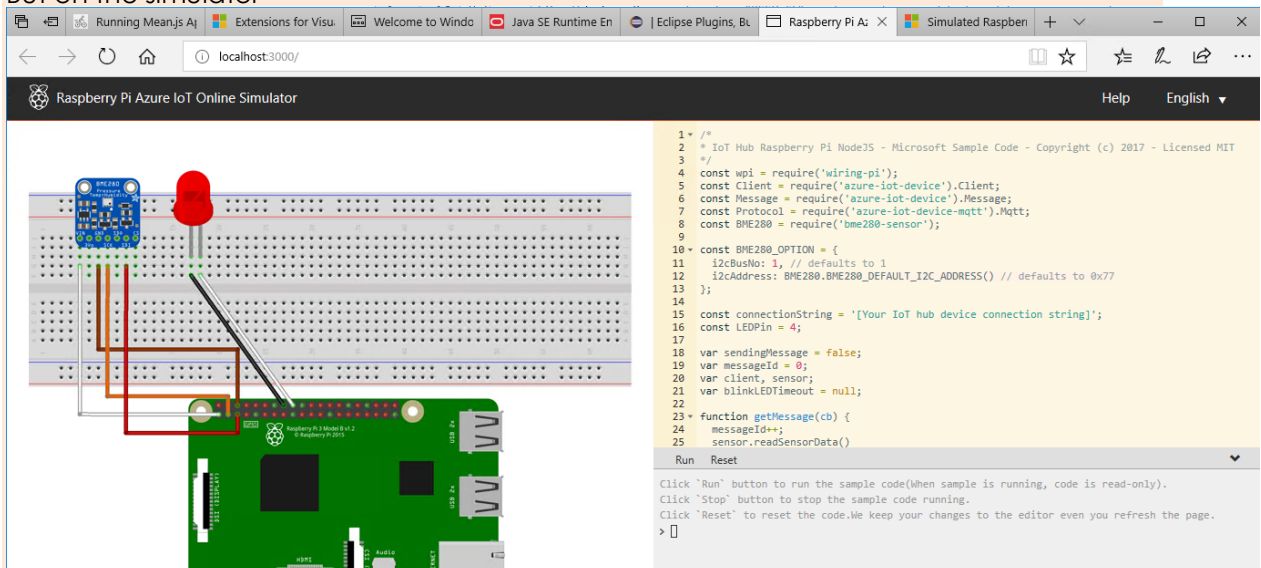
Select

Script>start.js and select :
run as node.js

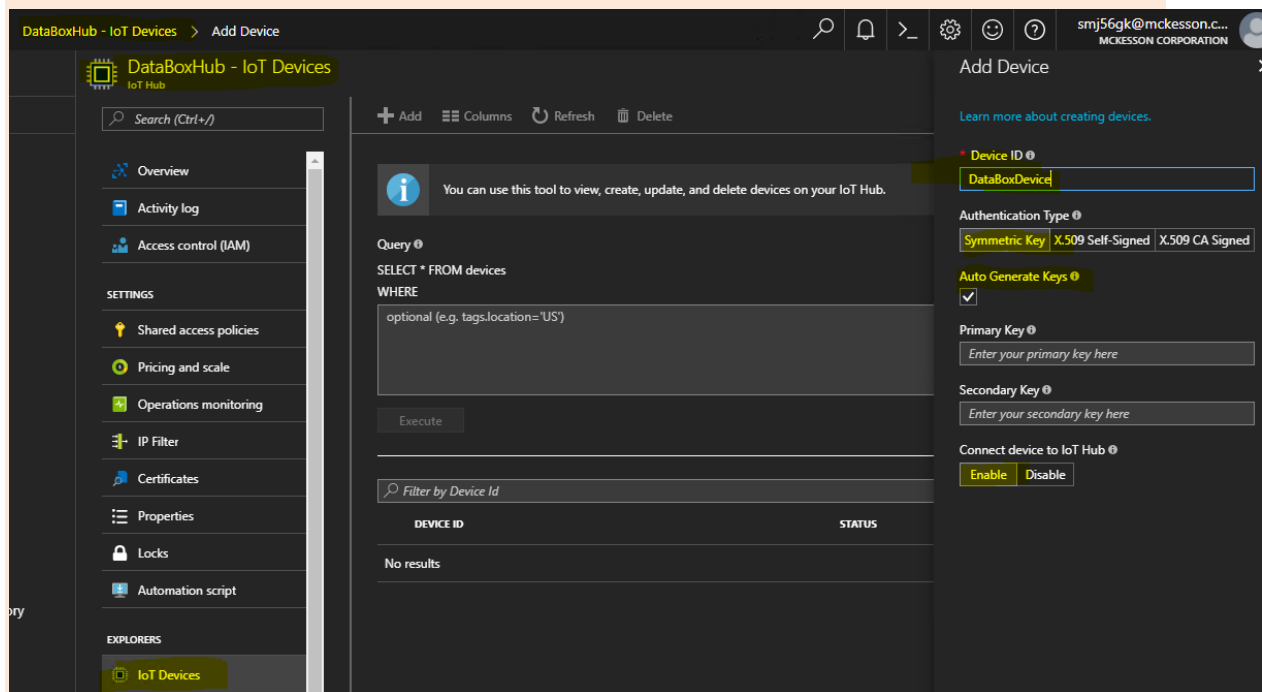




Now, the code they are talking about doesn't seem like to be on the node.js
But on the simulator



OK we need to create a device attached to the hub



Can I attach Disk to a raspberry PI

Looks like yes following: <https://blog.alexellis.io/attach-usb-storage/>

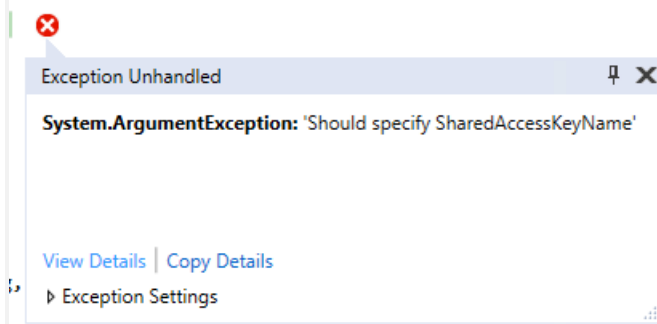
Yes, but the simulator is much more limited, a led and a sensor.

Humm....

Lets adapt Tp9 (wont be used at the end, but usefull to debug)

We don't need, yet, the C# IoT simulator. For the sake of this project, we will try to use this raspberry Node.js simulator as it as some visual.

```
namespace Tp9.device
{
    class Program
    {
        ///["{iot hub connection string}"]
        static string connectionString = "HostName=DataBoxHub.azure-
devices.net;DeviceId=DataBoxDevice;SharedAccessKey=*****+tP2y16CPGbR6kCL0oMy4=";
```



That was because the Tp9 was connecting to the hub and not directly to the device itself.

```
class Program
{
    ///["{iot hub connection string}"]
    static string connectionString = "HostName=DataBoxHub.azure-
devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=*****+Mh9IO=";
    static string iotDeviceId = "DataBoxDevice";

...
    private async static Task SendCloudToDeviceMessageAsync()
    {
        var commandMessage = new Message(Encoding.ASCII.GetBytes("Hello " +
iotDeviceId + " this is my cloud to device message.));
        commandMessage.Ack = DeliveryAcknowledgement.Full;
        try
        {
            await serviceClient.SendAsync(iotDeviceId, commandMessage);
        }
        catch (IotHubException e)
        {
            Console.WriteLine(String.Format("ERR cld 4.1 [{0}]", e.ToString()));
        }
    }
}
```

Well, we just need to point to the hub there, not the device

We recycled the cloud to device VC code. Just changed name to AppToDataBoxIoT and connection string and also the device ID which we make it appear in the code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

//Tp9
using Microsoft.ServiceBus.Messaging;
using System.Threading;
//Tp9 bidirection
using Microsoft.Azure.Devices;
using Microsoft.Azure.Devices.Common.Exceptions;

namespace Tp9.device
{
    class Program
    {
        //{"iot hub connection string}"
        static string connectionString = "HostName=DataBoxHub.azure-
devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=*****RZkXUKIZAC+Mh9I0=
";
        static string iotDeviceId = "DataBoxDevice";
        static string iotHubD2cEndpoint = "messages/events";
        static EventHubClient eventHubClient;

        //bidirection
        static ServiceClient serviceClient;

        private static async Task ReceiveMessagesFromDeviceAsync(string partition,
CancellationToken ct)
        {
            var eventHubReceiver =
eventHubClient.GetDefaultConsumerGroup().CreateReceiver(partition, DateTime.UtcNow);
            while (true)
            {
                if (ct.IsCancellationRequested)
                    break;
               EventData eventData = await eventHubReceiver.ReceiveAsync();
                if (eventData == null)
                    continue;
                string data = Encoding.UTF8.GetString(eventData.GetBytes());
                Console.WriteLine("inf cld 2.1 Message received. Partition: [{0}] Data:
[{1}]", partition, data);
            }
        }

        //bidirection
        private async static Task SendCloudToDeviceMessageAsync()
        {
            var commandMessage = new Message(Encoding.ASCII.GetBytes("Hello "+
iotDeviceId + " this is my Cloud to device message."));
            commandMessage.Ack = DeliveryAcknowledgement.Full;
            try
            {
                await serviceClient.SendAsync(iotDeviceId, commandMessage);
            }
            catch (IotHubException e)
            {
                Console.WriteLine(String.Format("ERR cld 4.1 [{0}]", e.ToString()));
            }
        }

        //Feedback
        private async static void ReceiveFeedbackAsync()
        {
            var feedbackReceiver = serviceClient.GetFeedbackReceiver();
            Console.WriteLine("\nReceiving c2d feedback from service");
        }
    }
}
```

```

        while (true)
        {
            var feedbackBatch = await feedbackReceiver.ReceiveAsync();
            if (feedbackBatch == null) continue;

            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("inf cld 3.1 Received feedback: {0}", string.Join(", ",
feedbackBatch.Records.Select(f => f.StatusCode)));
            Console.ResetColor();

            await feedbackReceiver.CompleteAsync(feedbackBatch);
        }
    }

    static void Main(string[] args)
    {
        Console.WriteLine("INF cld 1.0 Send Cloud-to-Device message\n");
        serviceClient = ServiceClient.CreateFromConnectionString(connectionString);
        ReceiveFeedbackAsync();

        Console.WriteLine("INF cld 1.1 Press any key to send a C2D message.");
        Console.ReadLine();
        SendCloudToDeviceMessageAsync().Wait();
        Console.WriteLine("INF cld 1.2 Message sent from Cloud To Thing.");

        Console.WriteLine("INF cld 1.3 Receive messages. Ctrl-C to exit.\n");
        eventHubClient = EventHubClient.CreateFromConnectionString(connectionString,
IoTHubD2cEndpoint);
        var d2cPartitions = eventHubClient.GetRuntimeInformation().PartitionIds;
        CancellationTokenSource cts = new CancellationTokenSource();
        System.Console.CancelKeyPress += (s, e) =>
        {
            e.Cancel = true;
            cts.Cancel();
            Console.WriteLine("INF cld 1.4 Exiting...");
        };
        var tasks = new List<Task>();
        foreach (string partition in d2cPartitions)
        {
            tasks.Add(ReceiveMessagesFromDeviceAsync(partition, cts.Token));
        }
        Task.WaitAll(tasks.ToArray());
        Console.WriteLine("INF cld 1.5 Press enter to exit program.");
        Console.ReadLine();
    }
}

```

pseudo raspberry interpreter src>data>sample.js

We need also to change the emulated raspberry script based on the sample.js script

```

/*
 * IoT Hub Raspberry Pi NodeJS - Microsoft Sample Code - Copyright (c) 2017 - Licensed MIT
 */
const wpi = require('wiring-pi');
const Client = require('azure-iot-device').Client;
const Message = require('azure-iot-device').Message;
const Protocol = require('azure-iot-device-mqtt').Mqtt;
//const BME280 = require('bme280-sensor');
const SIM908 = require('sim908-sensor');
//const SIM908 = require('bme280-sensor');

const SIM908_OPTION = {
  i2cBusNo: 1, // defaults to 1
  i2cAddress: SIM908.SIM908_DEFAULT_I2C_ADDRESS() // defaults to 0x77
};

// fturi begin : Azure MQTT device
const connectionString = 'HostName=DataBoxHub.azure-
devices.net;DeviceId=DataBoxDevice;SharedAccessKey=*****tP2y16CPGbr6kCL0oMy4=';
// fturi end : Azure MQTT device
const LEDPin = 7;

var sendingMessage = false;
var messageId = 0;
var client, sensor;
var blinkLEDTIMEOUT = null;

function getMessage(cb) {
  messageId++;
  sensor.readSensorData()
    .then(function (data) {
      cb(JSON.stringify({
        messageId: messageId,
        deviceId: 'DataBoxDevice',
        // fturi begin: add SIM908 data collection begin
        latitude: data.latitude,
        longitude: data.longitude,
        client:
          (
            (data.latitude >= 45.4930 && data.latitude <= 45.4950)
            &&
            (data.longitude >= -73.7640 && data.longitude <= -73.7240)
          ),
        azure:
          (
            (data.latitude >= 45.5100 && data.latitude <= 45.5120)
            &&
            (data.longitude >= -73.5770 && data.longitude <= -73.5370)
          )
      }),
      (
        (data.latitude >= 45.4930 && data.latitude <= 45.4950)
        &&
        (data.longitude >= -73.7640 && data.longitude <= -73.7240)
      )
    ) // fturi end : add SIM908 data collection
    );
  })
  .catch(function (err) {
    console.error('Failed to read out sensor data: ' + err);
  });
}

function sendMessage() {
  if (!sendingMessage) { return; }

  getMessage(function (content, temperatureAlert) {
    var message = new Message(content);
    message.properties.add('temperatureAlert', temperatureAlert.toString());
    //console.log('Sending message: ' + content);
    client.sendEvent(message, function (err) {
      if (err) {
        console.error('Failed to send message to Azure IoT Hub');
      }
    });
  });
}

```

```

    } else {
        //fturi disable blinking
        //blinkLED();

        //console.log('Message sent to Azure IoT Hub');
    }
    });
}

function onStart(request, response) {
    console.log('Try to invoke method start(' + request.payload + ')');
    sendingMessage = true;

    response.send(200, 'Successfully start sending message to cloud', function (err) {
        if (err) {
            console.error('[IoT hub Client] Failed sending a method response:\n' +
                err.message);
        }
    });
}

function onStop(request, response) {
    console.log('Try to invoke method stop(' + request.payload + ')');
    sendingMessage = false;

    response.send(200, 'Successfully stop sending message to cloud', function (err) {
        if (err) {
            console.error('[IoT hub Client] Failed sending a method response:\n' +
                err.message);
        }
    });
}

function receiveMessageCallback(msg) {
    var message = msg.getData().toString('utf-8');
    client.complete(msg, function () {

        //fturi add blinkLED when receiving message
        blinkLED();

        console.log('Receive message: ' + message);
    });
}

function blinkLED() {
    // Light up LED for 500 ms
    if(blinkLEDTIMEOUT) {
        clearTimeout(blinkLEDTIMEOUT);
    }
    wpi.digitalWrite(LEDpin, 1);
    blinkLEDTIMEOUT = setTimeout(function () {
        wpi.digitalWrite(LEDpin, 0);
    }, 500);
}

// set up wiring
wpi.setup('wpi');
wpi.pinMode(LEDpin, wpi.OUTPUT);
sensor = new SIM908(SIM908_OPTION);
sensor.init()
    .then(function () {
        sendingMessage = true;
    })
    .catch(function (err) {
        console.error(err.message || err);
    });

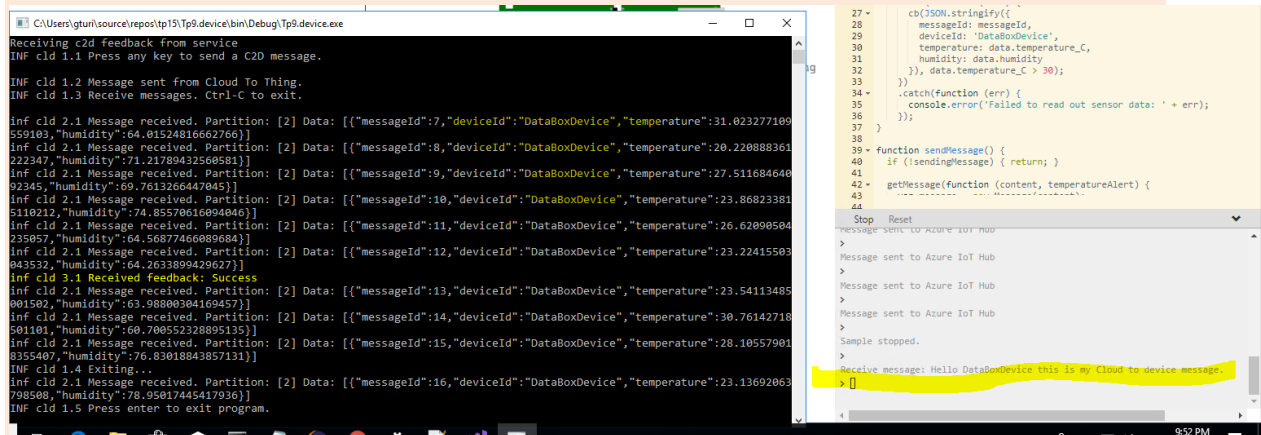
// create a client
client = Client.fromConnectionString(connectionString, Protocol);

client.open(function (err) {
    if (err) {
        console.error('[IoT hub Client] Connect error: ' + err.message);
        return;
    }
}

```

```
// set C2D and device method callback
client.onDeviceMethod('start', onStart);
client.onDeviceMethod('stop', onStop);
client.on('message', receiveMessageCallback);
setInterval(sendMessage, 2000);
});
```

And the C# code does communicate with the Node.js simulator. We adapted the code so the led would blink when an order is received to the simulator.



The screenshot shows a C# application running in a debugger. The console window displays the following output:

```

C:\Users\gtun\source\repos\tp15\Tp9\device\bin\Debug\Tp9.device.exe
Receiving c2d feedback from service
INF cld 1.1 Press any key to send a C2D message.
INF cld 1.2 Message sent from Cloud To Thing.
INF cld 1.3 Receive messages. Ctrl-C to exit.

inf cld 2.1 Message received. Partition: [2] Data: [{"messageId":7,"deviceId":"DataBoxDevice","temperature":31.023277109
559103,"humidity":64.01524816662766}]
inf cld 2.1 Message received. Partition: [2] Data: [{"messageId":8,"deviceId":"DataBoxDevice","temperature":20.220888361
222347,"humidity":71.21789432560581}]
inf cld 2.1 Message received. Partition: [2] Data: [{"messageId":9,"deviceId":"DataBoxDevice","temperature":27.511684640
92245,"humidity":69.7013266449045}]
inf cld 2.1 Message received. Partition: [2] Data: [{"messageId":10,"deviceId":"DataBoxDevice","temperature":23.86823381
5110212,"humidity":74.85570616094046}]
inf cld 2.1 Message received. Partition: [2] Data: [{"messageId":11,"deviceId":"DataBoxDevice","temperature":26.62090504
235057,"humidity":64.56877466089684}]
inf cld 2.1 Message received. Partition: [2] Data: [{"messageId":12,"deviceId":"DataBoxDevice","temperature":23.22415503
843532,"humidity":64.26338994296277}]
inf cld 3.1 Received feedback: Success
inf cld 2.1 Message received. Partition: [2] Data: [{"messageId":13,"deviceId":"DataBoxDevice","temperature":23.54113485
001502,"humidity":63.98800304169457}]
inf cld 2.1 Message received. Partition: [2] Data: [{"messageId":14,"deviceId":"DataBoxDevice","temperature":30.76142718
501101,"humidity":60.700552320895105}]
inf cld 2.1 Message received. Partition: [2] Data: [{"messageId":15,"deviceId":"DataBoxDevice","temperature":28.10557901
8355407,"humidity":76.83018843857131}]
INF cld 1.4 Exiting...
inf cld 2.1 Message received. Partition: [2] Data: [{"messageId":16,"deviceId":"DataBoxDevice","temperature":23.13692863
798508,"humidity":78.95017445417936}]
INF cld 1.5 Press enter to exit program.
  
```

The code editor shows the following C# code:

```

27 - cb(JSON.stringify({
28     messageId: messageId,
29     deviceId: 'DataBoxDevice',
30     temperature: data.temperature_C,
31     humidity: data.humidity
32 })), data.temperature_C > 30);
33 })
34 - .catch(function (err) {
35     console.error('Failed to read out sensor data: ' + err);
36 });
37 }
38
39 - function sendMessage() {
40     if (!sendingMessage) { return; }
41     getMessage(function (content, temperatureAlert) {
42         // ...
43     });
44 }
  
```

The console window also shows a message from the Azure IoT Hub: "Hello DataBoxDevice this is my Cloud to device message."

The code use a module BME280 to detect temperature and humidity. We want also geo-position. We are going to simul a new GPS module SIM-908, based on:

<https://www.cooking-hacks.com/documentation/tutorials/geolocation-tracker-gprs-gps-geoposition-sim908-arduino-raspberry-pi/>

src>lib>sim908.js

This emulator is based on the code of the bme220.js which is founded in the same directory.

```
class SIM908 {
  constructor() {
    console.log("SIM908 constructor");
    this.readSensorData = this.readSensorData.bind(this);
    this.init = this.init.bind(this);
    /* looks like I cannot use var there to declare local variable ...*/
    /*
    this.time=this.time.bind(this);
    this.tic=this.tic.bind(this);
    this.tps=this.tps.bind(this);
    */
  }

  init(option) {
    this.inited = true;

    // add new class variable following exactly _inited and this.inited
    /* looks like that variable must be initated here */
    this.time=0;
    this.tic=0;
    this.tps=0;

    return new Promise(function (resolve/*, reject*/) {
      resolve();
    });
  }

  reset() {
    return new Promise(function (resolve/*, reject */) {
      resolve();
    });
  }

  readSensorData() {
    //https://stackoverflow.com/questions/29734312/javascript-access-parent-object-attribute
    console.log("my inside read Sensor");
    var _inited = this.inited;

    // add new class variable following exactly _inited and this.inited
    var _time=this.time;
    var _tic=this.tic;
    var _tps=this.tps;
    /* */
    var _time=0;
    var _tic=1;
    var _tps=2;
    /**/
    // side effect must be done here and not into Promise lambda function
    if (_time>1000)
    {
      _tic++;
      if (_tic%2>0)
      {
        _tps=1000;
      }
      else
      {
        _tps=0;
      }
    }
    if (_tic%2>0)
    {
      _tps++;
    }
    else
    {
      _tps--;
    }
  }
}
/* * /
```



```

    parent.time=_time;
    parent.tic=_tic;
    parent.tps=_tps;
    /**
    this.time=_time;
    this.tic=_tic;
    this.tps=_tps;
    **/

    //return, so last instruction to be executed
    return new Promise(function (resolve, reject) {
    if (!_inited) {
        return reject('You must first call sim908.init()');
    }
    resolve({
        /**
        temperature_C: BME280.random(20, 32),
        humidity: BME280.random(60, 80),
        pressure_hPa: BME280.random(10, 12)
        **/
        /* Client 275 viger E:    45.511,73.557
        * Azure 4705 Dobrin  :    45.494,73.744
        * Delta                .17      .187
        */
        latitude: 45.494+(_tps* 1*0.0001),
        longitude: -73.744+(_tps*11*0.0001)
        /**
        */
    });
    });
}

static random(min, max) {
    return Math.random() * (max - min) + min;
}

static SIM908_DEFAULT_I2C_ADDRESS() {
    return 0x87;
}

static CHIP_ID1_SIM908() {
    return 0x156;
}

static CHIP_ID2_SIM908() {
    return 0x157;
}

static CHIP_ID3_SIM908() {
    return 0x158;
}

static CHIP_ID_SIM908() {
    return 0x160;
}

static convertCelciusToFahrenheit(c) {
    return c * 9 / 5 + 32;
}

static convertHectopascalToInchesOfMercury(hPa) {
    return hPa * 0.02952998751;
}

static convertMetersToFeet(m) {
    return m * 3.28084;
}
}

export default SIM908;

```

src>lib>sample.js

We add the following lines in:
src\lib\sample.js

We need to add this library. This is done in sample.js

```
import { Client, Message } from 'azure-iot-device'
import { traceEvent } from './telemetry.js';
import Protocol from './mqtt.js';
import wpi from './wiring-pi.js';
import codeFactory from './data/codeFactory.js';
import BME280 from './bme280.js';
import SIM908 from './sim908.js'; // fturi added sim908 library

class Clientwrapper extends Client {
  constructor(transport, connStr, blobUploadClient) {
    window.azure_iot_device_client = super(transport, connStr, blobUploadClient);
    return window.azure_iot_device_client;
  }

  static fromConnectionString(connStr, transport) {
    window.azure_iot_device_client = super.fromConnectionString(connStr, transport);
    return window.azure_iot_device_client;
  }
}

class Sample {
  constructor() {
    this.runningFunction = null;
    if(!window.oldSetTimeout) {
      window.timeoutList = new Array();
      window.intervalList = new Array();

      window.oldSetTimeout = window.setTimeout;
      window.oldSetInterval = window.setInterval;
      window.oldClearTimeout = window.clearTimeout;
      window.oldClearInterval = window.clearInterval;

      window.setTimeout = function(code, delay) {
        var retval = window.oldSetTimeout(code, delay);
        window.timeoutList.push(retval);
        return retval;
      };
      window.clearTimeout = function(id) {
        var ind = window.timeoutList.indexOf(id);
        if(ind >= 0) {
          window.timeoutList.splice(ind, 1);
        }
        var retval = window.oldClearTimeout(id);
        return retval;
      };
      window.setInterval = function(code, delay) {
        var retval = window.oldSetInterval(code, delay);
        window.intervalList.push(retval);
        return retval;
      };
      window.clearInterval = function(id) {
        var ind = window.intervalList.indexOf(id);
        if(ind >= 0) {
          window.intervalList.splice(ind, 1);
        }
        var retval = window.oldClearInterval(id);
        return retval;
      };
      window.clearAllTimeouts = function() {
        for(var i in window.timeoutList) {
          window.oldClearTimeout(window.timeoutList[i]);
        }
        window.timeoutList = new Array();
      };
      window.clearAllIntervals = function() {
        for(var i in window.intervalList) {
          window.oldClearInterval(window.intervalList[i]);
        }
        window.intervalList = new Array();
      };
    }
  }
}
```

```

    };
  }
  this.actualClient = null;
}

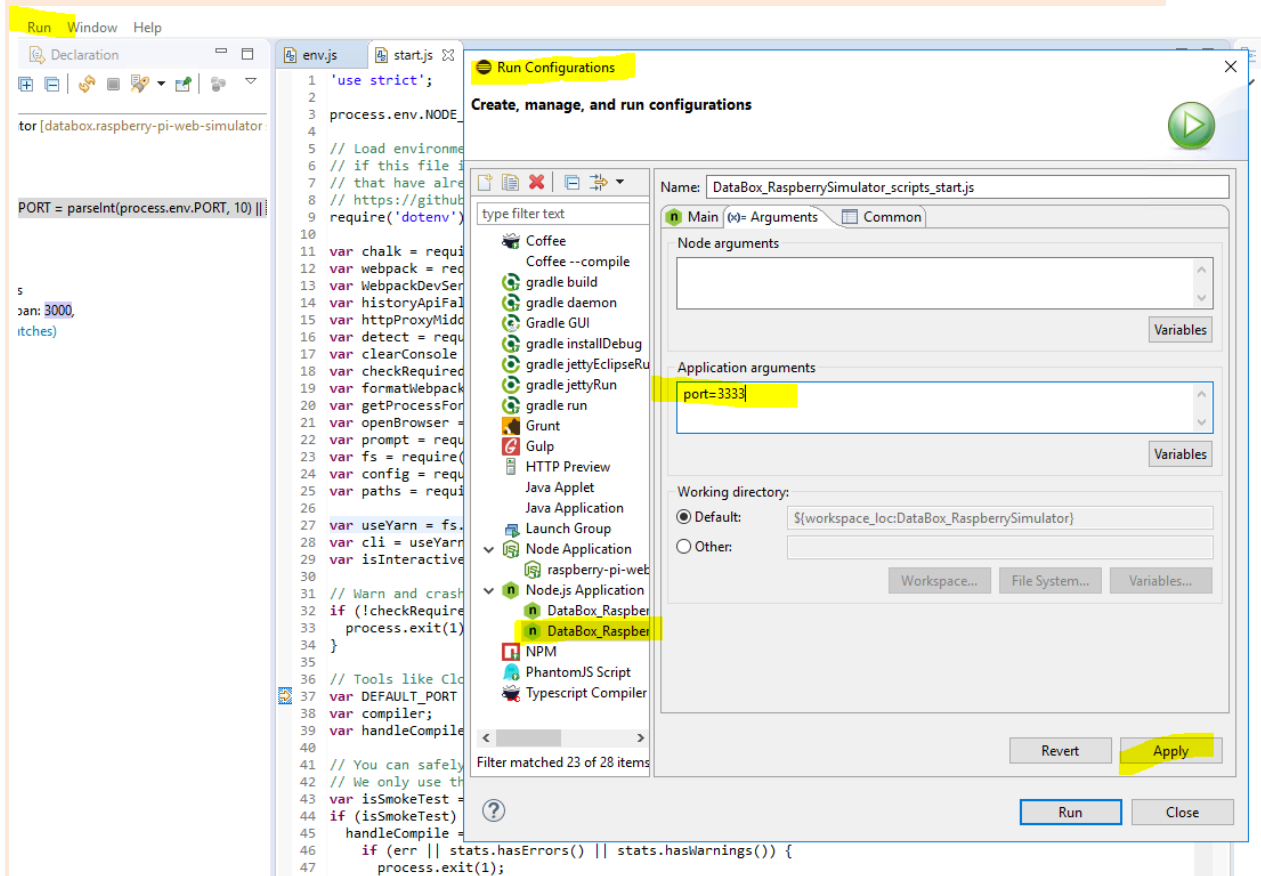
stop() {
  window.clearAllIntervals();
  window.clearAllTimeouts();
  if(window.azure_iot_device_client) {
    window.azure_iot_device_client.close();
  }
}

run(option) {
  // a prefix of UUID to avoid name conflict, here just use a fix one
  const prefix = '76f98350';
  var replaces = [
    {
      src: /require\('wiring-pi'\)/g,
      dest: 'wpi'
    }, {
      src: /require\('azure-iot-device'\)\.Client/g,
      dest: 'Client'
    }, {
      src: /require\('azure-iot-device'\)\.Message/g,
      dest: 'Message'
    }, {
      src: /require\('azure-iot-device-mqtt'\)\.Mqtt/g,
      dest: 'Protocol'
    }, {
      src: /require\('bme280-sensor'\)/g,
      dest: 'BME280'
    }, { // fturi begin Added SIM908 declaration into raspberry emulator language
      src: /require\('sim908-sensor'\)/g,
      dest: 'SIM908'
    }, { // fturi end Added SIM908 declaration into raspberry emulator language
      src: /console\.log/g,
      dest: 'msgCb'
    }, {
      src: /console\.error/g,
      dest: 'errCb'
    }
  ];
  wpi.setFunc(option.ledSwitch);
  try {
    traceEvent('run-sample');
    var src = codeFactory.getRunCode('index', replaces, prefix);
    this.runningFunction = new Function('replaces' + prefix, src);
    this.runningFunction({
      wpi: wpi,
      Client: ClientWrapper,
      Message: Message,
      Protocol: Protocol,
      BME280: BME280,
      SIM908: SIM908, // fturi added SIM908 constants
      msgCb: option.onMessage,
      errCb: option.onError
    });
    // if (src.search(/^(?!\\\/).)*setInterval/gm) < 0) {
    //   option.onFinish();
    // }
  } catch (err) {
    traceEvent('run-error', { error: err });
    option.onError(err.message || JSON.stringify(err));
    option.onFinish();
  }
}

export default Sample;

```

We also need to change the default port of 3000 using eclipse project
 Click run -> Run configuration
 Node.js doesn't support environment variable, so we pass it as a parameter

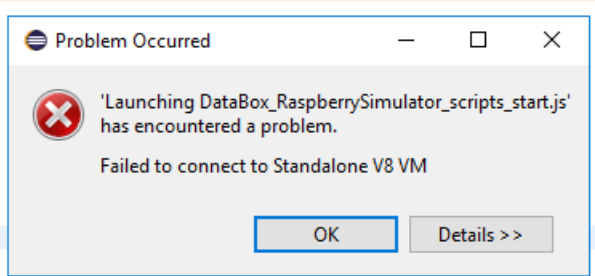


It should be taken in account in
 script>start.js

But , it doesn't work...

Debugging Node.js

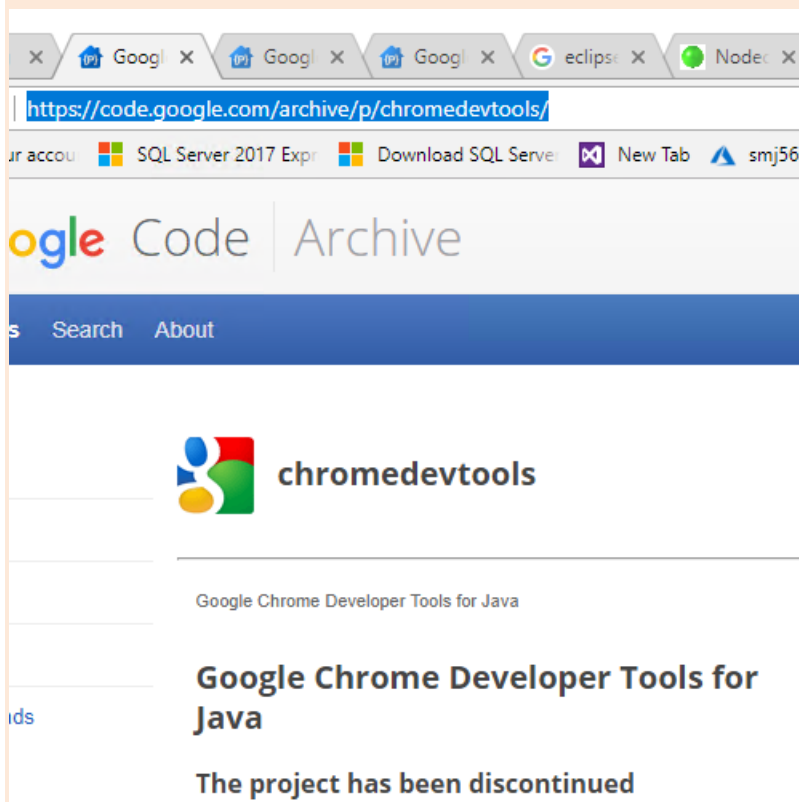
The eclipse doesn't come with debugger with node.js (complains about not being able to connect to the VM)



The Standalone V8 VM is a discontinued project

<https://github.com/nodejs/node/wiki/Using-Eclipse-as-Node-Applications-Debugger>

then <https://code.google.com/archive/p/chromedevtools/>



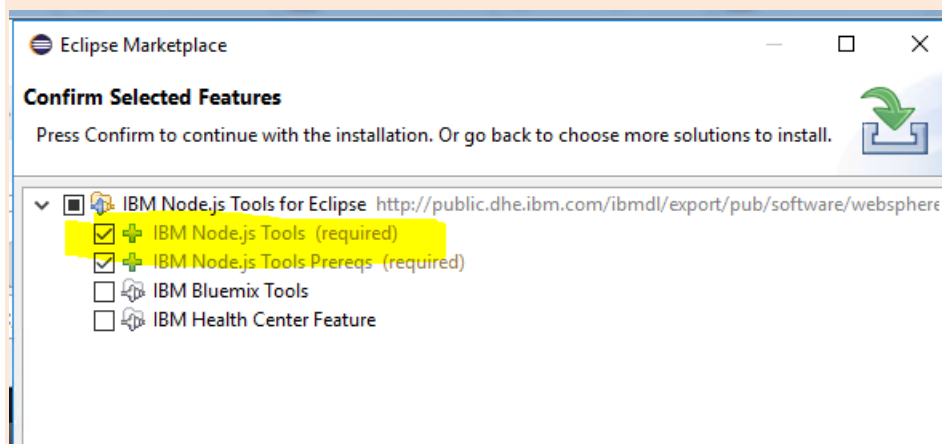
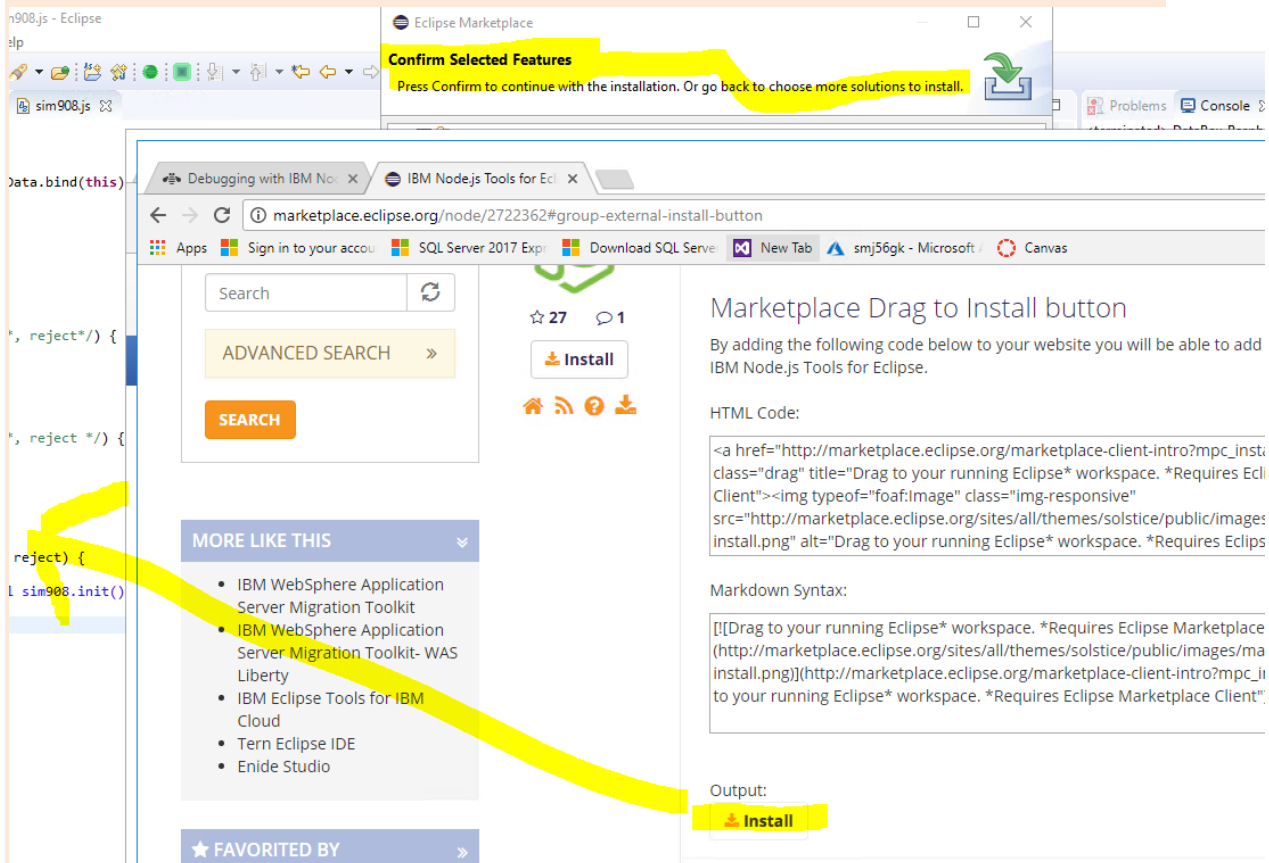
We will install instead the IBM debugging tools

[https://www.ibm.com/developerworks/community/blogs/nodejstools/entry/Debugging with IBM Node.js Tools for Eclipse Beta?lang=en](https://www.ibm.com/developerworks/community/blogs/nodejstools/entry/Debugging%20with%20IBM%20Node.js%20Tools%20for%20Eclipse%20Beta?lang=en)

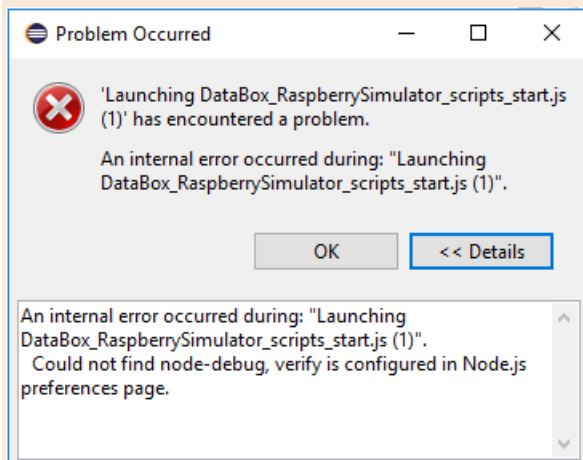
then

<http://marketplace.eclipse.org/node/2722362#group-external-install-button>

By dragging the button to eclipse



Humm, looks like we are missing some parts



"Could not find node-debug, verify is configured in Node.js preferences page."

Lets go back to the Node.js installation guide:

<http://www.nodeclipse.org/updates/>

- 1) Node.js: done
- 2) Nodeeclipse CLI and Express

<https://github.com/Nodeclipse/nodeclipse-1/tree/master/org.nodeclipse.ui/templates#nodeclipse-cli--installer>

```
$ cd ~/source/repos/Tp15
$ npm install -g nodeclipse
C:\Users\gturi\AppData\Roaming\npm\nodeclipse -> C:\Users\gturi\AppData\Roaming\npm\node_modules\nodeclipse\bin\nodeclipse.js
C:\Users\gturi\AppData\Roaming\npm\nci -> C:\Users\gturi\AppData\Roaming\npm\node_modules\nodeclipse\bin\nci.js
C:\Users\gturi\AppData\Roaming\npm\epm -> C:\Users\gturi\AppData\Roaming\npm\node_modules\nodeclipse\bin\epm.js
+ nodeclipse@0.17.2
added 5 packages in 1.886s
$ npm install -g express
+ express@4.16.2
added 49 packages in 2.834s
$ npm install -g express-generator
C:\Users\gturi\AppData\Roaming\npm\express ->
C:\Users\gturi\AppData\Roaming\npm\node_modules\express-generator\bin\express-cli.js
+ express-generator@4.15.5
added 6 packages in 1.112s
```


- 3) Install **JDK** (8u161)

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>


- 4) Install Eclipse: done

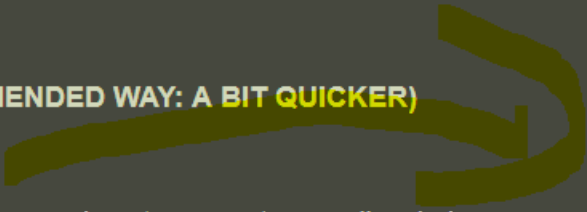
- 5) Install Enide plugin (drag icon to eclipse)

INSTALLATION/UPDATE INSTRUCTIONS
(for Eclipse or Enide [Studio])

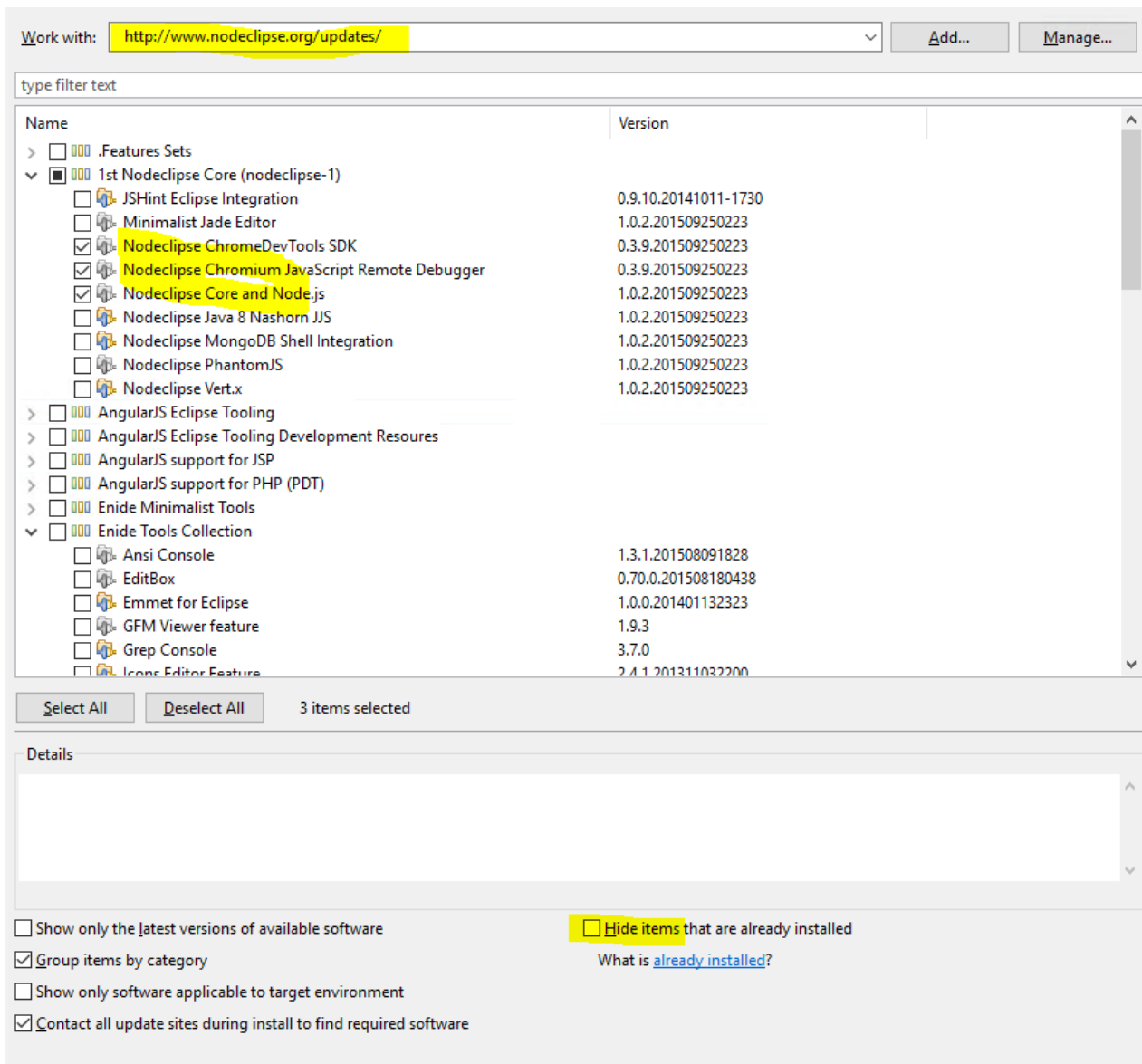
In short words, drag-and-drop  on Eclipse main toolbar and get **Enide 2015** plugins.

A) MARKETPLACE (RECOMMENDED WAY: A BIT QUICKER)

1. Start Eclipse.
2. Drag and drop  into a running Eclipse (menu area) to install Nodeclipse.



- 6) Check installation with Eclipse > Help > install software. Looks like its all there



7) OK, according to <https://www.npmjs.com/package/node-debug>, we need to run

```
$ npm install -g node-debug
npm WARN deprecated node-debug@0.1.0: functionality now built in to node-inspector
npm WARN deprecated connect@2.12.0: connect 2.x series is deprecated
npm WARN deprecated minimatch@0.3.0: Please update to minimatch 3.0.2 or higher to avoid
a RegExp DoS issue
C:\Users\gturi\AppData\Roaming\npm\node-debug ->
C:\Users\gturi\AppData\Roaming\npm\node_modules\node-debug\bin\node-debug.js

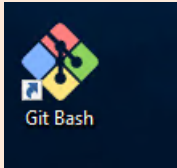
> ws@0.4.32 install C:\Users\gturi\AppData\Roaming\npm\node_modules\node-
debug\node_modules\ws
> (node-gyp rebuild 2> builderror.log) || (exit 0)

C:\Users\gturi\AppData\Roaming\npm\node_modules\node-debug\node_modules\ws>if not defined
npm_config_node_gyp (node "C:\Program Files\nodejs\node_modules\npm\node_modules\npm-
lifecycle\node-gyp-bin\..\..\node_modules\node-gyp\bin\node-gyp.js" rebuild) else
(node "C:\Program Files\nodejs\node_modules\npm\node_modules\node-gyp\bin\node-gyp.js"
rebuild )
```

```
+ node-debug@0.1.0
added 57 packages in 6.193s
```

Debugging not working

Looks like the node.js debugging is broken under VC and Eclipse. Documentation shows that it was working in the past. Only debugger left is the command line still using the GIT bash tools

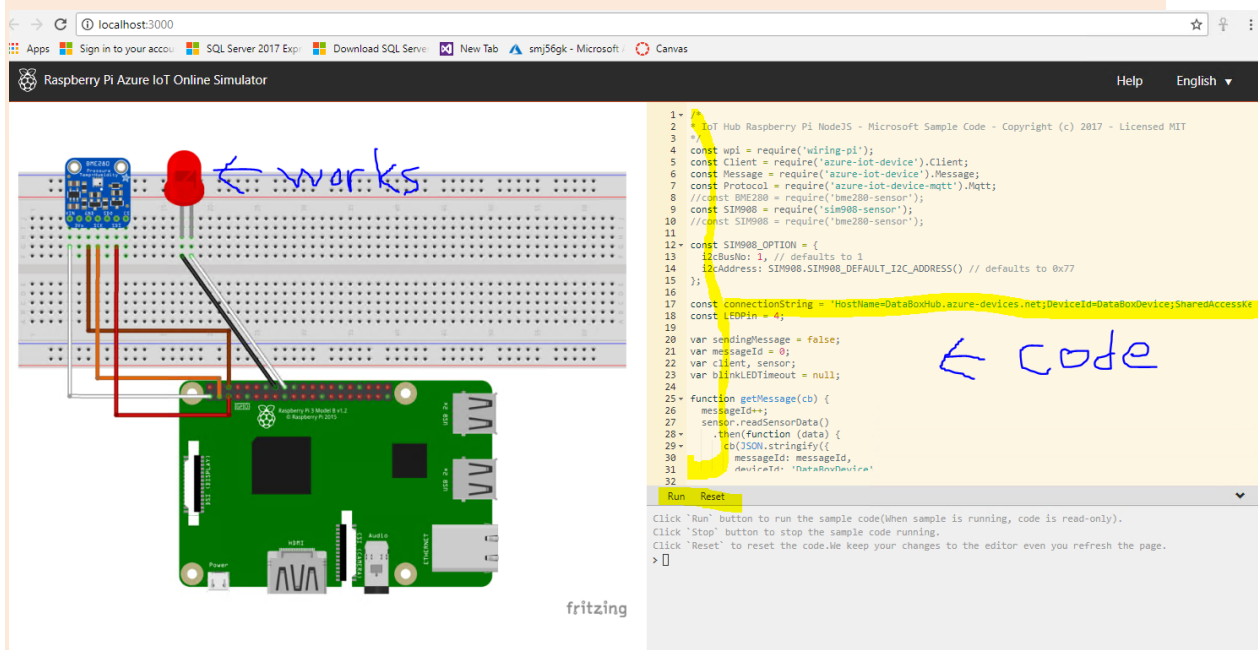


```
$ cd ~/source/repos/Tp15/databox.raspberry-pi-web-simulator
gturi@emj56gkF1S MINGW64 ~/source/repos/Tp15/databox.raspberry-pi-web-simulator (source)
$ node debug scripts/start.js
(node:62944) [DEP0068] DeprecationWarning: `node debug` is deprecated. Please use `node inspect` instead.
< Debugger listening on ws://127.0.0.1:9229/8b6c83f7-c466-4e5b-afd2-cd6a8515436f
< For help see https://nodejs.org/en/docs/inspector
Break on start in scripts\start.js:1
> 1 (function (exports, require, module, __filename, __dirname) { 'use strict';
  2
  3 process.env.NODE_ENV = 'development';
debug>
(To exit, press ^C again or type .exit)
debug>
```

Raspberry emulator: GPS function for databox

Implementing the simulator

The good news is that the simulator is working and does interpret the code on the screen. Which is the nice part of that emulator. Nothing is compiled, code is on the screen



The led does react to the code which is written in javascript

The RUN interpret the code and does act as an MQTT

Limitation

The only dynamic part of the left side (drawing) is the Led, the rest is static.

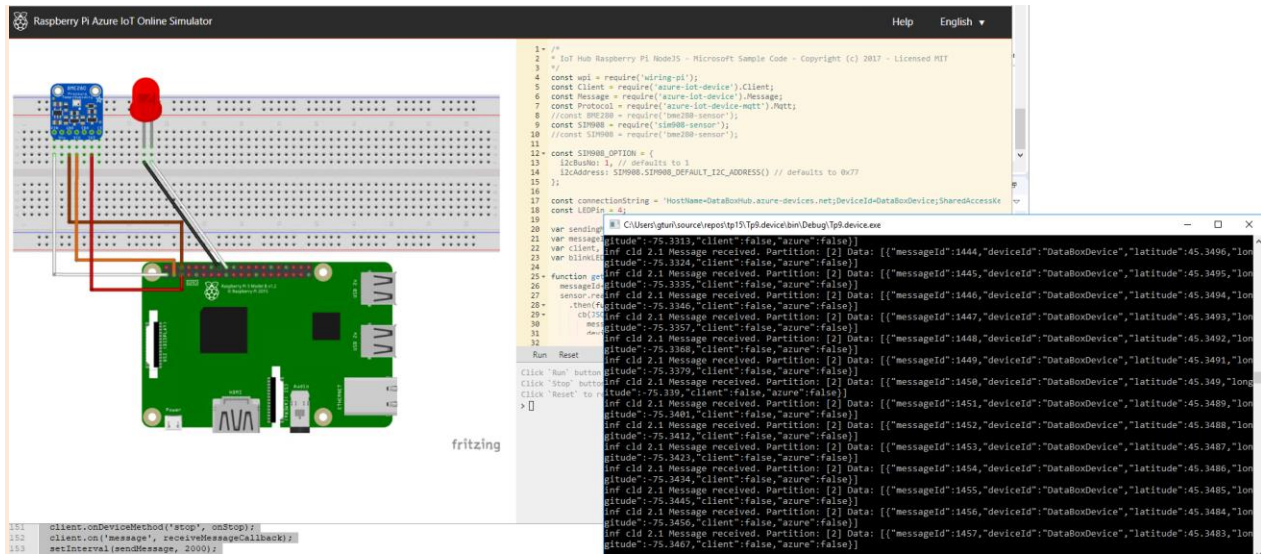
Only one emulated captor is implemented : the BME280 which is a temperature and pressure captor. Code shows that values are just random.

Alert doesn't seems to work

Well, lets go develop on that emulator framework

So, we add another captor: a simulated geopositioning which would trace the steps between McKesson Canada office (seen as client) and IBM viger datacenter (seen as azure)

Geopositioning and geofencing is working



OK, next step is to represent on a map the geotravelling of our databox.

Dynamic mapping of our databox

We will follow the instruction on

<https://docs.microsoft.com/en-us/azure/location-based-services/quick-demo-map-app>

Azure service: Location Based Services (preview)

Location Based Services (preview)

smj56gk@mckesson.c...
MCKESSON CORPORATION

Location Based Services (preview)
Microsoft

Azure Location Based Services (LBS) is a portfolio of geospatial services that include service APIs for Maps, Search, Routing, Traffic and Time Zones. The portfolio of Azure OneAPI compliant services allows you to use familiar developer tools to quickly develop and scale solutions that integrate location information into your Azure solutions. Azure LBS provides developers from all industries powerful geospatial capabilities packed with fresh mapping data imperative to providing geographic context to web and mobile applications.

Twitter Facebook LinkedIn YouTube Google+ Email

Seattle

Create

We create the account

Create Location Based Services Account

PREVIEW

✕

* Name

DataBoxLocation ✓

* Subscription

McKesson Deep Dive Training (7) ▾

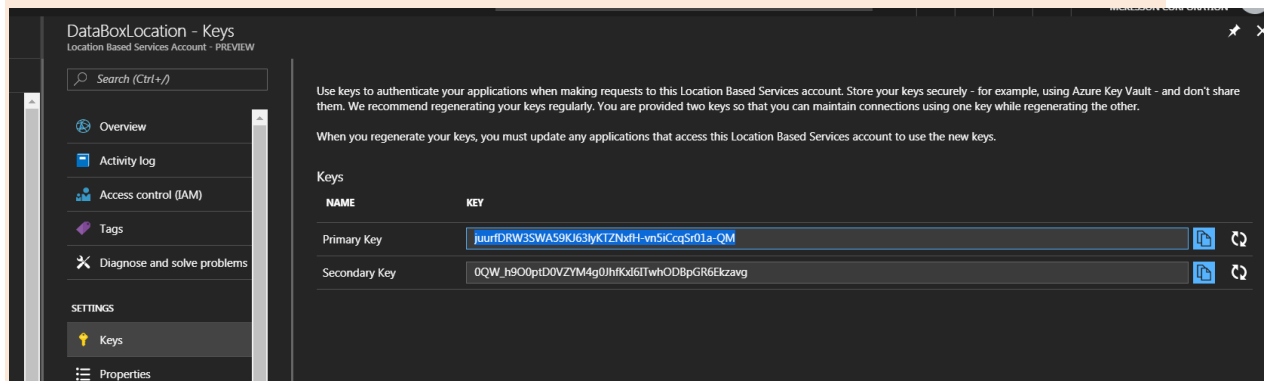
* Resource group

☐ Create new ☒ Use existing

smj56gk ▾

☒ * I confirm I have read and agree to the [Preview Terms](#)

Now we need the primary key: juurfDRW3SWA59KJ63lyKTZNxfH-vn5iCcqSr01a-QM



Install the code

We will follow TP11 instructions to install IIS and the sample location page. Well kind of as we have windows10 following the instruction on:
<https://letitknow.wordpress.com/2012/10/22/install-iis-on-windows-8-with-powershell/>

```
Get-Command -Module dism
PS C:\WINDOWS\system32> Enable-windowsOptionalFeature -Online -FeatureName IIS-WebServerRole
```

```
Path : 
Online : True
RestartNeeded : False
```

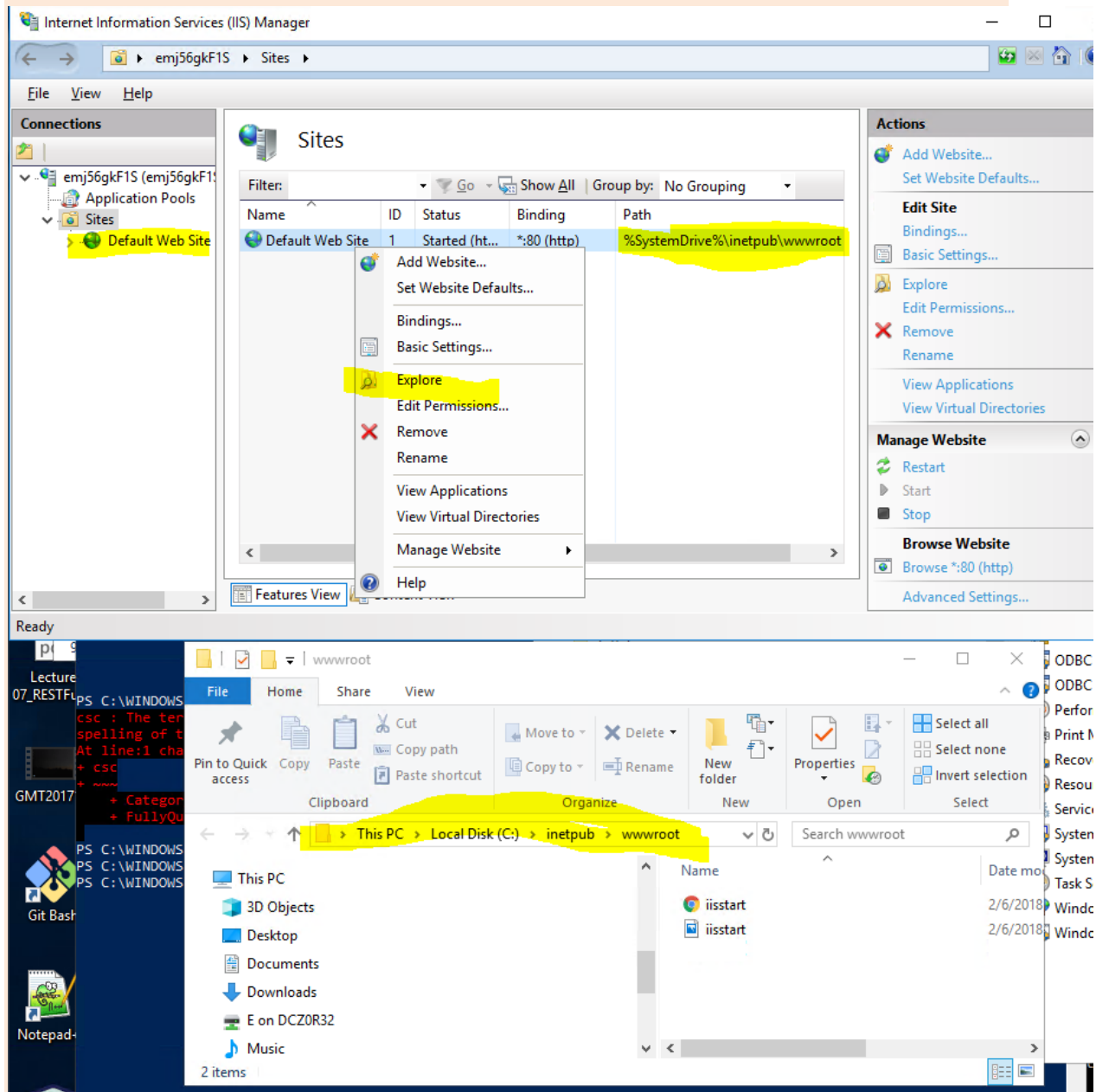

Configure a new webserver.

We follow: <https://support.microsoft.com/en-ca/help/323972/how-to-set-up-your-first-iis-web-site>

Open explorer and go to :
Control Panel\System and Security\Administrative Tools

Right click on Internet Information Service (run as admin)

Well there is already a default website. We will use that one



Basic settings give us the path:

%SystemDrive%\inetpub\wwwroot

For practical point of view, we will change the default directory to the project. We need to update ACLs using the following command line

(<https://stackoverflow.com/questions/2928738/how-to-grant-permission-to-users-for-a-directory-using-command-line-in-windows>) and <https://support.microsoft.com/en-ca/help/919240/the-icaccls-exe-utility-is-available-for-windows-server-2003-with-servi>

```
C:\>icaccls "C:\Users\gturi\source\repos\tp15\geotracking" /grant IIS_IUSRS:(OI)(CI)R,RD /T
processed file: C:\Users\gturi\source\repos\tp15\geotracking
processed file: C:\Users\gturi\source\repos\tp15\geotracking\interactiveSearch.html
Successfully processed 2 files; Failed processing 0 files
```

We also need to create a symbolic link as home directory is protected (using cmd as administrator)

```
C:\WINDOWS\system32>mklink /J %SystemDrive%\inetpub\wwwroot\geotracking
Junction created for C:\inetpub\wwwroot\geotracking <<===>>
C:\Users\gturi\source\repos\tp15\geotracking
```

Well it doesn't work that way (403). Better to do the other way. Create a symbolic link from home directory towards wwwroot

```
mklink /J "C:\Users\gturi\source\repos\tp15\geotracking"
%SystemDrive%\inetpub\wwwroot\geotracking
```

Well, the geolocalisation from Azure doesn't give an easy way to pinpoint a location.

We need two pieces of code:

- one very simple node.js client connecting IOT to read data
- &
- one very simple node.js to show googlemap

<https://tympanus.net/codrops/2012/10/11/real-time-geolocation-service-with-node-js/>
<https://www.sitepoint.com/google-maps-made-easy-with-gmaps-js/>

and

<https://prazjain.wordpress.com/2012/04/19/maps-example-with-google-maps-and-nodejs/>

That one was interesting too

<https://tympanus.net/codrops/2012/10/11/real-time-geolocation-service-with-node-js/>

We download the gmaps.js from github

<https://raw.githubusercontent.com/HPNeo/gmaps/master/gmaps.js>

We need to get a key from google linked to our account

<https://developers.google.com/maps/documentation/javascript/get-api-key>

Sample script:

```
<!doctype html>
<html>
  <head>
    <style>
      #map {
        width: 400px;
```

```

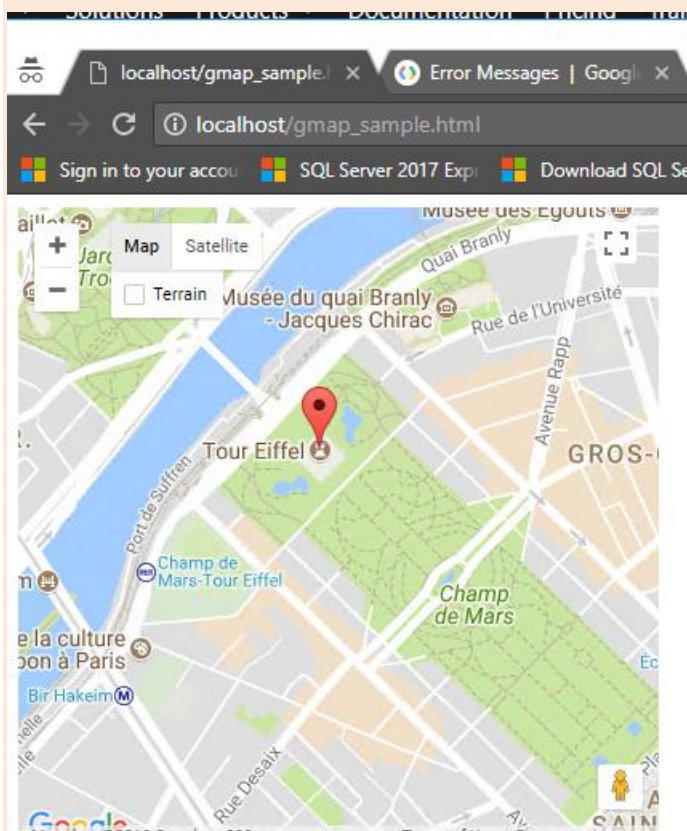
        height: 400px;
    }
</style>
</head>
<body>
    <div id="map"></div>

    <!-- Google Maps JS API -->
    <script
src="https://maps.googleapis.com/maps/api/js?key=[GoogleApiKey]&callback=initMap"></scrip
t>
    <!-- GMaps Library -->
    <script src="gmaps.js"></script>
    <script>
        /* Map Object */
        var mapObj = new GMaps({
            el: '#map',
            lat: 48.857,
            lng: 2.295
        });

        var m = mapObj.addMarker({
            lat: 48.8583701,
            lng: 2.2944813,
            title: 'Eiffel Tower',
            infowindow: {
                content: '<h4>Eiffel Tower</h4><div>Paris, France</div>',
                maxWidth: 100
            }
        });
    </script>
</body>
</html>

```

Result on the screen: Yes, we have a marker.



Ok, we have a functional javascript to pinpoint a map.

The definitive reference for gmaps is

<https://developers.google.com/maps/documentation/javascript/reference#Animation>
and

<https://hpneo.github.io/gmaps/examples/markers.html>

Simple code to connect to MQTT

A few promising framework were around is paho framework explained here
http://workswithweb.com/html/mqttbox/mqtt_client_settings.html

The paho provide JS implementation
<https://www.eclipse.org/paho/clients/js/utility/>

Which work using websocket implemented by message broker as explained here
<https://social.microsoft.com/Forums/azure/en-US/ca68041a-d098-4d11-b108-fe3c76420281/using-mqtt-over-websockets-on-port-443?forum=azureiothub>
 and explained in that excellent guide:
<https://blog.risingstack.com/getting-started-with-nodejs-and-mqtt/>

There is some mention about AMqp over websockets
<https://blogs.msdn.microsoft.com/zhigang/2017/03/14/connect-to-azure-iot-hub-in-browser-using-amqp-over-websockets/>
 But again, its AMqp and not MQTT.

A way to test it is to use:
<https://www.hivemq.com/blog/build-javascript-mqtt-web-application>
 and
<http://www.hivemq.com/demos/websocket-client/>

This page mention MQTT broker project.
<https://azure.microsoft.com/en-us/resources/samples/iot-gateway-mqtt-http/>
 Did not try it as went the node.js avenue instead

It looks like that paho is not supported by azure
<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support>
<https://stackoverflow.com/questions/36660410/python-paho-mqtt-connection-with-azure-iot-hub>
 But node.js is.

Node.js framework for MQTT (express)

We followed the Azure instruction:

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-node-node-getstarted>

<http://expressjs.com/>

Node.js web framework

We follow <https://docs.microsoft.com/en-us/azure/app-service/app-service-web-get-started-nodejs>

And the excellent:
<https://blog.risingstack.com/your-first-node-js-http-server/>

Build the console window

We follow the tedious form (but work at the end) creation guide using Node.js

<https://www.sitepoint.com/forms-file-uploads-security-node-express/>

This one also light a different light on some details

<https://blog.risingstack.com/node-hero-node-js-request-module-tutorial/>

and that one too

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/forms/Create_book_form

Final Project: On the Emulator-side

After a week, we had to add a few components.

On the emulator side, we added an SdCard to store the image:

```
class SDCARD {

    constructor() {
        console.log("SDCARD constructor");
        this.uploadFile = this.uploadFile.bind(this);
        this.downloadFile = this.downloadFile.bind(this);
        this.init = this.init.bind(this);
    }

    init(option) {
        this.inited = true;

        // add new class variable following exactly _inited and this.inited
        /* looks like that variable must be initated here */
        this.data=0;

        return new Promise(function (resolve/*, reject*/) {
            resolve();
        });
    }

    reset() {
        return new Promise(function (resolve/*, reject */) {
            resolve();
        });
    }

    downloadFile() {
        //https://stackoverflow.com/questions/29734312/javascript-access-parent-object-attribute
        console.log("my inside downloadFile");
        var _inited = this.inited;
        var _data = this.data;

        //return, so last instruction to be executed
        return new Promise(function (resolve, reject) {
            if (!_inited) {
                return reject('You must first call SDCARD.init()');
            }
            resolve({data:_data});
        });
    }

    uploadFile(_data) {
        //https://stackoverflow.com/questions/29734312/javascript-access-parent-object-attribute
        console.log("my inside uploadFile");
        var _inited = this.inited;
        this.data=_data ;

        //return, so last instruction to be executed
        return new Promise(function (resolve, reject) {
            if (!_inited) {
                return reject('You must first call SDCARD.init()');
            }
            resolve({
                size:length(_data)
            });
        });
    }

    static SDCARD_DEFAULT_I2C_ADDRESS() {
        return 0x97;
    }

    static CHIP_ID1_SDCARD() {
```

```
    return 0x166;
}

static CHIP_ID2_SDCARD() {
    return 0x167;
}

static CHIP_ID3_SDCARD() {
    return 0x168;
}

static CHIP_ID_SDCARD() {
    return 0x170;
}

}

export default SDCARD;
```

SIM908: Geolocalisation

We also adapted the SIM908, so the apparent motion can be directed from the emulator using some status. Status 66, will let him go back and forth.
 We tried hard to create a console class to log the steps inside the library. It just never worked. Seems like the class structure was another Jail inside the react-jail (see last chapter)

```
//import {MY} from './myconsole.js';
//import {mylog} from './my.js';
//import {} from react-native-fs;

class SIM908 {

  constructor() {
    //this.console=new MY();
    //super();
    //this.myconsole("SIM908 constructor");
    this.readSensorData = this.readSensorData.bind(this);
    //this.move=this.move.bind(this);
    this.init = this.init.bind(this);
    this.getState=this.getState.bind(this);
    this.setState=this.setState.bind(this);
    //this.myconsole=this.myconsole.bind(this);
    /* looks like I cannot use var there to declare local variable ...*/
    this.time=this.time.bind(this);
    this.tic=this.tic.bind(this);
    this.tps=this.tps.bind(this);
  }

  init(option) {
    this.inited = true;
    //this.myconsole();

    // add new class variable following exactly _inited and this.inited
    /* looks like that variable must be initated here */
    this.time=0;
    this.tic=0;
    this.tps=0;
    this.wait=0;
    this.status=0;
    this.status=3;

    return new Promise(function (resolve/*, reject*/) {
      resolve();
    });
  }

  reset() {
    return new Promise(function (resolve/*, reject */) {
      resolve();
    });
  }

  getState()
  {
    return this.status;
  }
  setState(new_state)
  {
    this.status=new_state;
  }

  readSensorData() {
    //https://stackoverflow.com/questions/29734312/javascript-access-parent-object-attribute
    //this.myconsole.log("my inside read Sensor");
    var _inited = this.inited;

    // add new class variable following exactly _inited and this.inited
  }
}
```

```

var _time=this.time;
var _tic=this.tic;
var _tps=this.tps;
var steps=20;
var _wait=this.wait;
var isteps=0.001/steps;

/* */
var _time=0;
var _tic=1;
var _tps=2;
/**/
// side effect must be done here and not into Promise lambda function
if (_tps>steps || _tps==0)
{
if (_wait++>20)
{
_wait=0;
}
}
if (this.status<66)
{
_wait=1;
switch (this.status)
{
case 0:
_tic=1;_tps=steps;
break;
case 1:
_tic=1;_tps=steps;
break;
case 2:
_tic=1;_wait=0;
break;
case 3:
_tic=1;
break;
case 4:
_tic=1;
break;
case 5:
_tic=0;_wait=0;
break;
case 6:
_tic=0;
break;
}
}
if (0==_wait)
{
if (_tic%2==0)
{
_tps++;
}
else
{
_tps--;
}

if (_tps>steps || _tps<0)
{
if (_tic%2==0)
{
_tps=steps;
}
else
{
_tps=0;
}
if (66==this.status)
_tic++;
}
}

/* */
parent.time=_time;
parent.tic=_tic;
parent.tps=_tps;
/* */

```

```

    this.time=_time;
    this.tic=_tic;
    this.tps=_tps;
    this.wait=_wait;
    /**/

    //return, so last instruction to be executed
    return new Promise(function (resolve, reject) {
    if (!_inited) {
        return reject('You must first call sim908.init()');
    }
    resolve({
        /** */
        temperature_C: BME280.random(20, 32),
        humidity: BME280.random(60, 80),
        pressure_hPa: BME280.random(10, 12)
        /**/
        /* Client 275 viger E: 45.511,73.557
        * Azure 4705 Dobrin : 45.494,73.744
        * Delta .17 .187
        */
        latitude: 45.494+(_tps*17* 1*isteps),
        longitude: -73.744+(_tps*17*11*isteps)
        /**/
    });
    });
}

static SIM908_DEFAULT_I2C_ADDRESS() {
    return 0x87;
}

static CHIP_ID1_SIM908() {
    return 0x156;
}

static CHIP_ID2_SIM908() {
    return 0x157;
}

static CHIP_ID3_SIM908() {
    return 0x158;
}

static CHIP_ID_SIM908() {
    return 0x160;
}
}

export default SIM908;

```

The Simulator page: sample.js

The simulator is mostly under control of the raspberry window. Reasons being is that console.log inside that simulated code works and shows something. Was valuable at the end to understand whats going on.

```

/*
 * IoT Hub Raspberry Pi NodeJS - Microsoft Sample Code - Copyright (c) 2017 - Licensed MIT
 * 2018-02-11 mr.fturi@gmail.com added SIM908 geopos simulator and basic SdDisk device
 * simulator
 */
const wpi = require('wiring-pi');
const Client = require('azure-iot-device').Client;
const Message = require('azure-iot-device').Message;
const Protocol = require('azure-iot-device-mqtt').Mqtt;
//const BME280 = require('bme280-sensor');
const SIM908 = require('sim908-sensor');
const SDCARD = require('SdCard-device');

const SIM908_OPTION = {
  i2cBusNo: 1, // defaults to 1
  i2cAddress: SIM908.SIM908_DEFAULT_I2C_ADDRESS() // defaults to 0x77
};

const SDCARD_OPTION = {
  i2cBusNo: 2, // defaults to 1
  i2cAddress: SDCARD.SDCARD_DEFAULT_I2C_ADDRESS() // defaults to 0x77
};

// fturi begin : Azure MQTT device
const connectionString = 'HostName=DataBoxHub.azure-devices.net;DeviceId=DataBoxDevice;SharedAccessKey=*****+tP2y16CPGbr6kCL0oMy4=';
// fturi end : Azure MQTT device
const LEDPin = 4;

var sendingMessage = false;
var messageId = 0;
var client,sensor,device;
var blinkLEDTIMEOUT = null;

function getMessage(cb) {
  messageId++;
  //console.log("#inf reading sensor data"+messageId);
  sensor.readSensorData()
    .then(function (data) {
      var _client=
        (
          (data.latitude >= 45.4930 && data.latitude <= 45.4950)
          &&
          (data.longitude >= -73.7640 && data.longitude <= -73.7240)
        );
      var _azure=
        (
          (data.latitude >= 45.5100 && data.latitude <= 45.5120)
          &&
          (data.longitude >= -73.5770 && data.longitude <= -73.5370)
        );
      _status=sensor.getState();
      if (_client && 2==_status)
        sensor.setState(_status=3);
      if (_azure && 5==_status)
        sensor.setState(_status=6);
      var _travelling=(!_client && !_azure);
      //console.log("#inf travelling="+_travelling);
      var _myJsonMsg=JSON.stringify({
        messageId: messageId,
        deviceId: 'DataBoxDevice',
        // fturi begin: add SIM908 data collection begin
        latitude: data.latitude,
        longitude: data.longitude,
        client:_client,
        azure:_azure,
        status:_status
      });
      //console.log("#inf myJsonMsg="+_myJsonMsg);
      cb(_myJsonMsg, _travelling);
    });
}

```

```

        // fturi end : add SIM908 data collection
    })
    .catch(function (err) {
        console.error('# KO_ Failed to read out sensor data: ' + err);
    });
}

function sendMessage() {
    if (!sendingMessage) { return; }

    getMessage(function (content, travellingAlert) {
        var message = new Message(content);
        message.properties.add('travellingAlert', travellingAlert.toString());
        //console.log('Sending message: ' + content);
        client.sendEvent(message, function (err) {
            if (err) {
                console.error('# KO_ cb:Failed to send message to Azure IoT Hub');
            } else {

                //fturi disable blinking
                if (travellingAlert)
                {
                    blinkLED();
                    //console.log("#inf cb:travelling!");
                }
                //console.log("#inf cb:Message sent to Azure IoT Hub");
            }
        });
    });
}

function upload_azure(){
    device.downloadFile().then(function(payload)
    {
        console.log("payload is"+payload+"data="+payload.data);
        console.log("payload name"+payload.data.originalname);
        var myJsonMsg=JSON.stringify({
            messageId: messageId++,
            deviceId: 'DataBoxDevice',
            azure:payload.data
        });
        console.log("file to upload is:"+myJsonMsg+",file is:"+payload.data.originalname);
        var message = new Message(myJsonMsg);
        message.properties.add('nature', 'dummy property');
        console.log('Sending message: ' + message);
        client.sendEvent(message, function (err) {
            if (err) {
                console.error('# KO_ cb:Failed to send message to Azure IoT Hub');
            }
        });
    })
    .catch(function (err) {
        console.error('# KO_ Failed to read out device data: ' + err);
    });
}

function onStart(request, response) {
    console.log('Try to invoke method start(' + request.payload + ')');
    sendingMessage = true;

    response.send(200, 'Successfully start sending message to cloud', function (err) {
        if (err) {
            console.error('[IoT hub Client] Failed sending a method response:\n' +
            err.message);
        }
    });
}

function onStop(request, response) {
    console.log('Try to invoke method stop(' + request.payload + ')');
    sendingMessage = false;

    response.send(200, 'Successfully stop sending message to cloud', function (err) {
        if (err) {
            console.error('[IoT hub Client] Failed sending a method response:\n' +
            err.message);
        }
    });
}

```



```

});
}

function react(cmd){
var status=sensor.getState();
result="ok";

switch (cmd){
  case "order":
    if (0===status)
    {
      status=1;
    }
    else
    {
      result="KO, your databox as already been ordered";
    }
    break;
  case "ship":
    switch (status)
    {
      case 0:
        result="KO, you need to order a databox first";
        break;
      case 1:
        status=2;
        break;
      case 2:
        result="war: your databox is already shipping";
        break;
      default:
        result="war: your databox is already there";
    }
    break;
  case "fill":
    status=3;
    switch (status)
    {
      case 3:
        status=4;
        break;
      case 4:
        break;
      default:
        result="KO: cannot fill your databox now";
    }
    break;
  case "return":
    if (4===status)
      status=5;
    else
      result="KO:cannot ship your databox now";
    break;
  case "upload":
    if (1==1)
    {
      upload_azure();
      status=1;
    }
    else
      result="KO: cannot upload your files now";
    break;
  default:
    result="KO, cmd unknown["+cmd+"]";
  }
}
/**/
sensor.setState(status);
return result+"("+status+")";
}

//We hijack the receive message to use it for upload/download to SDcard
function receiveMessageCallback(msg) {
  console.log("msg");

  var message = msg.getData().toString('utf-8');
  //console.log('Receive FTURIdb message: ' + message);
  var json=JSON.parse(message);

```

```

var cmd=json.cmd;
console.log('order is: ' + cmd);
if ("fill"==cmd)
{
  console.log("file is"+json.file);
  console.log("filename"+json.file.originalname);
  console.log("file size"+json.file.size);
  device.uploadFile(json.file);
}
console.log("result: "+react(json.cmd));
client.complete(msg, function () {

  //fturi add blinkLED when receiving message

});
/**/
}

function blinkLED() {
  // Light up LED for 500 ms
  if(blinkLEDDisabled) {
    clearTimeout(blinkLEDDisabled);
  }
  wpi.digitalWrite(LEDpin, 1);
  blinkLEDDisabled = setTimeout(function () {
    wpi.digitalWrite(LEDpin, 0);
  }, 500);
}

// set up wiring
wpi.setup('wpi');
wpi.pinMode(LEDpin, wpi.OUTPUT);
sensor = new SIM908(SIM908_OPTION);
sensor.init()
  .then(function () {
    sendingMessage = true;
  })
  .catch(function (err) {
    console.error(err.message || err);
  });

device = new SDCARD(SDCARD_OPTION);
device.init()
  .then(function () {
    sendingMessage = true;
  })
  .catch(function (err) {
    console.error(err.message || err);
  });

// create a client
client = Client.fromConnectionString(connectionString, Protocol);

client.open(function (err) {
  if (err) {
    console.error('[IoT hub Client] Connect error: ' + err.message);
    return;
  }

  // set C2D and device method callback
  client.onDeviceMethod('start', onStart);
  client.onDeviceMethod('stop', onStop);
  client.on('message', receiveMessageCallback);
  setInterval(sendMessage, 2000);
});

```

The library loader sample.js

```

import { Client, Message } from 'azure-iot-device'
import { traceEvent } from './telemetry.js';
import Protocol from './mqtt.js';
import wpi from './wiring-pi.js';
import codeFactory from './data/codeFactory.js';
import BME280 from './bme280.js';
import SIM908 from './sim908.js'; // fturi added sim908 library
import SDCARD from './SdCard.js'; // fturi added SdCard library
//import IOTHUB from './IotHub.js'; // fturi added EventHub library

//fturi 1/3: DO import above library

class ClientWrapper extends Client {
  constructor(transport, connStr, blobUploadClient) {
    window.azure_iot_device_client = super.constructor(transport, connStr, blobUploadClient);
    return window.azure_iot_device_client;
  }

  static fromConnectionString(connStr, transport) {
    window.azure_iot_device_client = super.fromConnectionString(connStr, transport);
    return window.azure_iot_device_client;
  }
}

class Sample {
  constructor() {
    this.runningFunction = null;
    if(!window.oldSetTimeout) {
      window.timeoutList = new Array();
      window.intervalList = new Array();

      window.oldSetTimeout = window.setTimeout;
      window.oldSetInterval = window.setInterval;
      window.oldClearTimeout = window.clearTimeout;
      window.oldClearInterval = window.clearInterval;

      window.setTimeout = function(code, delay) {
        var retval = window.oldSetTimeout(code, delay);
        window.timeoutList.push(retval);
        return retval;
      };
      window.clearTimeout = function(id) {
        var ind = window.timeoutList.indexOf(id);
        if(ind >= 0) {
          window.timeoutList.splice(ind, 1);
        }
        var retval = window.oldClearTimeout(id);
        return retval;
      };
      window.setInterval = function(code, delay) {
        var retval = window.oldSetInterval(code, delay);
        window.intervalList.push(retval);
        return retval;
      };
      window.clearInterval = function(id) {
        var ind = window.intervalList.indexOf(id);
        if(ind >= 0) {
          window.intervalList.splice(ind, 1);
        }
        var retval = window.oldClearInterval(id);
        return retval;
      };
      window.clearAllTimeouts = function() {
        for(var i in window.timeoutList) {
          window.oldClearTimeout(window.timeoutList[i]);
        }
        window.timeoutList = new Array();
      };
      window.clearAllIntervals = function() {
        for(var i in window.intervalList) {
          window.oldClearInterval(window.intervalList[i]);
        }
        window.intervalList = new Array();
      };
    }
    this.actualClient = null;
  }
}

```

```

stop() {
    window.clearAllIntervals();
    window.clearAllTimeouts();
    if(window.azure_iot_device_client) {
        window.azure_iot_device_client.close();
    }
}

//fturi 2/3 sed interpreter
run(option) {
    // a prefix of UUID to avoid name conflict, here just use a fix one
    const prefix = '76f98350';
    var replaces = [
        {
            src: /require\('wiring-pi'\)/g,
            dest: 'wpi'
        }, {
            src: /require\('azure-iot-device'\)\.Client/g,
            dest: 'Client'
        }, {
            src: /require\('azure-iot-device'\)\.Message/g,
            dest: 'Message'
        }, {
            src: /require\('azure-iot-device-mqtt'\)\.Mqtt/g,
            dest: 'Protocol'
        }, {
            src: /require\('bme280-sensor'\)/g,
            dest: 'BME280'
        }, { // fturi begin Added SIM908 and SDCARD declaration into raspberrypi
emulator language
            src: /require\('sim908-sensor'\)/g,
            dest: 'SIM908'
        }, { // fturi SDCARD declaration
            src: /require\('sdcard-device'\)/g,
            dest: 'SDCARD'
        }, { // fturi SDCARD declaration
            src: /require\('iohub-service'\)/g,
            dest: 'IOTHUB'
        }, { // fturi end Added SIM908 and SDCARD declaration into raspberrypi emulator
language
            src: /console\.log/g,
            dest: 'msgCb'
        }, {
            src: /console\.error/g,
            dest: 'errCb'
        }
    ];
    wpi.setFunc(option.ledSwitch);

    //fturi 3/3 Import constants
    try {
        traceEvent('run-sample');
        var src = codeFactory.getRunCode('index', replaces, prefix);
        this.runningFunction = new Function('replaces' + prefix, src);
        this.runningFunction({
            wpi: wpi,
            Client: ClientWrapper,
            Message: Message,
            Protocol: Protocol,
            BME280: BME280,
            SIM908: SIM908, // fturi added SIM908 constants
            SDCARD: SDCARD, // fturi added SDCARD constants
            IOTHUB: IOTHUB, //fturi added IOTHUB
            msgCb: option.onMessage,
            errCb: option.onError
        });
        // if (src.search(/^(?!\\\/).)*setInterval/gm) < 0) {
        //     option.onFinish();
        // }
    } catch (err) {
        traceEvent('run-error', { error: err });
        option.onError(err.message || JSON.stringify(err));
        option.onFinish();
    }
}
}

```

```
export default Sample;
```

GeoTracking

We tried to use the Azure geolocalisation, but it was missing important features in term of map customisation. On the other hand, google maps was perfect.

We created a simple geotracking node.js code to represent it.

At the end, I realised that the raspberry emulator wont be able to upload file to Azure directly as the react-native azure-storage is not finished. So I had to hopped on the geotracking to do the node.js uploading part.

Handling data was tortuous. Had to understand that the ".then function" is a very important features dues to acarn aysynchronous consideration (horrible implementation....)

```
// index.js
const path = require('path')
const express = require('express')
const exphbs = require('express-handlebars')

const app = express()

/**
var lastMessage=null;
var lastObject=null;
var firstObject=null;
var _lat1=48.858;
var _lon1=2.296;
var _lat2=48.859;
var _lon2=2.297;
**/

app.listen(3300)
app.engine('.hbs', exphbs({
  defaultLayout: 'DataBoxMain',
  extname: '.hbs',
  layoutsDir: path.join(__dirname, 'views/layouts')
}))
app.set('view engine', '.hbs')
app.set('views', path.join(__dirname, 'views'))

const azure=require('azure-storage');
const
blobconnectionstring='DefaultEndpointsProtocol=https;AccountName=databoxstorage;AccountKe
y=*****/Dt1l61YszErChnOYWD1PUx1CMA7w3VM0N5q2CUQ==;EndpointSuffix=core.windows.net';

//const multer=require('multer');
//npm install fs
const fs=require('fs');

//npm install multer
//npm install azure-storage
function upload_azure()
{
  var payload=lastObject.azure;
  console.log("#ok order to upload to azure");
  console.log("#file object is"+lastObject.azure);
  console.log("#inf filename is "+payload.originalname);

  //https://www.w3schools.com/nodejs/nodejs_uploadfiles.asp
  //https://www.ibm.com/developerworks/community/blogs/a509d54d-d354-451f-a0dd-
  89a2e717c10b/entry/How_to_upload_a_file_using_Node_js_Express_and_Multer?lang=en
  /*
  var storage = multer.diskStorage({
    destination: function (req, azure, cb) {
      cb(null, '/uploads/')
    },
    filename: function (req, azure, cb) {
```

```

        cb(null, azure.originalname)
    }
    */
}

//https://docs.microsoft.com/en-us/azure/storage/blobs/storage-nodejs-how-to-use-blob-storage
var blobSvc = azure.createBlobService(blobconnectionstring);

blobSvc.createContainerIfNotExists('databoxpayload', function(error, result, response){
    if(!error)
    {
        console.log("#ok container is available");
        fs.writeFile("./"+payload.originalname, payload.b64, 'base64',
function(err) {
            console.log(err);
        });
        console.log("#inf azure.b64 is"+payload.b64);
        // Container exists and is private
        //https://github.com/Azure/azure-storage-node/issues/22
        //https://azure.github.io/azure-storage-node/BlobService.html#createBlockBlobFromLocalFile__anchor
        var blobExist=0;
        blobSvc.listBlobsSegmented('databoxpayload', null, function(error, result, response){
            if(!error){
                blobExist=1;

//https://dmrelease.blob.core.windows.net/azurestoragejssample/samples/sample-blob.html
                console.log("#inf there is a blob"+result.entries[0])
                // result.entries contains the entries
                // If not all blobs were returned, result.continuationToken
has the continuation token.
            }
            else
            {
                console.log("#inf not blob exist.all cool")
            }
        });
        if (blobExist>0)
        {
            blobSvc.deleteBlobIfExists('databoxpayload', 'databoxfile',
function(error, result, response){
                if(!error){
                    // file uploaded
                    console.log("# cool, file is
uploaded. Mission accomplished")
                }
                else
                {
                    console.log("# cannot delete,
something went wrong:"+error);
                }
            }
        }
        blobSvc.createAppendBlobFromLocalFile('databoxpayload',
'databoxfile',"./"+payload.originalname,
function(error, result, response){
            if(!error){
                // file uploaded
                console.log("# cool,
file is uploaded. Mission accomplished")
            }
            else
            {
                console.log("# not cool,
cannot createsomething went wrong:"+error);
            }
        });
    }
    else
    {
        console.log("#KO cannot create container");
    }
});

```

```

}

function trackit()
{
  if (null==lastObject || null==lastObject["latitude"])
  {
    console.log("# war message is not geopos");
    if (null==lastObject["azure"])
    {
      console.log("#ko message is unknown"+lastObject)
    }
    else
    {
      upload_azure();
      lastObject=null;
    }
  }
  if (null==firstObject)
  {
    _lat1=48.858;
    _lon1=2.296;
    _lat2=48.859;
    _lon2=2.297;
    console.log("#inf 1.1 is null fturi")
  }
  else
  {
    myJson=firstObject;
    _lat1=myJson["latitude"];
    _lon1=myJson["longitude"];
    console.log("#inf 1.2 not null fturi")
  }
  if (null!=lastObject)
  {
    myJson=lastObject;
    _lat2=myJson["latitude"];
    _lon2=myJson["longitude"];
    console.log("#inf 2.1 not null fturi")
  }
  else
  {
    console.log("#inf 2.2 null fturi")
  }
}

app.get('/', (request, response) => {
  response.render('DataBoxMap', {
    lat1: _lat1,
    lon1: _lon1,
    lat2: _lat2,
    lon2: _lon2
  })
})

/***** MQTT section ****/

const EventHubClient = require('azure-event-hubs').Client;

var connectionString = 'HostName=DataBoxHub.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=*****MSZjFvxPNYRZkXUKIzAC+Mh9i0=';

var printError = function (err) {
  console.log(err.message);
};

var printMessage = function (message) {
  console.log('Message received: ');
  lastObject=message.body;
  if (null==firstObject)
    firstObject=lastObject;
  lastMessage=JSON.stringify(message.body);
  trackit();
  console.log(lastMessage);
  console.log('');
};

var client = EventHubClient.fromConnectionString(connectionString);

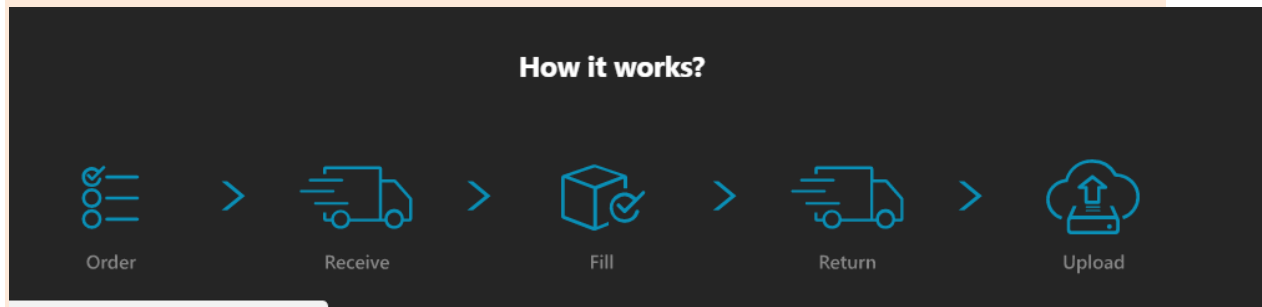
```



```
client.open()
  .then(client.getPartitionIds.bind(client))
  .then(function (partitionIds) {
    return partitionIds.map(function (partitionId) {
      return client.createReceiver('$Default', partitionId, { 'startAfterTime' :
Date.now()}).then(function(receiver) {
        console.log('Created partition receiver: ' + partitionId)
        receiver.on('errorReceived', printError);
        receiver.on('message', printMessage);
      });
    });
  })
  .catch(printError);
```

Console Part

Last part of the puzzle. I needed a DataBox console to simulate the order, ship ... cycle



We based the code on node.forms basic sample which was actually quite a poor sample (did not include handling of files)

Server.js

```

// server.js
'use strict';

const path = require('path')
const express = require('express')
const layout = require('express-layout')
const routes = require('./routes')
const app = express()

//valid form
const cookieParser = require('cookie-parser')
const session = require('express-session')
const flash = require('express-flash')

app.set('views', path.join(__dirname, 'views'))
app.set('view engine', 'ejs')

const bodyParser = require('body-parser')
const validator = require('express-validator')

const middleware = [
  layout(),
  express.static(path.join(__dirname, 'public')),
  bodyParser.urlencoded(),
  validator(),
  cookieParser(),
  session({
    secret: 'super-secret-key',
    key: 'super-secret-cookie',
    resave: false,
    saveUninitialized: false,
    cookie: { maxAge: 60000 }
  }),
  flash()
]
app.use(middleware)

app.use(bodyParser.json()); // to support JSON bodies
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/', routes)
  
```

```
app.use((req, res, next) => {
  res.status(404).send("Sorry can't find that!")
})

app.use((err, req, res, next) => {
  console.error(err.stack)
  res.status(500).send('Something broke!')
})

app.listen(3500, () => {
  console.log(`App running at http://localhost:3500`)
})
```

Route.js

You can see by the number of try and error that handling file upload is not an easy task. Most guide failed me.

```
// routes.js
'use strict';

const express = require('express')
const router = express.Router()

//validate
const { check, validationResult } = require('express-validator/check')

//sanitize
const { matchedData } = require('express-validator/filter')

/***** MQTT declaration *****/
/*
npm install azure-event-hubs --save
npm install azure-iot-device --save
npm install azure-iot-device-mqtt --save
*/
// following https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-node-node-getstarted#create-a-simulated-device-app
//const EventHubClient = require('azure-event-hubs').Client;

/*
//This time we use the device connection and not the hub
//otherwise we get: unkown function sendEvent
const connectionString = 'HostName=DataBoxHub.azure-
devices.net;DeviceId=DataBoxDevice;SharedAccessKey=*****tP2y16CPGbr6kCL0oMy4=';
const clientFromConnectionString = require('azure-iot-device-
mqtt').clientFromConnectionString;
const Message = require('azure-iot-device').Message;
*/
const Client = require('azure-iothub').Client;
var Message = require('azure-iot-common').Message;
const connectionString = 'HostName=DataBoxHub.azure-
devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=****FvxPNYRZkXUKIzAC+Mh9I0=';
const targetDevice="DataBoxDevice";
const client = Client.fromConnectionString(connectionString);

/***** Form implementation*****/

//file
//npm install multer
//npm install datauri
const multer=require('multer');
const upload = multer({ storage: multer.memoryStorage() })
const Datauri = require('datauri');
const fs=require('fs');

//https://www.sitepoint.com/forms-file-uploads-security-node-express/
router.get('/', (req, res) => {
  res.render('index')
})

router.get('/console', (req, res) => {
  res.render('console', {
    data: {},
    errors: {}
  })
})

router.post('/console',upload.single('photo'), [
  //validate
], (req, res) => {
  // Create a genre object with escaped and trimmed data.
  console.log("# inf request is",req.body);
}
```

```

const errors = validationResult(req)
if (!errors.isEmpty()) {
  return res.render('console', {
    data: req.body,
    errors: errors.mapped()
  })
}

//sanitize
const data = matchedData(req);
console.log('Sanitized: ', data);
// Homework: send sanitized data in an email or persist in a db

if (req.file) {
  console.log('Uploaded: ', req.file);
  //https://blog.stvmlbrn.com/2017/12/17/upload-files-using-react-to-
node-express-server.html
  //https://www.ibm.com/developerworks/community/blogs/a509d54d-d354-
451f-a0dd-89a2e717c10b/resource/BLOGS_UPLOADED_IMAGES/ScreenShot2016-11-29at14.58.43.png
  //https://github.com/expressjs/multer
  //https://www.manthanhd.com/2016/06/26/handling-file-uploads-in-
express/
  //https://github.com/expressjs/multer/issues/198#issuecomment-
327403675
  //https://github.com/expressjs/multer/issues/337#issuecomment-
251575772

  var multer_storage = multer.memoryStorage();
  var multer_file = multer({ storage : multer_storage
}).single('image');

  var datauri_image = function (req, res, next) {
    upload(req,res,function(err) {
      var datauri = new Datauri();
      if ('image/png'==req.file.mimetype)
        datauri.format('.png', req.file.buffer)
      else
        datauri.format('.jpg', req.file.buffer)
      cloundinary.uploader.upload(datauri.content,
        function(result) {
          if(result.public_id){
            //res.sendStatus(200)
          }else{
            res.sendStatus(500);
          }
        }
      );
    });
  };

  /*
  //https://www.thepolyglotdeveloper.com/2016/02/convert-an-uploaded-
image-to-a-base64-string-in-node-js/
  var fileInfo = [];
  var i=0;
  for(i = 0; i < req.file.length; i++) {
    console.log("dealing with file "+i)
    fileInfo.push({
      "originalName": req.files[i].originalName,
      "size": req.files[i].size,
      "b64": new
Buffer(fs.readFileSync(req.files[i].path)).toString("base64")
    });
    console.log("#inf file treated:"+i)
    var file_images=fileInfo[0];
  }
  */
  var file_image={
    "originalname": req.file.originalname,
    "size": req.file.size,
    "b64": req.file.buffer.toString("base64")
  };

  /*
  var multer_storage = //multer({storage:
multer.memoryStorage()
//}).single('file')

```

```

; //file being the file
attribute
    //var upload = multer({ storage: storage })
    var multer_file= multer({ storage : multer_storage }).single('file');
    /*
    //console.log("multer_storage="+JSON.stringify(multer_storage));
    //console.log("multer_file="+JSON.stringify(multer_file));
    //console.log("datauri_image"+datauri_image);
    //console.log("files_image"+files_image);
    console.log("file_image"+file_image);

    sendCmd(req.body,file_image);
    // Homework: Upload file to S3
    }
    else
        sendCmd(req.body)

    req.flash('success', 'Order taken in account')
    //res.redirect('/')
})
module.exports = router

/***** MQTT implementation *****/
client.open()

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);
    };
}

function receiveFeedback(err, receiver){
    receiver.on('message', function (msg) {
        console.log('Feedback message:')
        console.log(msg.getData().toString('utf-8'));
    });
}

function sendCmd(body,file)
{
    //https://www.codementor.io/codementorteam/how-to-use-json-files-in-node-js-85hndqt32
    //var myContent=JSON.parse(body);
    var myJSON=JSON.stringify(body);
    var myContent=JSON.parse(myJSON);
    console.log("# inf json is",myContent);
    if (myContent.hasOwnProperty('cmd'))
    {
        var cmd=myContent.cmd;
        console.log("# inf send cmd:[" +cmd+""]);
        if ("fill"==cmd)
        {
            var data = JSON.stringify({ deviceId: 'DataBoxDevice', cmd: cmd,file:
file});
            console.log("long data is file="+file.originalname);
        }
        else
            var data = JSON.stringify({ deviceId: 'DataBoxDevice', cmd: cmd});
        var message = new Message(data);
        //message.properties.add('temperatureAlert', (temperature > 30) ? 'true' :
'false');

        console.log("Sending message: " + message.getData());

        client.open(function (err) {
            if (err) {
                console.error('Could not connect: ' + err.message);
            } else {
                console.log('Service client connected');
                client.getFeedbackReceiver(receiveFeedback);
                //var message = new Message('Cloud to device message. ');
                //message.ack = 'full';
                message.ack = 'positive';
                message.messageId = "My Message ID";
                console.log('Sending message: ' + message.getData());
                client.send(targetDevice, message, printResultFor('send'));
            }
        });
    }
}

```

```

        }
    });

    /*
    client.sendEvent(message, printResultFor('send'));
    client.on('message', function (message) {
        console.log(message);
        client.complete(message, function () {
            console.log('completed');
        });
    });
    */
    /*
    var connectCallback = function (err,data) {
        if (err) {
            console.error('Could not connect: ' + err);
        } else {
            console.log('Client connected');
            var message = new Message('some data from my device');
            client.sendEvent(message, function (err) {
                if (err) console.log(err.toString());
            });

            client.on('message', function (msg) {
                console.log(msg);
                client.complete(msg, function () {
                    console.log('completed');
                });
            });
        }
    };

    //client.open(connectCallback);

    */
}
else
    console.log("# err body as no cmd in it");
}

```

Console.ejs

```

<form method="post" action="/console" novalidate enctype="multipart/form-data">
<table>
<tr><td>

</td><td>
<table border="0">
<tr><td>

</td><td>
<input type="radio" name="cmd" value="order" checked> Order DataBox
</td></tr><tr><td>

</td><td>
<input type="radio" name="cmd" value="ship"> ship databox
</td></tr><tr><td>

</td><td>
<input type="radio" name="cmd" value="fill"> fill databox
<div class="form-field">
<label for="photo">Photo</label>
<input class="input" id="photo" name="photo" type="file" />
</div>
</td></tr><tr><td>

</td><td>
<input type="radio" name="cmd" value="return"> return databox
</td></tr><tr><td>

</td><td>
<input type="radio" name="cmd" value="upload"> upload data
</td></tr></table>
</td></tr></table>

<div class="form-actions">
<button class="btn" type="submit">Send</button>
</div>
</form>

```


Index.ejs

```
<!-- views/index.ejs -->
<% if (messages.success) { %>
  <div class="flash flash-success"><%= messages.success %></div>
<% } %>

<h1>Databox</h1>
<a href="console">console</a>
```

Node.js & Native-React rants

The good, the bad & the ugly

The good: All C# example works perfectly and are solid. No crash, no nothing. Easy to produce, to debug and solid when manipulated

The bad: Node.js

Node.js is difficult to integrate with any modern GUI. You can run the program by alt-clicking on the right startup.js script or main script (you got to guess it) Executing the package.js for npm works too.

But executing is broken easily and often you have to restart the eclipse because of some weird error (in particular if you build when software is running).

There is CLI node_inspect command to debug it step by step. The debugger of Eclipse is not working because the debugger relies on V8 google VM (?) to debug it which is an abandoned software for years.

So basically there is no modern debugger available for node.js

Usefull links:

<https://github.com/nodejs/node/wiki/Using-Eclipse-as-Node-Applications-Debugger>

<https://code.google.com/archive/p/chromedevtools/>

<http://www.nodeclipse.org/usage>

Strange, recent talk about V8

https://medium.com/@nodejs_recipes/eclipse-node-js-error-when-debugging-failed-to-connect-to-standalone-v8-vm-connect-timed-out-35cb44f2603a

<http://mikethetechie.com/post/5541305522/installing-a-debugger-for-nodejs>

<https://www.w3resource.com/node.js/nodejs-console-logging.php>

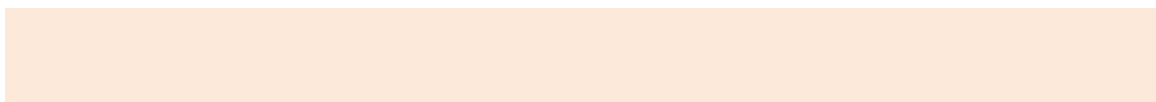
<http://www.nodeclipse.org/enide/2015/>

<https://atom.io/packages/node-debugger>

<https://www.npmjs.com/package/node-inspector>

<https://github.com/facebook/jest/issues/1652>

https://www.eclipse.org/community/eclipse_newsletter/2016/may/article3.php



The ugly: native-react

Native-React is not a set of extra node.js library. Its actually a kind of VM running on top of Node.js.

So, from Native-React you cannot do the following:

- calling any Node.js library using require Basically anything usefull
- communicate with Node.js underneath !
- have access to any console. So forget about printing out any debugging information
- it will fail any debugger, including the command-line debugger, because native-react is interpreting the js file on top of it

So, what can you do ? Well you can use the "native" package. That's it. And of course, the native-react package are scarce.

The raspberry-pi emulator is written in react. And it was a painfull realisation to be so limited with that react JAIL. You cannot escape it.

Usefull links on React:

<https://github.com/facebookarchive/node-haste/pull/69>

<https://github.com/philikon/ReactNativify>

To bad: not working yet. Because of that I could not upload directly to the storage from the emulator. I had to hop on the geotracker node.js to upload the file.

<https://github.com/letrungkien211/react-native-azure-storage>

GitHub repositories

The three projects are:

Raspbery emulator (branch from Azure code):

<https://github.com/sysarchitek/DataBox.raspberry-pi-web-simulator>

The console:

<https://github.com/sysarchitek/DataBox.Console>

The Tracker and Storage hopper-uploader

<https://github.com/sysarchitek/DataBox.geotracking>

The main documentation project is there:

<https://github.com/sysarchitek/DataBox-IOT-Emulator>

Youtube videos

YouTube Links:

3 (sorry :-*) Min: <https://youtu.be/yg7ieQeYTS0>

15 Min: <https://youtu.be/Es7b7VV7s78>