Mesay Magicho

Internet of Things: Smart Home Device

Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics

Thesis

8 March 2018

Author Title Number of Pages Date	Mesay Magicho Internet of Things: Smart Home Automation 33 pages + 2 appendices
Degree	Bachelor of Engineering
Degree Program	Electronics Engineering
Professional Major	
Instructors	Matti Fischer, Principal Lecturer

The purpose of this project aims to design and implement a smart home device that can be deployed into residential and office settings to control temperature and humidity to a desired level. At the same time this helps us to make better use of energy and improve the well-being of the residents.

To achieve the desired results, a Texas Instruments Sensor for real time measurement and Raspberry Pi 3 gate way, along with IMB Watson platform for Internet of Things were applied. The theoretical part will discuss and describe operations of the smart home device elements in details.

This thesis uses open source community resources for its own specific Smart Home Device design and can be used for further developments of similar open source projects. There are unlimited number of applications which can be done using a wireless sensor and Raspberry Pi. A successful and better devices can be achieved using the latest products. This project can be helpful to understand and learn the core of Internet of Things and Smart Home Automation.

Keywords	IoT, Sensortag, Raspberry pi 3, Temperature, Humidity , Node Red, IBM Cloud

Contents

List of Abbreviations

BLE	Bluetooth Low Energy
DevPaks	Development Packages
IaaS	Infrastructure as a Service
IoT	Internet of Things
LED	Light Emitting Diode
MEMS	Micro Electro-Mechanical Systems
MCU	Multipoint Control Unit
PaaS	Platform as a Service
RF	Radio Frequency
SD	Secure Digital
SSH	Secure Shell
TI	Texas Instruments
USB	Universal Serial Bus
VNC	Virtual Network Computing

1 Introduction

In less than a couple of years more than 50 Billion Internet of Things (IoT) devices will be connected to the Internet. And smart home devices will overtake smart phones by 2021 as a share of deployed connected/ IoT devices [1]. Smart home or a residence area can be defined as a home that uses programmable electronics control and sensors that read real time measurements and regulate heating, humidity, lighting etc. in a way that responds to indoor climate conditions, in order to conserve energy or optimize the well-being of its residents.

In this thesis project we go through detailed design and implementation of smart home device to control temperature and humidity level for a desired residential or office area. The smart home device can detect the temperature and humidity level in the desired area and use a Bluetooth Low Energy technology to communicate with a Raspberry pi 3 as a gate way to IBM cloud which is used as IoT platform service provider.

The smart home device consists of Texas Instrument (TI) SensorTag CC2650, which can detect and respond to the resident area climate conditions and Raspberry Pi 3, which is used to read values from sensor using Node Red on the Raspberry Pi, using BLE technology, and send commands to IoT platform on IBM cloud server. This customize the values according to end user’s specific needs.

The thesis will cover theoretical background and working principles of each element of the smart home device. Special attention is given to the implementation and working principle of the device. Procedural instructions are provided for set up and operation of each device elements. The later sections of the thesis presents the results with pictures, conclusions and recommendations for future development.

2 Smart Home Device Elements

2.1 SensorTag

The TI Bluetooth Smart SensorTag CC2650 is a new type of smart device that packs many different sensors into small battery-powered unit. We can use the SensorTag for a wide variety of IoT applications. It is perfect to use the device for quick prototyping, for example new kinds of IoT applications, various sensor-based alarms, interaction devices that use the accelerometer/gyroscope, game controllers, smartwatch prototypes, sports sensors and much more. For all these, mobile applications play a central role to display information to the end user.

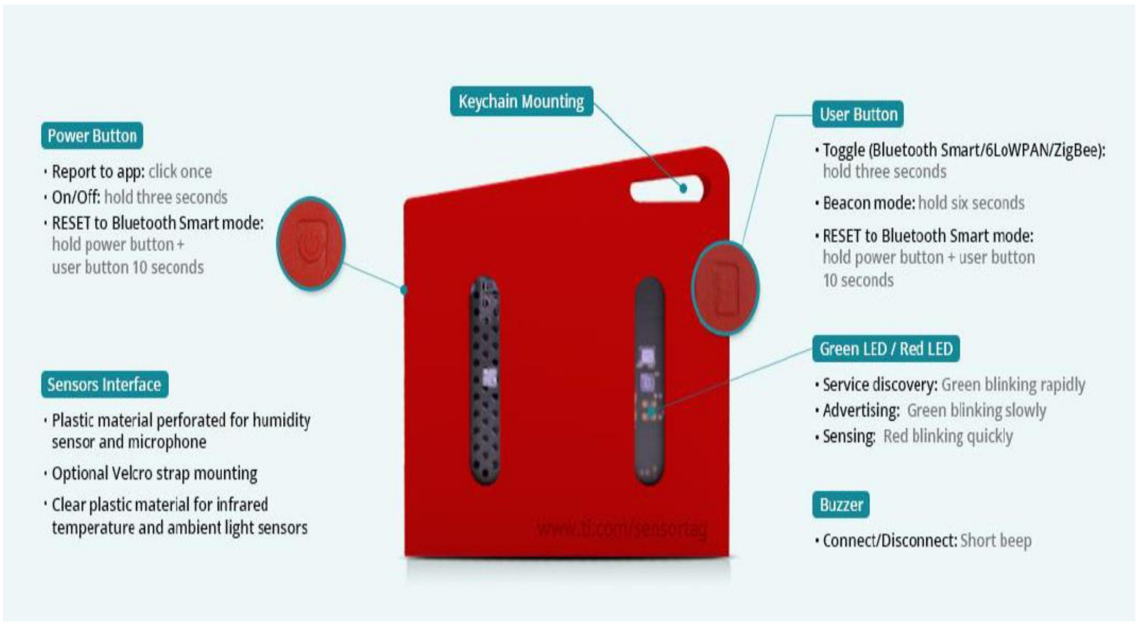

Keychain Mounting
Power Button
• Report to app: click once
• On/Off: hold three seconds
• RESET to Bluetooth Smart mode: hold power button + user button 10 seconds
User Button
• Toggle (Bluetooth Smart/6LoWPAN/ZigBee): hold three seconds
• Beacon mode: hold six seconds
• RESET to Bluetooth Smart mode: hold power button + user button 10 seconds
Sensors Interface
• Plastic material perforated for humidity sensor and microphone
• Optional Velcro strap mounting
• Clear plastic material for infrared temperature and ambient light sensors
Green LED / Red LED
• Service discovery: Green blinking rapidly
• Advertising: Green blinking slowly
• Sensing: Red blinking quickly
Buzzer
• Connect/Disconnect: Short beep
www.ti.com/sensortag

Figure 1. SensorTag [2]

The new SensorTag now includes 10 low-power MEMS (Micro Electro-Mechanical Systems) sensors in a tiny red package (figure 1). And it is expandable with DevPacks (Development Packages) to make it easy to add your own sensors or actuators.

2.1.1 Description

The Sensortag CC2650 device is a wireless Multi control unit (MCU) targeting Bluetooth, ZigBee® and 6LoWPAN, and ZigBee RF4CE remote control applications.

The device is a member of the CC26xx family of cost-effective, ultralow power, 2.4-GHz RF devices. Very low active RF and MCU current and low-power mode current consumption provide excellent battery lifetime and allow for operation on small coin cell batteries and in energy-harvesting applications [2].

The CC2650 device contains a 32-bit ARM Cortex-M3 processor that runs at 48 MHz as the main processor and a rich peripheral feature set that includes a unique ultralow power sensor controller (figure 2). This sensor controller is ideal for interfacing external sensors and for collecting analog and digital data autonomously while the rest of the system is in sleep mode. Thus, the CC2650 device is ideal for applications within a whole range of products including industrial, consumer electronics, and medical.

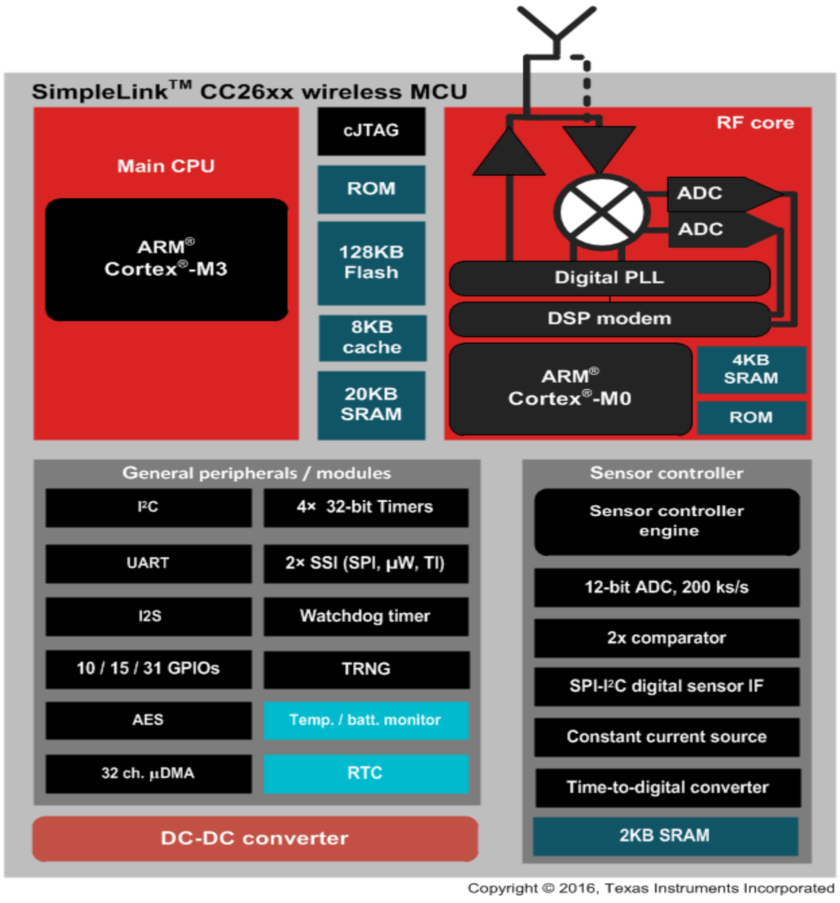


Figure 2. Functional Diagram [2]

The Bluetooth Low Energy controller and the IEEE 802.15.4 MAC are embedded into ROM and are partly running on a separate ARM Cortex-M0 processor. This architecture improves overall system performance and power consumption and frees up flash memory for the application [3].

2.1.2 Operation

During normal operation the sensortag will start to advertise and indicate with green LED blinking. On start-up, the SensorTag advertises with a 100 millisecond interval indicated by the green LED blinking at a 1 Hz rate. Advertising can be stopped/started by pressing the power button (right). The SensorTag must be advertising for the device to be discoverable by a smart phone or any other Bluetooth Smart central device [4]. The central device (e.g. Raspberry, smartphone) can only connect to a SensorTag that is advertising.

In the advertising state the central device can scan, discover and establish connection based on user defined connection parameters. To conserve battery power, the SensorTag uses Limited Advertising and will advertise for approximately 3 minutes after pressing the power button. When the SensorTag is not advertising or established in a BLE connection, the on board sensors are powered down into a low power state and not collecting sensor data.

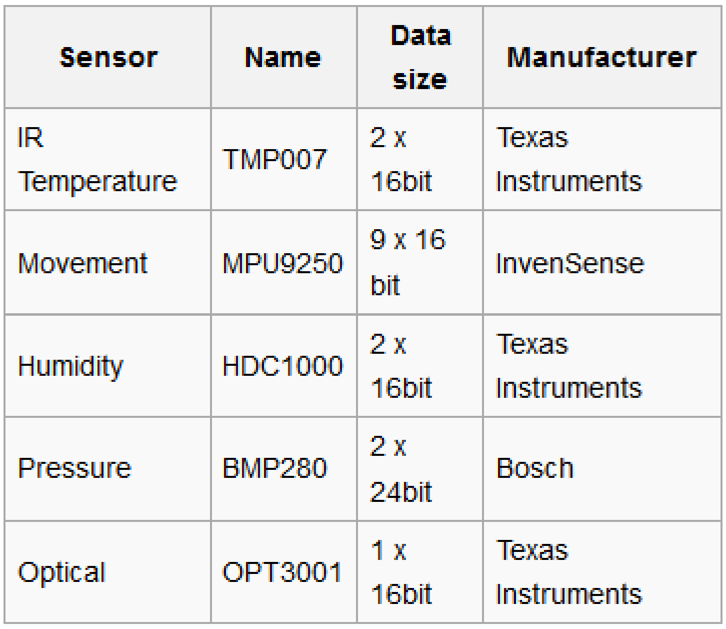
When connection established by central device and the sensors can then be configured to provide measurement data. To obtain the data, the corresponding sensor must first be activated, which is done via a Characteristic Value write to appropriate service. The most power efficient way to obtain measurements for a sensor is to

- Enable notification
- Enable Sensor
- When notification with data is obtained at the Master side, disable the sensor (notification still on though)

2.1.3 Sensors

The sensortag has IoT connectivity support for the following sensors

- IR Temperature, both object and ambient temperature
- Movement, 9 axis (accelerometer, gyroscope, magnetometer)
- Humidity, both relative humidity and temperature
- Barometer, both pressure and temperature
- Optical, light intensity



Sensor	Name	Data size	Manufacturer
IR Temperature	TMP007	2 x 16bit	Texas Instruments
Movement	MPU9250	9 x 16 bit	InvenSense
Humidity	HDC1000	2 x 16bit	Texas Instruments
Pressure	BMP280	2 x 24bit	Bosch
Optical	OPT3001	1 x 16bit	Texas Instruments

Figure 3. Sensor Overview [4]

The individual sensors require varying delays to complete measurements. Recommended setting is 100 ms for fast movement data updates, if only the environmental data is of interest, the interval can be slower to increase battery life time. Selecting a sensor measurement period that is faster than the connection interval will result in unnecessary battery power consumption since sensor reading can only be sent at scheduled connection events. Notifications can be stopped and the sensors turned on/off.

2.2 Raspberry Pi

The Raspberry Pi 3 is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries [5]. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not include peripherals (such as keyboards, mice and cases). However, its powerful programming features, the small size and price affordability makes it a perfect choice for our smart home device design and act as a core element to communicate with sensortag and collect data.

Figure 4. Raspberry Pi 3 [5]

2.2.1 Setup

To set up Raspberry Pi for IoT project it is essential to have an SD card of size 8GB preinstalled with Raspbian Jessie. For a display we use HDMI/DVI monitor and any TV would work fine. Any standard USB keyboard and mouse will work with Raspberry Pi and the Pi will be powered by a USB Micro power supply like most standard mobile phone chargers.

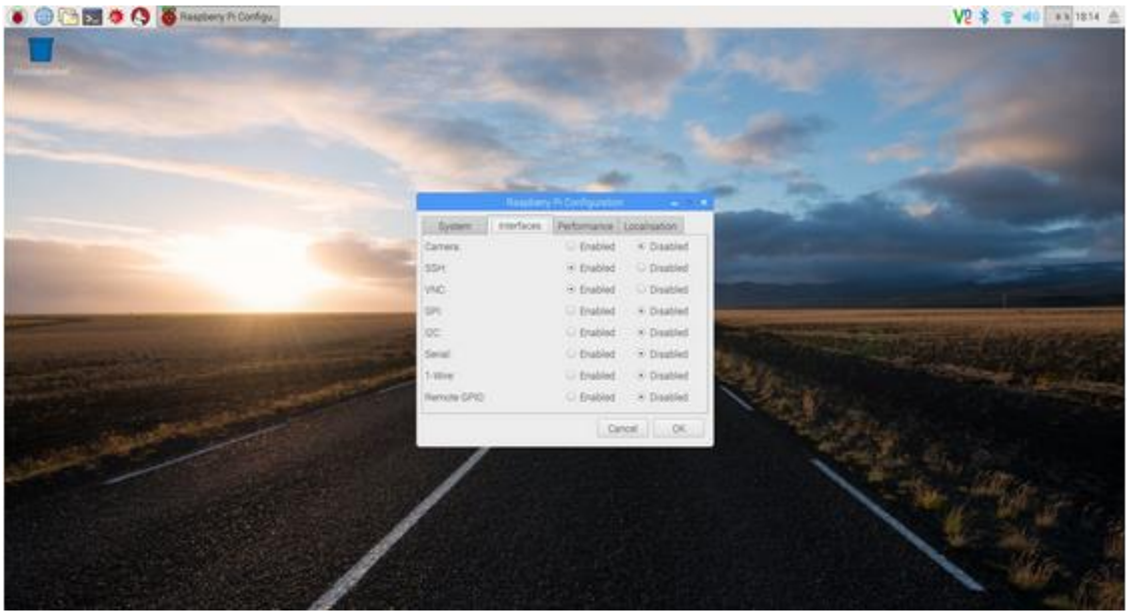
On a formatted SD card we download and install Raspbian Jessie which is a free operating system that makes our Raspberry Pi run on windows user. Etcher software were used to write the images on SD card.

Etcher is a graphical SD card writing tool that works on Mac OS, Linux and Windows, and is the easiest option for most users. Etcher also supports writing images directly from the zip file, without any unzipping required.

2.2.2 Remote Raspberry Pi

For this IoT project it is important to control and run Raspberry Pi from another computer, such as a PC or Mac to send and access data. There are different ways to do that, the most preferred use specifically for convenient and technology support are Virtual Network Computing (VNC) and Secure Shell (SSH).

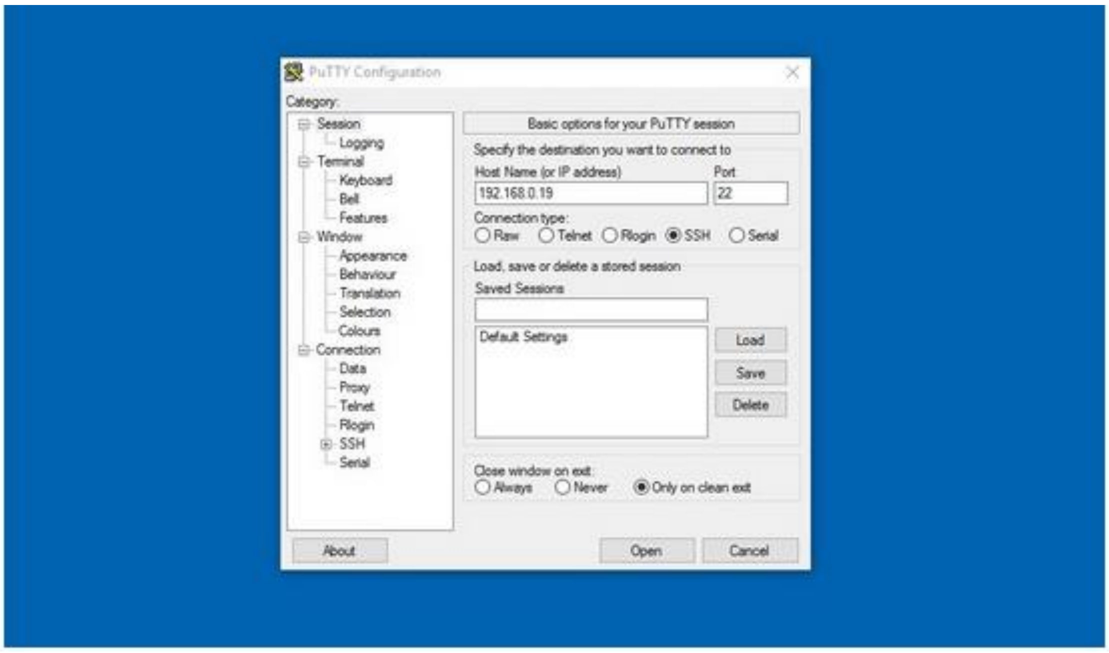
By default, VNC Server from RealVNC gives you direct control over your Raspberry Pi, just as though you were sitting in front of it. This is great for controlling lightweight Raspbian projects, such as IoT builds. As we'll see in the steps, Raspbian Jessie with PIXEL includes Server by default. However, you will need to enable it yourself. From then on, Server will be loaded every time you switch on your Raspberry Pi [6].



Camera	Enabled [✓ Disabled]
SSH	[✓ Enabled] Disabled
VNC	[✓ Enabled] Disabled
SPI	Enabled [✓ Disabled]
I2C	Enabled [✓ Disabled]
Serial	Enabled [✓ Disabled]
1-Wire	Enabled [✓ Disabled]
Remote GPIO	Enabled [✓ Disabled]

Figure 5. Virtual Network Computing

Secure Shell is handy if you want to quickly connect to your Raspberry Pi from a terminal on another computer. It’s also ideal for lightweight distro installations that don’t have an interface. It’s especially useful when creating Internet of Things (IoT) projects, as these may be embedded and not require a desktop.



Host Name (or IP address): 192.168.0.19
Port: 22
Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial
Saved Sessions: Default Settings
Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]

Figure 6. Secure Shell (SSH) using Putty

2.3 IBM Cloud

IBM offers hardware platforms for cloud computing. IBM Developer Cloud platform is IBM’s solution for the cloud. IBM cloud is a platform as a service solution, as well as Infrastructure as a Service. IBM cloud service extensions provide functionality that is ready to use by web or mobile applications. The predefined services include database, messaging, push notifications and elastic caching for web applications [7].

IBM cloud is an open standards based cloud computing platform as a service PaaS (Platform as a Service) with infrastructure as a service (IaaS) to develop, build, test, deploy, run and manage the application in the cloud. In IaaS (Infrastructure as a Service) the hardware, storage and network can be managed on the cloud (figure 7).

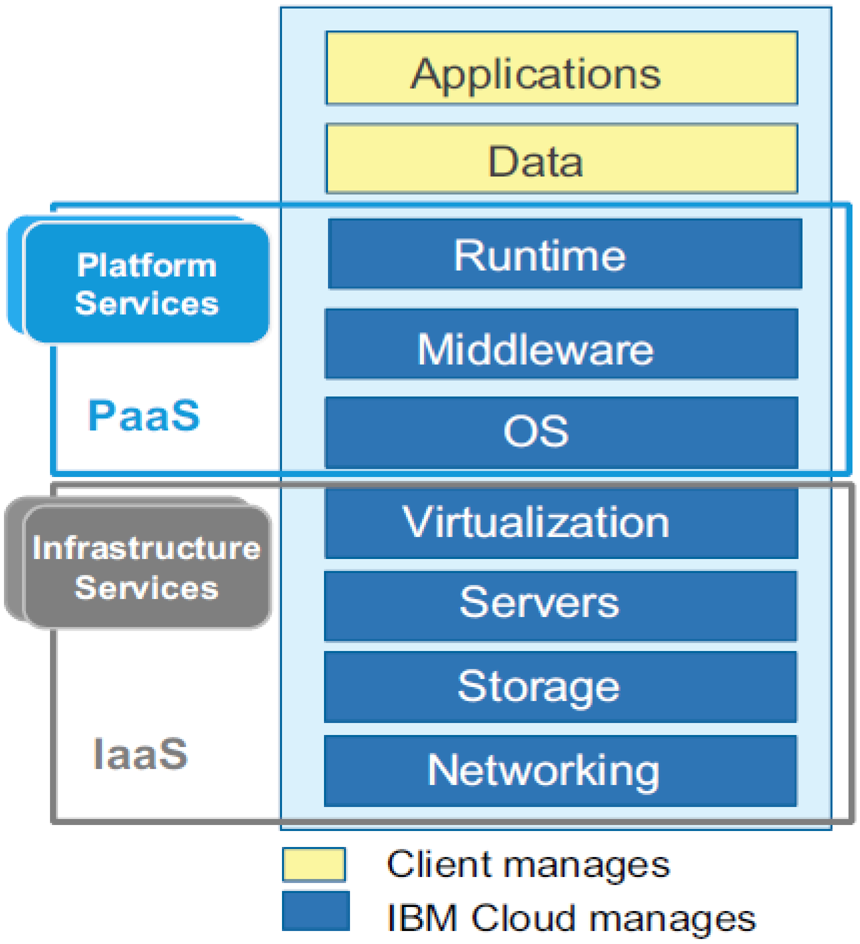


Figure 7. IBM Cloud platform [7]

The PaaS model includes services that build on IaaS services. They add value to the IaaS services by providing a platform on which the cloud users can provision their own applications, or conduct application development activities. The user does not need to manage the underlying cloud infrastructure (network, storage, operating systems), but can control configuration of the provisioned platform services.

The IaaS model is the simplest for cloud service providers to provision. It can include Processing, Storage and Network. As an IaaS user we can deploy and run our chosen software, including operating systems and applications. We do not need to manage or control the underlying cloud infrastructure, but we have control over the operating systems, storage, and deployed applications.

2.3.1 Creating IBM Cloud Account

IBM Cloud has a free lite service and run times to build our applications, to store data and analyze them. IBM account can be created on the website for free of cost with no time restriction. The free account offers 256 MB of instantaneous cloud foundry runtime memory, access to usage capped plans for selected services, such as API connect, Watson conversion, Watson discovery and for our project Internet of Things.

After registration we can be able to connect our devices to the IBM Watson IoT platform in IBM cloud to watch live sensor data stream into a sample dashboard. IBM cloud offer a detailed procedure to make device types in the platform. IBM offers two approaches for creating an IoT organization for our projects.

- Cloud IoT Boilerplate that will create a node-red application that is bound to our new IoT organization (figure 8)

- IoT Cloud tile to create a new instance of IoT organization

We use the first approach to create our device types under IBM given organization name.

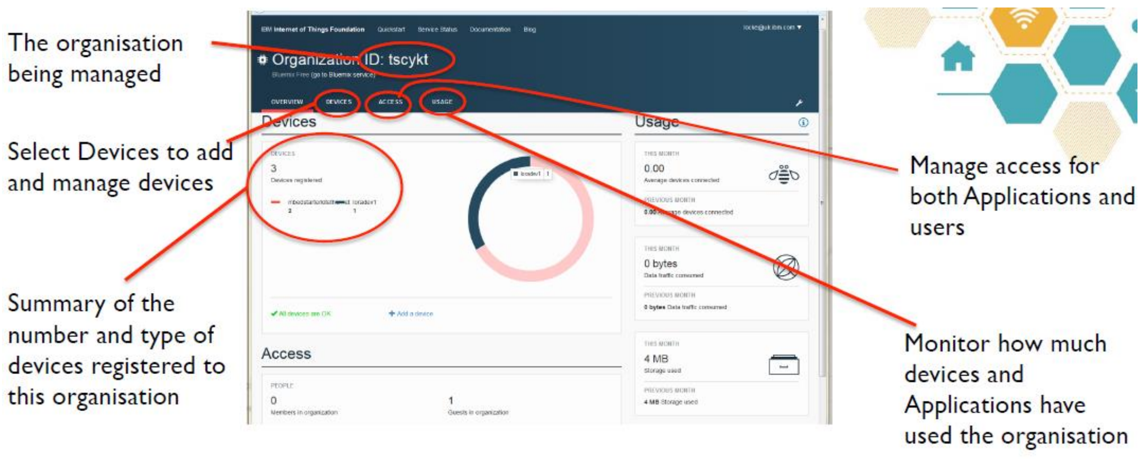

The organisation being managed
Select Devices to add and manage devices
Summary of the number and type of devices registered to this organisation
Manage access for both Applications and users
Monitor how much devices and Applications have used the organisation
Organization ID: tscykt
OVERVIEW
DEVICES
ACCESS
USAGE
Devices
Usage
Access

Figure 8. IBM Cloud IoT dashboard overview [8]

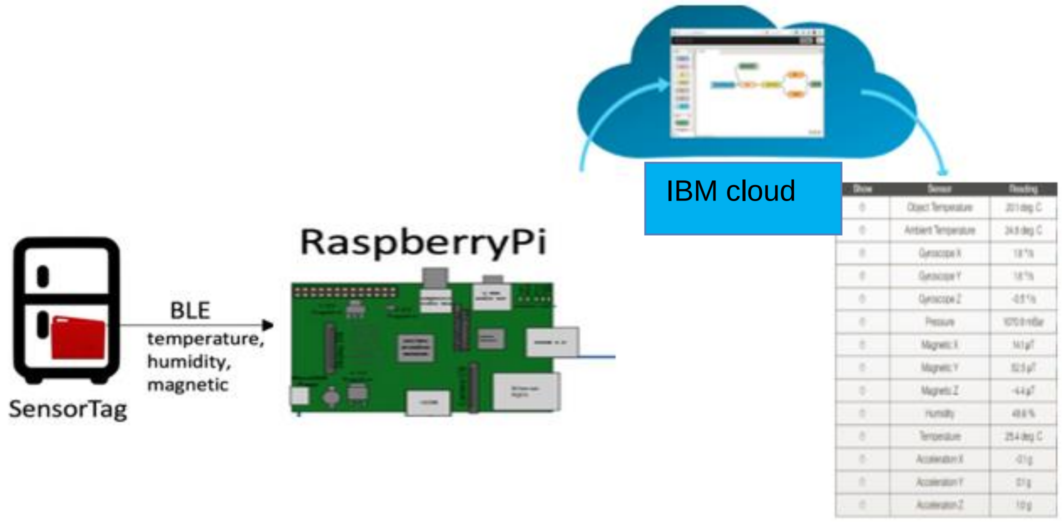
The platform offers us to create and add devices based on our preference. This make IBM cloud an ideal tool for our project.

3 Design of Smart Home Device

The Smart home device consists of Sensortag CC2650 Bluetooth sensor, Raspberry Pi 3 and IBM cloud platform (figure 9). The sensortag used to collect real time temperature and humidity readings from the surrounding. The collected data send to Raspberry pi using LE Bluetooth connection. Raspberry Pi used as a gate way to send those data’s to IBM Cloud platform. In the following procedure we see how to connect our elements and send the sensor data to IoT platform.

Elements needed

- IBM cloud account
- Raspberry Pi 3
- TI Sensortag CC2650



Show	Sensor	Reading
	Object Temperature	20.1 deg C
	Ambient Temperature	24.8 deg C
	Gyroscope X	1.8 °/s
	Gyroscope Y	1.8 °/s
	Gyroscope Z	-0.5 °/s
	Pressure	1010.8 mBar
	Magnetic X	14.1 µT
	Magnetic Y	52.5 µT
	Magnetic Z	-4.4 µT
	Humidity	48.8 %
	Temperature	25.4 deg C
	Acceleration X	-0.1 g
	Acceleration Y	0.1 g
	Acceleration Z	1.0 g

Figure 9. Ingredients for Smart Home Device

3.1 Overview

This instruction will lead us how to easily connect Sensortag to our Pi, using Bluetooth, and send their sensor data to IoT Platform (figure 10).

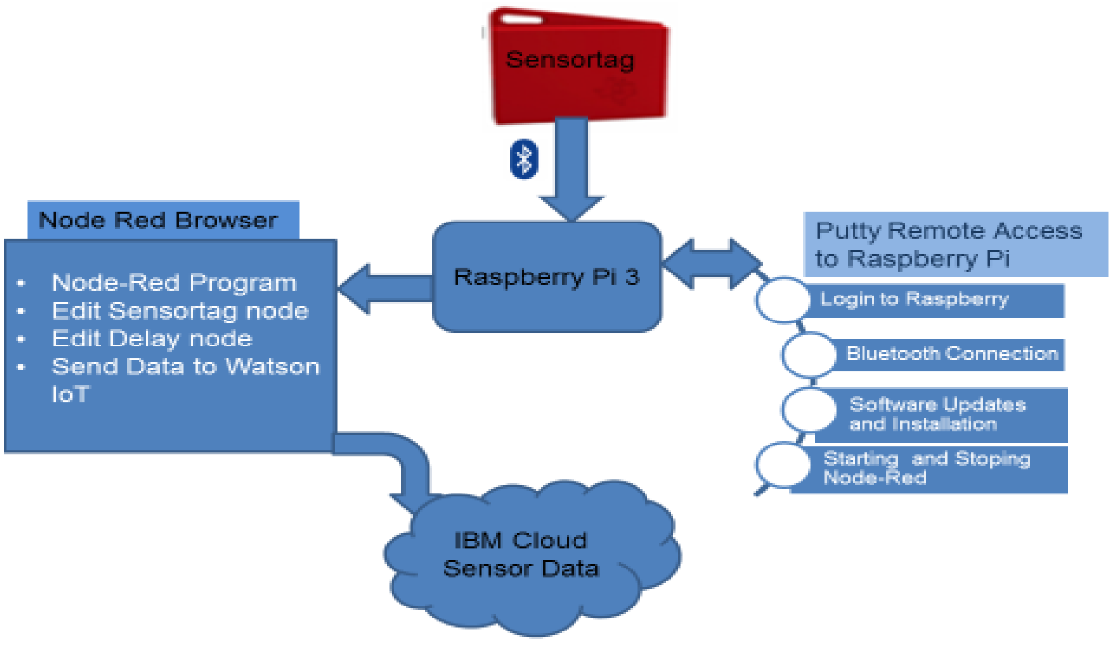


Figure 10. Flow diagram for programming and communication

The instruction uses a python script to scan for, connect to, and read sensor data from the sensortag, and collects this data into Node-RED using a daemon node (called node-red-node-daemon) which can run a script and collect its output into a node red flow. This flow will convert the string into a JSON object to send to IoT Platform, with the Pi connected as a gateway and the data being sent from a named device through the gateway.

The configuration of the node will allow us to give the sensortag specific names to choose (based on the tag's unique Bluetooth address, called a uid) which are sent with the data into IoT Platform. The script can also select which specific sensors to read on the sensortag, and how quickly to read them, leaving the unused sensors powered down, conserving battery power. If you don't choose to make a selection of devices or sensors, the script defaults to automatically connecting with all active sensortag sensors it collects, and repeatedly reads their data.

3.2 Installing required software’s

In this section we will install all the necessary software for the raspberry and see how to do that.

3.2.1 Connecting Using Putty to Raspberry Pi 3

In this recipe I will connect to my Pi from my laptop using Putty for command prompt and using a browser to access Node Red. If we are using a keyboard/monitor connected to our Pi, we can adapt the instructions below by starting a terminal and browser on the Pi directly.

Whenever we start working with a new Raspbian image, it’s good practice to make sure it us updated – this can take a while but ensures more predictable results. Or we could skip this step and come back only if we have problems later on in the recipe.

Run the following commands at the command prompt:

- `sudo apt-get update /*update new version of raspbian Jessie`
- `sudo apt-get upgrade /*upgrade to new version`
- `sudo apt-get dist-upgrade`

3.2.2 Node Red Installation

The node-red-node-sensortag allows connecting a single Sensortag to your Pi to connect to IoT Platform. To ensure node-red installation is up to date there are two methods to proceed with Node-RED.

The first one is, the default or current Node-RED setup, the only thing need to be done is if npm (the package manager for JavaScript) isn’t installed then install npm to discover packages of reusable code and assemble them in powerful new ways, using the following commands:

- `sudo apt-get -y install -y npm –upgrade /*upgrade the existing npm`

- `npm i -g npm@2.x /*install npm`

Or, you can upgrade to latest node.js and Node-RED – but note this may affect any custom nodes installed before and will get the latest usability improvements using this update. There are plenty of warnings, but these steps and the recipe works fine. This update also installs npm. At the command prompt:

```
update-nodejs-and-nodered
```

The most recent run through this recipe used the latest Node-RED and ensure NodeRed has been run.

Nod-RED is already installed in Raspbian. This recipe will add a custom node, and it can only do that if Node-RED has been run at least once because this creates a folder ~/.node-red in home folder.

Run the command:

```
node-red-start /*this will start the node red
```

When the server finishes loading, then it can test Node-RED accessing the user interface using the browser point to the address http://<iprasberrypi>:1880

Now the Node-RED server have to stop in the terminal window by typing ctrl+C and then type the command:

```
node-red-stop /*stop node red running on raspberry
```

After this two important nodes installation are needed on Node-RED to connect sensortag. The vanilla Node-RED (primarily installed on Raspberry) installation does not include two nodes that complete this recipe to connect the sensortag with Raspberry Pi and IoT platform. The first one is node-red-bluemix-nodes which provide a node to connect to IoT Platform, and the second one is node-red-node-daemon, which provide the node to run a python script to connect sensortag.

Run the following commands:

- `cd ~/.node-red /*destination file to install`
- `npm install node-red-bluemix-nodes /*connect to IoT cloud`
- `npm install node-red-node-daemon /*run python script`

Then install bluez, bluepy and other dependencies. Bluez provides utilities to access Bluetooth Low Energy (BLE) devices (such as the CC2650 Sensortag) on the pi. Bluepy provides a python interface to bluez.

At the command prompt:

- `sudo apt-get -y install bluez build-essential libglib2.0-dev libdbus-1-dev python-dev`
- `sudo pip install bluepy`

In some occasions pip command might fail, pip is a package management system used to install and manage software packages written in Python. Which might be due to a conflict between newer/older versions of the python library requests. Seems like an older (incompatible) version of requests can get installed when a python package requires that older version, and this breaks pip. To fix this problem, do the following commands then retry the pip command:

- `sudo apt-get remove python-pip /*remove older Python-pip`
- `sudo easy_install -U pip /*install new version pip`

- `sudo pip install –upgrade requests /*upgrades to the new version`

NOTE: When this recipe was created Raspbian Jessie was used and the above commands include libopenobex1. Which might not be essential in later versions of Raspbian.

After that installation and upgrading it have to be reboot and reconnect to complete the recipe. At the command prompt:

```
sudo reboot/*this reboot the raspberry
```

Copy python script to /home/pi and save. After the reboot completes, connect to Raspberry pi again then download and extract the python script sensortagcollector.py and copy it to home folder on Raspberry pi – /home/pi

Download this file directly using the following commands:

- `wget https://ibm.box.com/shared/static/pwkgxejzk92ktjn53owm1ka6bom9o0oq.zip -O sensortagcollector.zip`

- `unzip sensortagcollector.zip`

This folder have to be in the same folder with Node-RED flow otherwise it will not work because the daemon node expects the file to be in home folder. The daemon node have to be edited with the path to put the file. Probably simplest to first get everything working with the script file in the location indicated here, then later move it and update the daemon node.

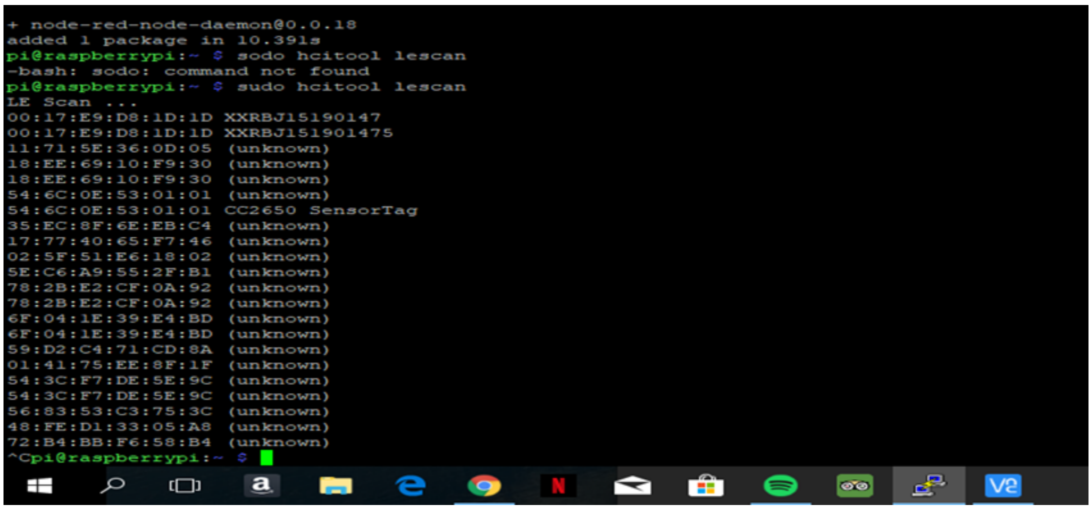
3.3 Testing the Bluetooth Connection

The data from Sensortag is transmitted to Raspberry Pi using Low Energy Bluetooth connection. And we need to check the connection before we create a node red flow if something isn't working. We find out the cause of the problem this way quickly and build confidence that our Sensortag are working before we get in to node red. This is also a useful test that our Sensortag battery is not flat [8].

At the command prompt, run:

- `sudo hcitool lescan /*Raspberry scanning Bluetooth device`

Now switch on the sensortag by pressing the power button and see the green light flashes and we should see the ID being printed on the command window too (figure 11).



```
+ node-red-node-daemon@0.0.18
added 1 package in 10.391s
pi@raspberrypi:~ $ sodo hcitool lescan
-bash: sodo: command not found
pi@raspberrypi:~ $ sudo hcitool lescan
LE Scan ...
00:17:E9:D8:1D:1D XXRBJ15190147
00:17:E9:D8:1D:1D XXRBJ151901475
11:71:5E:36:0D:05 (unknown)
18:EE:69:10:F9:30 (unknown)
18:EE:69:10:F9:30 (unknown)
54:6C:0E:53:01:01 (unknown)
54:6C:0E:53:01:01 CC2650 SensorTag
35:EC:8F:6E:EB:C4 (unknown)
17:77:40:65:F7:46 (unknown)
02:5F:51:E6:18:02 (unknown)
5E:C6:A9:55:2F:B1 (unknown)
78:2B:E2:CF:0A:92 (unknown)
78:2B:E2:CF:0A:92 (unknown)
6F:04:1E:39:E4:BD (unknown)
6F:04:1E:39:E4:BD (unknown)
59:D2:C4:71:CD:8A (unknown)
01:41:75:EE:8F:1F (unknown)
54:3C:F7:DE:5E:9C (unknown)
54:3C:F7:DE:5E:9C (unknown)
56:83:53:C3:75:3C (unknown)
48:FE:D1:33:05:A8 (unknown)
72:B4:BB:F6:58:B4 (unknown)
^Cpi@raspberrypi:~ $
```

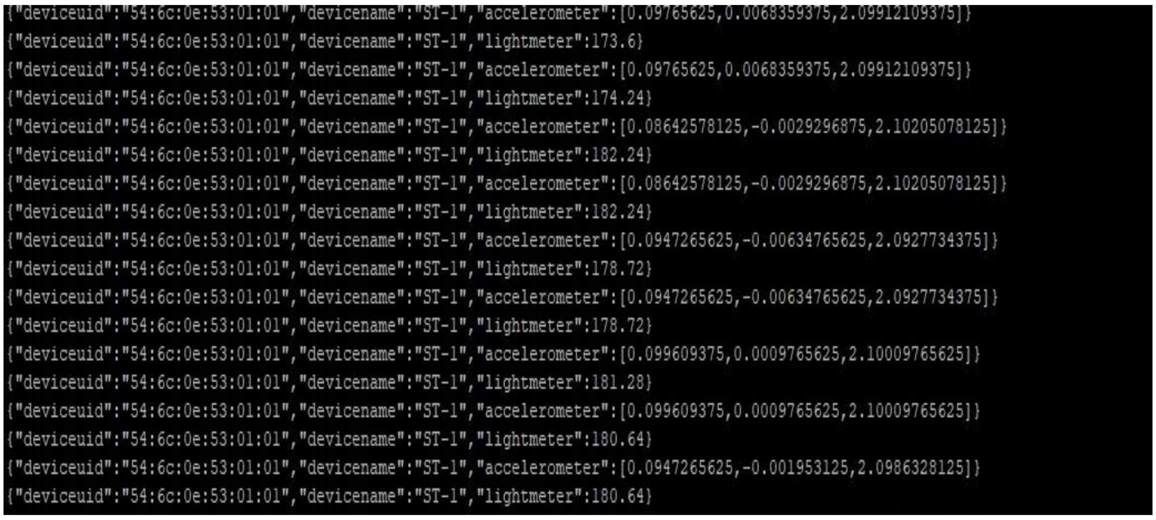
Figure 11. Bluetooth Connection between Sensortag and Raspberry pi [screen shot]

Now that our Sensortag is visible on Bluetooth, let's use the python script sensortagcol-lector.py from the command line to ensure that works.

Run the script with its default settings, so it will scan for BLE devices (such as the sen-sortag) and when it finds them – any number up to 16 – will automatically connect and start reading sensors and print results on screen. Make sensortag is off and run the command:

```
  sudo python sensortagcollector.py /*collects sensor data
```

Switch on the sensortag. Its light will start flashing. Shortly, the scanning script will see the tag and print a message to say it has been found. Then there is a short delay while the Pi Bluetooth stack connects to the tag, starts reading sensors and printing the readings (figure 12).



```
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","accelerometer":[0.09765625,0.0068359375,2.09912109375]}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","lightmeter":173.6}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","accelerometer":[0.09765625,0.0068359375,2.09912109375]}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","lightmeter":174.24}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","accelerometer":[0.08642578125,-0.0029296875,2.10205078125]}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","lightmeter":182.24}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","accelerometer":[0.08642578125,-0.0029296875,2.10205078125]}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","lightmeter":182.24}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","accelerometer":[0.0947265625,-0.00634765625,2.0927734375]}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","lightmeter":178.72}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","accelerometer":[0.0947265625,-0.00634765625,2.0927734375]}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","lightmeter":178.72}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","accelerometer":[0.099609375,0.0009765625,2.10009765625]}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","lightmeter":181.28}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","accelerometer":[0.099609375,0.0009765625,2.10009765625]}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","lightmeter":180.64}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","accelerometer":[0.0947265625,-0.001953125,2.0986328125]}
{"deviceuid":"54:6c:0e:53:01:01","devicename":"ST-1","lightmeter":180.64}
```

Figure 12. Sensor data on Raspberry

At this point when the Pi has connected to our tag, the green light on the sensortag stops flashing, which can be confusing because this appears the same as the tag not being turned on, and also once it is connected the way to turn off the sensortag has changed, so you have to give the power button a long hold and release to make it disconnect from the Pi, then the green LED flashes again, and to actually turn it off quickly press the power button again briefly.

It is important to do this before the pi connects again because that will also stop the LED flashing. The sensor readings seem to start at 0 initially then after a few readings start to show real readings. Press control-c to stop the script. This disconnects the Pi from all sensortag it has connected with.

It is possible to give this device a specific name in case there are similar operating device in the surrounding. Write this code in the command line as follows and give a friendly name.

```
sudo python sensortagcollector.py -d uid=friendlyname

/*name the sensortag with friendly name
```

3.4 Create a node red flow reading from sensortag data

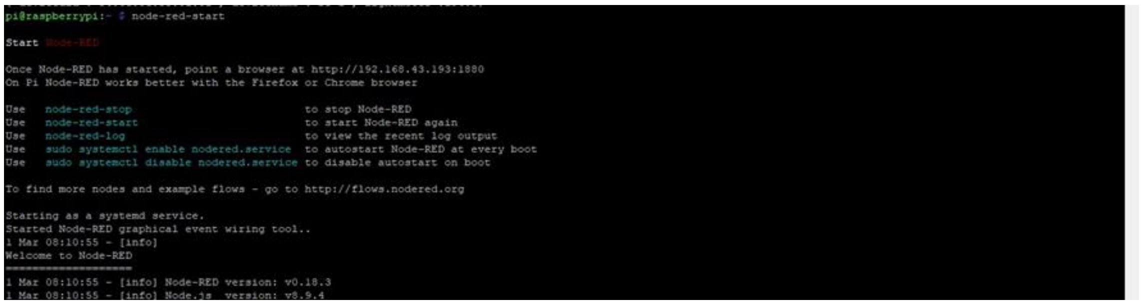
Node-RED is a powerful tool for building Internet of Things (IoT) applications with a focus on simplifying the ‘wiring together’ of code blocks to carry out tasks. It uses a visual programming approach that allows developers to connect predefined code blocks, known as ‘nodes’, together to perform a task. The connected nodes, usually a combination of input nodes, processing nodes and output nodes, when wired together, make up a ‘flows’ [9].

Originally developed as an open source project at IBM in late 2013, to meet their need to quickly connect hardware and devices to web services and other software – as a sort of glue for the IoT – it has quickly evolved to be a general purpose IoT programming tool. Importantly, Node-RED has rapidly developed a significant and growing user base and an active developer community who are contributing new nodes that allow programmers to reuse Node-RED code for a wide variety of tasks.

Although Node-RED was originally designed to work with the Internet of Things, i.e. devices that interact and control the real world it become useful tool to design our smart home device.

In this section we create a flow which shows the sensortag data in a debug node, then in the next sections we will see how to configure our own IoT Platform application and then connect the Node-Red flow to IoT Platform. Using putty or terminal on VNC open Node-Red in browser. At the command prompt (figure 13):

```
node-red-start /*connecting node red using putty
```



```
pi@raspberrypi:~ $ node-red-start

Start Node-RED

Once Node-RED has started, point a browser at http://192.168.43.193:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use   node-red-stop                          to stop Node-RED
Use   node-red-start                         to start Node-RED again
Use   node-red-log                           to view the recent log output
Use   sudo systemctl enable nodered.service  to autostart Node-RED at every boot
Use   sudo systemctl disable nodered.service to disable autostart on boot

To find more nodes and example flows - go to http://flows.nodered.org

Starting as a systemd service.
Started Node-RED graphical event wiring tool..
1 Mar 08:10:55 - [info]
Welcome to Node-RED
===================
1 Mar 08:10:55 - [info] Node-RED version: v0.18.3
1 Mar 08:10:55 - [info] Node.js  version: v8.9.4
```

Figure 13. Node-Red browser access on Pi

Node-RED displays a message including the address of the browser access point on the Pi. In our browser we connect to Node-Red (figure 14). For me, with the Pi on IP address 192.168.43.193.1880, Node-RED is on http:// 192.168.43.193.1880. If you are running your browser locally on your Pi then Node-RED is on http://127.0.0.1:1880.

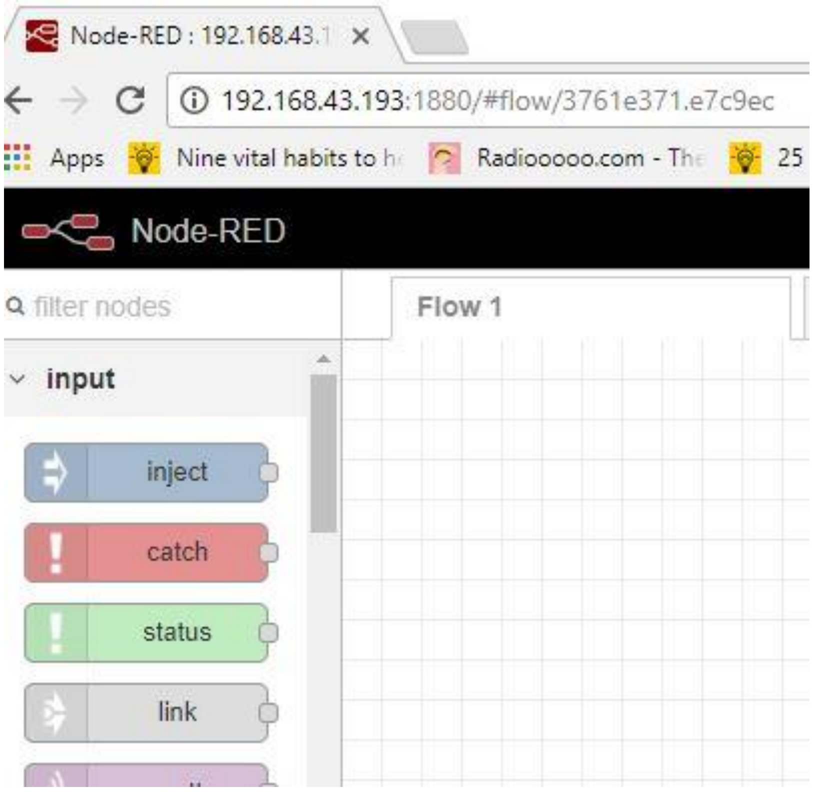


Figure 14. Node-Red IP address to perform node coding

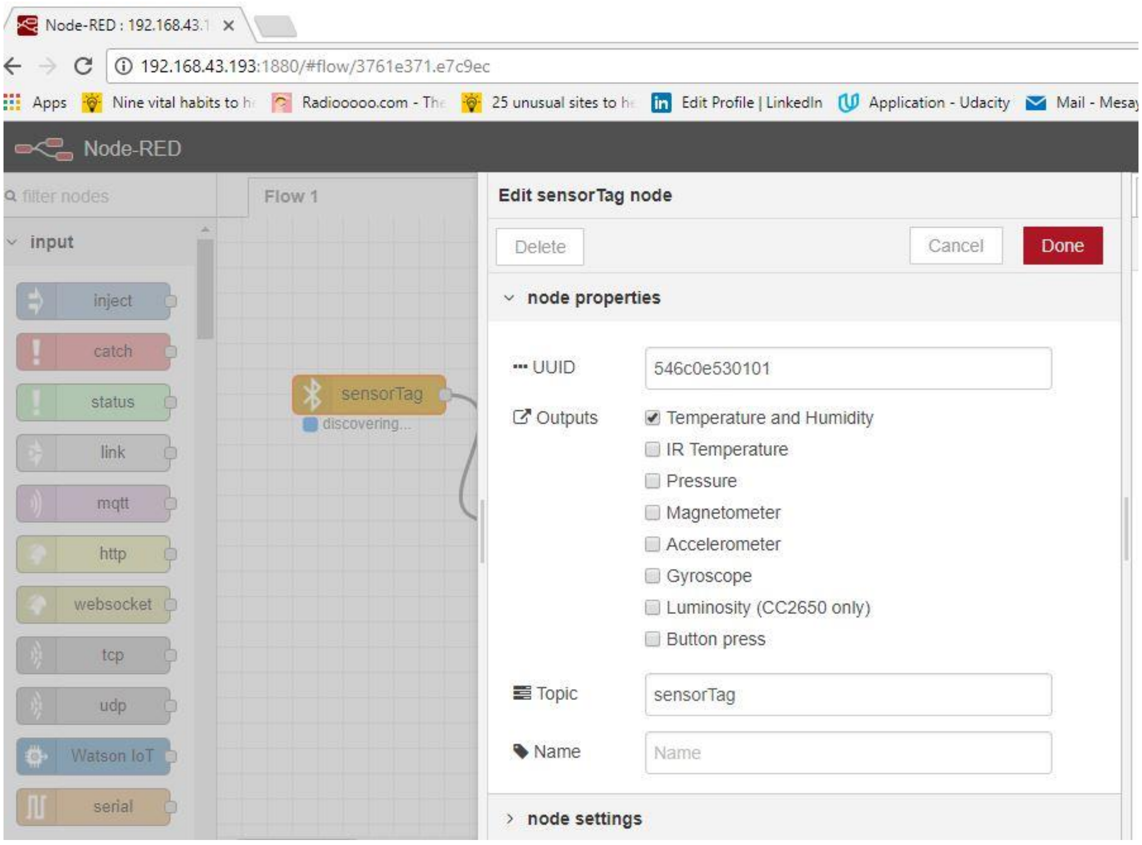
If there is no Sensortag on advanced group of tags in Node-Red, it have to be installed using NodeJS that has the code access for Sensortag. At the command prompt:

```
npm i node-red-node-sensortag/*installing sensortag
node on node red
```

This command will take some minutes to finish. Restart Raspberry pi when it finish. And then start again Node-RED server using the command:

```
node-red-start
```

To send data from sensortag to Watson IoT platform a flow code have to be built on Node-Red using nodes. Select sensortag from node list on the left side and double click the node to configure the sensor ID. Then select the data to be captured which are Temperature and Humidity for smart home device (figure 15).



Edit sensorTag node

node properties

UUID: 546c0e530101

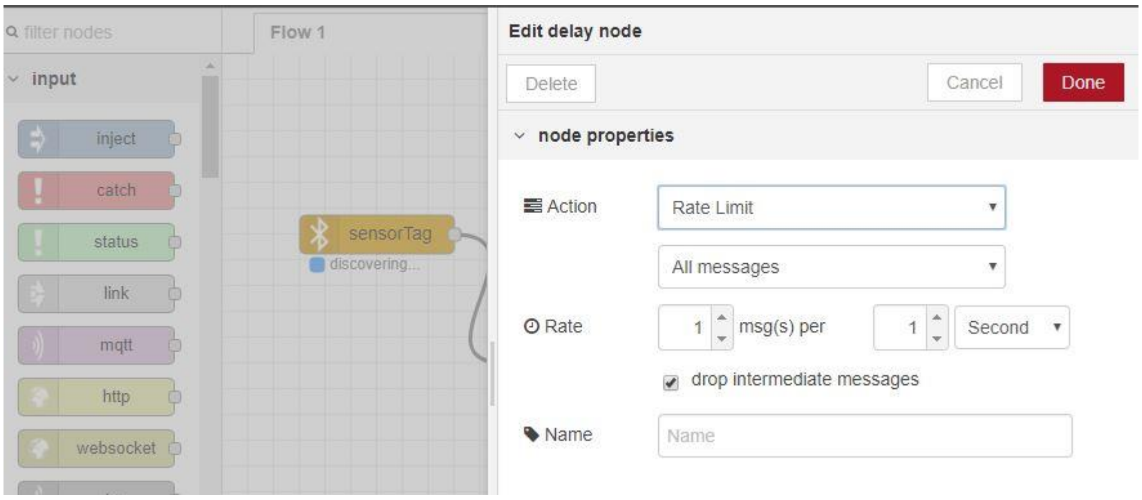
Outputs:
[✓] Temperature and Humidity
[ ] IR Temperature
[ ] Pressure
[ ] Magnetometer
[ ] Accelerometer
[ ] Gyroscope
[ ] Luminosity (CC2650 only)
[ ] Button press

Topic: sensorTag

Name: Name

Figure 15. Screen shot editing Sensortag node to select temperature and humidity

The next step is to connect a switch node to filter the messages from sensortag, which configure the node to just pass the objects messages that are in the topic as values of temperature and humidity.



Edit delay node

node properties

Action: Rate Limit

All messages

Rate: 1 msg(s) per 1 Second

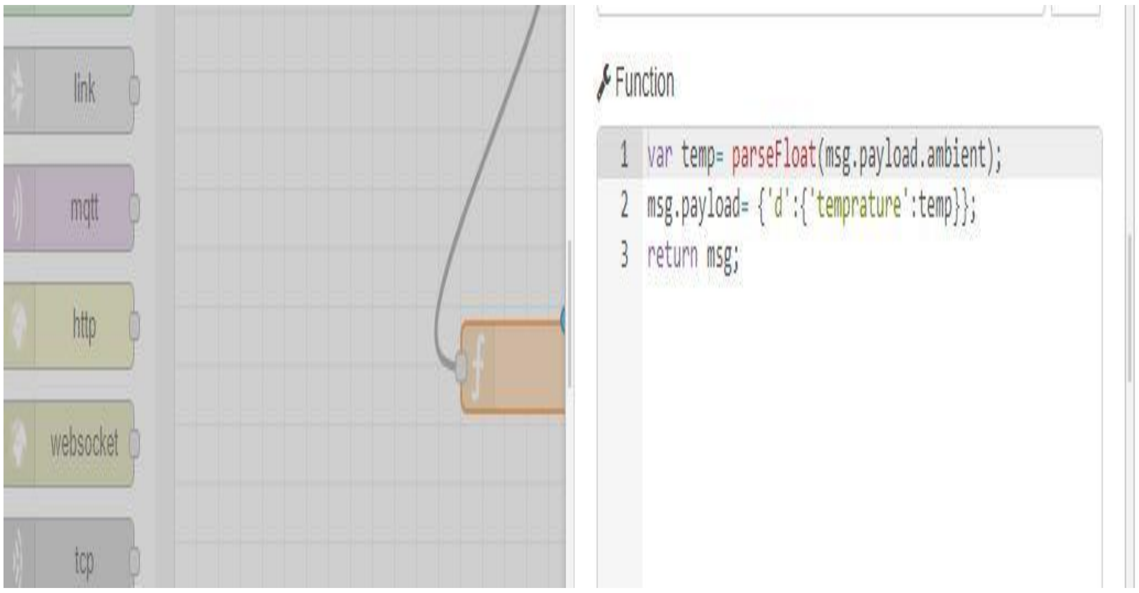
☑ drop intermediate messages

Name: Name

Figure 16. Edit delay node

Even if we filter the messages, sensortag will send a lot messages, to limit the number of messages, we will use the delay node and configure to limit the number of messages for 1 per minute (figure 16).

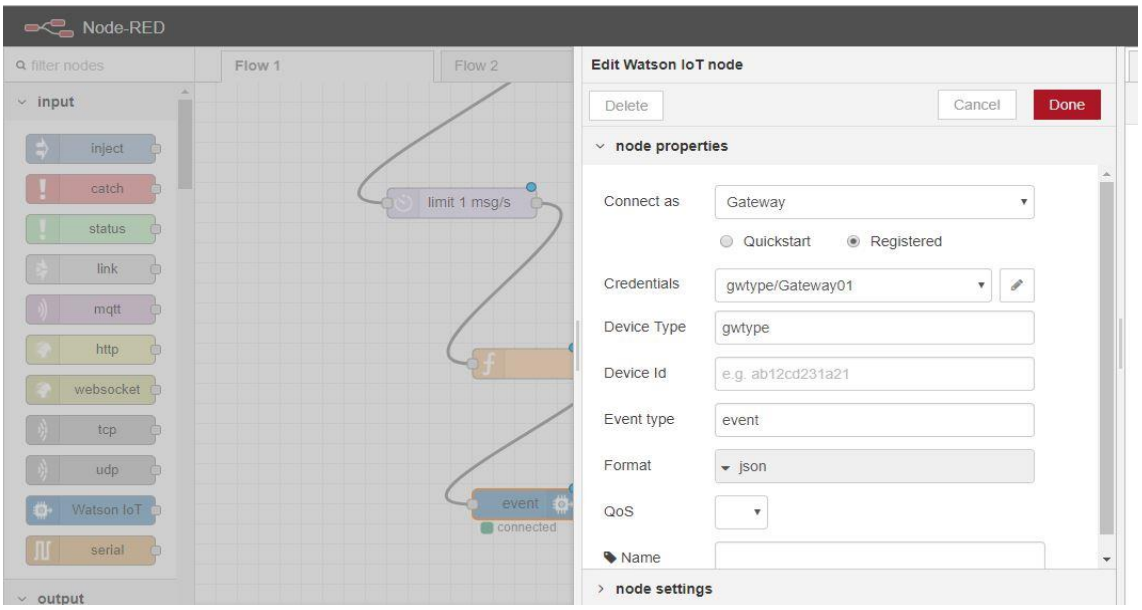
Before we send the message to Watson Iot, we need to format the data to mqtt server in Cloud server to accept. This node function has this java script code just to correct the format of the data (figure 17).



```
var temp= parseFloat(msg.payload.ambient);
msg.payload= {'d':{'temprature':temp}};
return msg;
```

Figure 17. Screen shot of Node function code

The last step would be to connect the node Watson IoT on Node-Red with function node output to send the message to IBM Cloud. A device have to be created onto Watson IoT platform and get the credentials to configure this node. Click on Deploy button and the raspberry pi will connect with sensortag to send data on Cloud (figure 18).



Connect as: Gateway
( ) Quickstart (●) Registered
Credentials: gwtype/Gateway01
Device Type: gwtype
Device Id: e.g. ab12cd231a21
Event type: event
Format: json
QoS:
Name:

Figure 18. Screen shot of Watson IoT node to send data to cloud

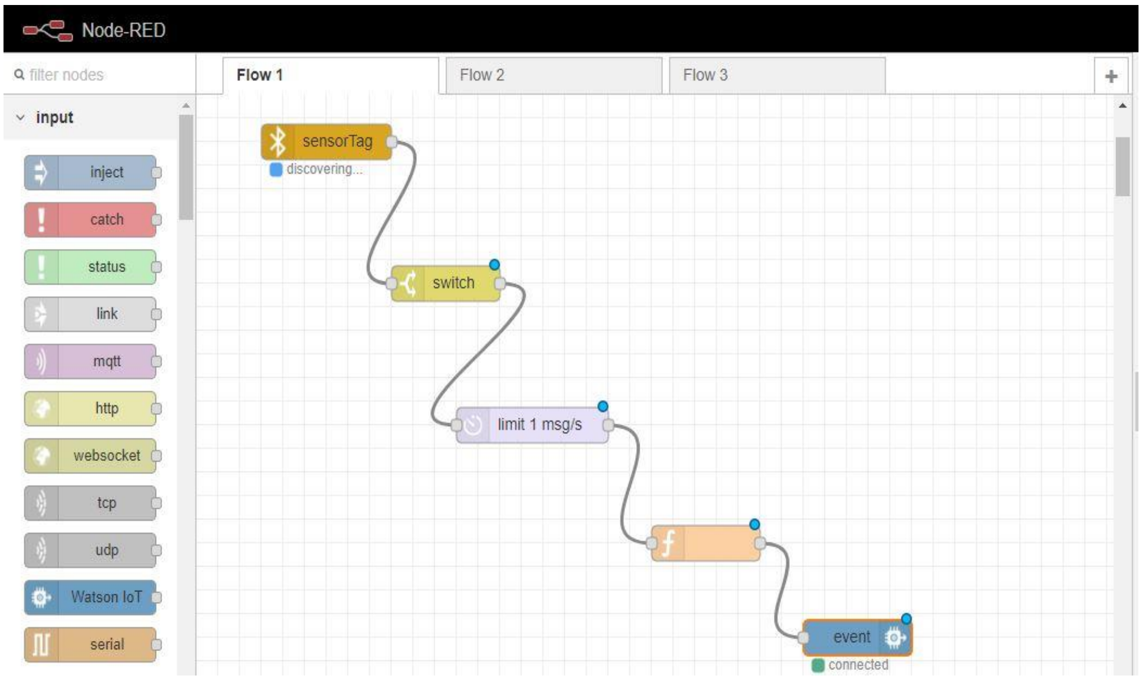


Figure 19. Node Red flow to connect to the IoT cloud

3.5 Preparing the Application in IoT Platform to receive Sensor Data

This section will focus on creating IoT Platform application and go through on how to create the needed gateway device type, gateway device, and the sensortag device type. There is no need to create a sensortag device for sensortag. This will be created automatically because our Pi will be acting as a gateway, and a gateway can automatically register the devices it is acting for this case that sensortag it finds and connects to.

The IBM Watson Internet of Things Platform is a fully managed, cloud-hosted service that makes it simple to derive value from Internet of Things (IoT) devices. When combined with the IBM Cloud platform, Watson IoT Platform provides simple, but powerful application access to IoT devices and data.

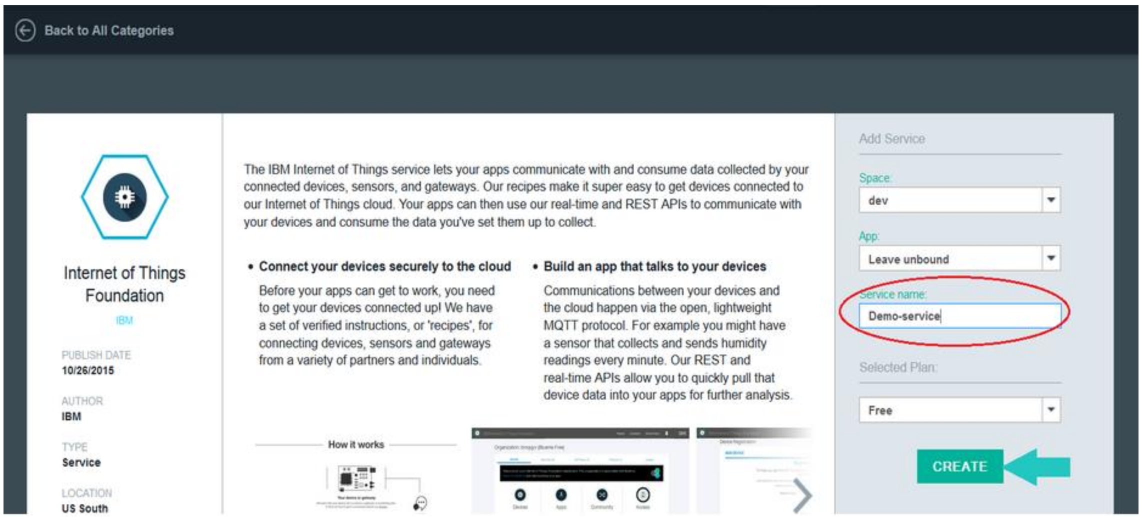
Gateways are a specialized class of devices in Watson IoT Platform which serve as access points to the Watson IoT Platform for other devices. Gateway devices have additional permission when compared to regular devices and can perform the following functions:

- Register new devices to Watson IoT Platform
- Send and receive its own sensor data like a directly connected device,
- Send and receive data on behalf of the devices connected to it
- Run a device management agent, so that it can be managed, also manage the devices connected to it.

3.5.1 Create Watson IoT Organization

When you create a service “Internet of Things Platform” in IBM Cloud, a new Organization will be created for you. This Watson IoT organization is a space used for connecting and managing devices (and Gateway devices) to the Watson IoT Platform, so that your applications can access their live and historical data.

First we have to sign in to our IBM account and type a name for our service and click create button as shown below (figure 20)



PUBLISH DATE
10/26/2015

AUTHOR
IBM

TYPE
Service

LOCATION
US South

The IBM Internet of Things service lets your apps communicate with and consume data collected by your connected devices, sensors, and gateways. Our recipes make it super easy to get devices connected to our Internet of Things cloud. Your apps can then use our real-time and REST APIs to communicate with your devices and consume the data you've set them up to collect.

- **Connect your devices securely to the cloud**

  Before your apps can get to work, you need to get your devices connected up! We have a set of verified instructions, or 'recipes', for connecting devices, sensors and gateways from a variety of partners and individuals.

- **Build an app that talks to your devices**

  Communications between your devices and the cloud happen via the open, lightweight MQTT protocol. For example you might have a sensor that collects and sends humidity readings every minute. Our REST and real-time APIs allow you to quickly pull that device data into your apps for further analysis.

Add Service

Space: dev

App: Leave unbound

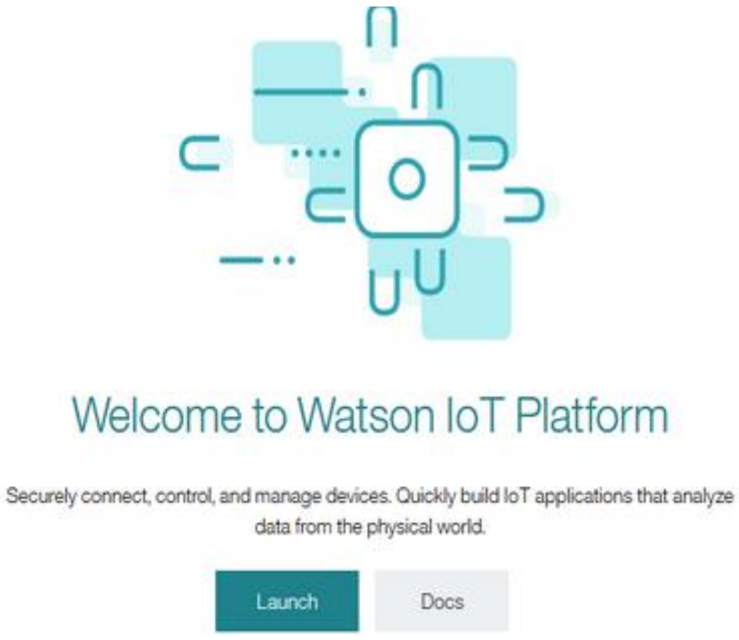
Service name: Demo-service

Selected Plan:

Free

Figure 20. IBM Watson IoT platform to create organization [7]

Observe a Welcome page, click on Launch button (figure 21) to enter into the IBM Watson IoT Platform organization space. The IoT organization is a space used for connecting and managing devices to the IBM Watson IoT Platform, so that your applications can access their live and historical data.



Welcome to Watson IoT Platform

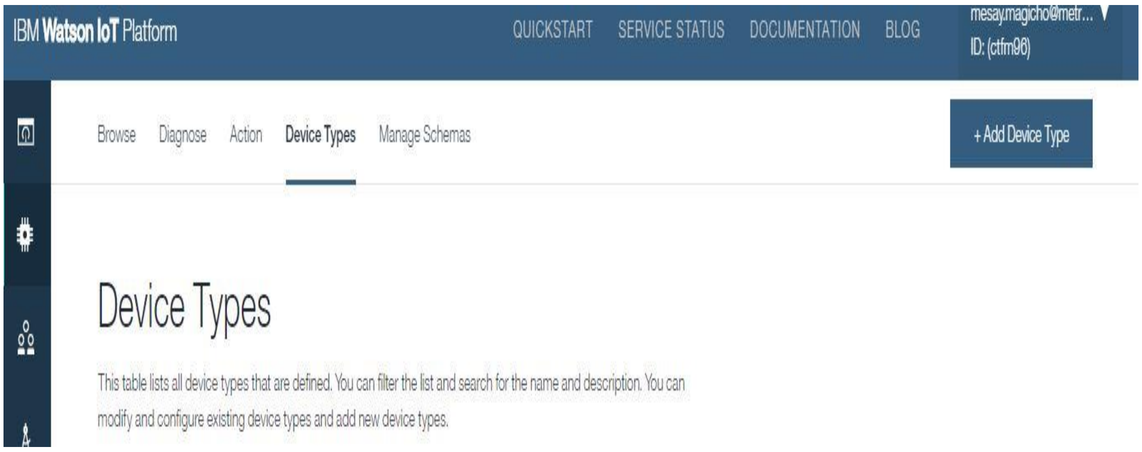
Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.

Figure 21. Launching your IoT Organization

3.5.2 Create Gateway Device Type

Each Gateway device connected to the Watson IoT Platform is associated with a device type. Device types are intended to be groups of devices which share common characteristics. So in order to add Gateway devices in Watson IoT Platform, one need to create a device type. Following are the instructions for creating a Gateway Device Type.

In the Watson IoT Platform dashboard, click Browse tab and then Add Device button as shown below (figure 22)

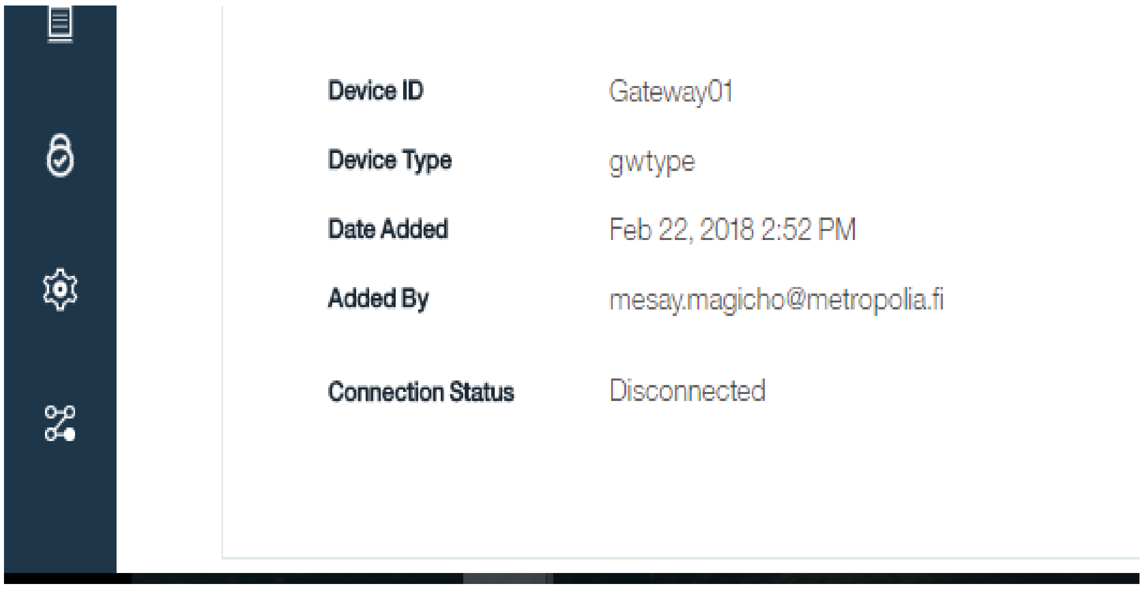


Device Types

This table lists all device types that are defined. You can filter the list and search for the name and description. You can modify and configure existing device types and add new device types.

Figure 22. Creating device type

We choose device type as gateway and give it a name. You need to have a unique device name and device id (figure 23), which will distinguish it your gateway from other devices that you might connect to Watson IoT platform. You can leave all optional values and attributes then following the procedure add the gateway device. To get your Gateway connected, you need to add the credentials to your Gateway. So make a note of them.



Device ID	Gateway01
Device Type	gwtype
Date Added	Feb 22, 2018 2:52 PM
Added By	mesay.magicho@metropolia.fi
Connection Status	Disconnected

Figure 23. Gateway credentials

3.6 Connecting the Pi to IoT Platform to send sensortag data

This is the final step, which connect Raspberry Pi as a gateway and send sensortag data to IoT Platform. In the Pi, in Node-RED you will add a Watson IoT node and configure it to connect to your IoT Platform application as a gateway, and you will specify the device type to be used. The device name will be automatically created in IoT Platform using the device name from the sensortagcollector script. FYI the device name is inserted into msg.deviceId in the JSON String to object node.

In the Pi Node-RED in your browser, in the Output section of the palette, add a Watson IoT output node (figure 24)

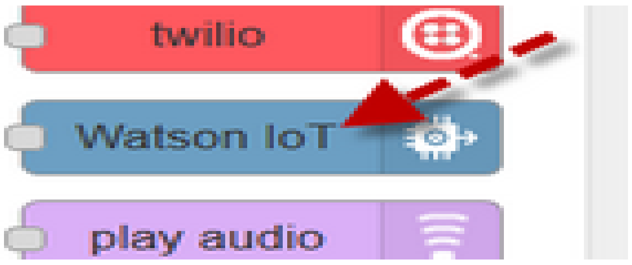
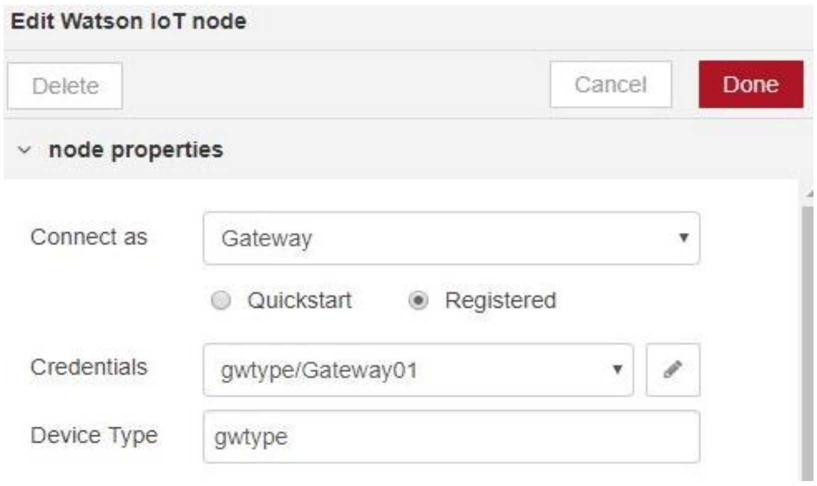


Figure 24. Watson IoT output node

Double-click the IBM IoT Device node and change the setting to Connect as Gateway (figure 25), Registered, and Device Type gwtype:



Connect as: Gateway

Quickstart [✓ Registered]

Credentials: gwtype/Gateway01

Device Type: gwtype

Figure 25. Register device type

4 Results

The Smart Home Device elements are now ordered in the right flow procedures and configured the previous chapters. The sensortag reads temperature and humidity levels and send the data to the Raspberry Pi via LE Bluetooth connection (figure 12). Those python scripts scanned and collected data by the Raspberry using Node-Red which can run a script and collect its output in to a node red flow (figure 19).

The flow will convert the python scripts from a unique IP address IoT platform on IBM cloud using the Raspberry as a gateway (figure 26).

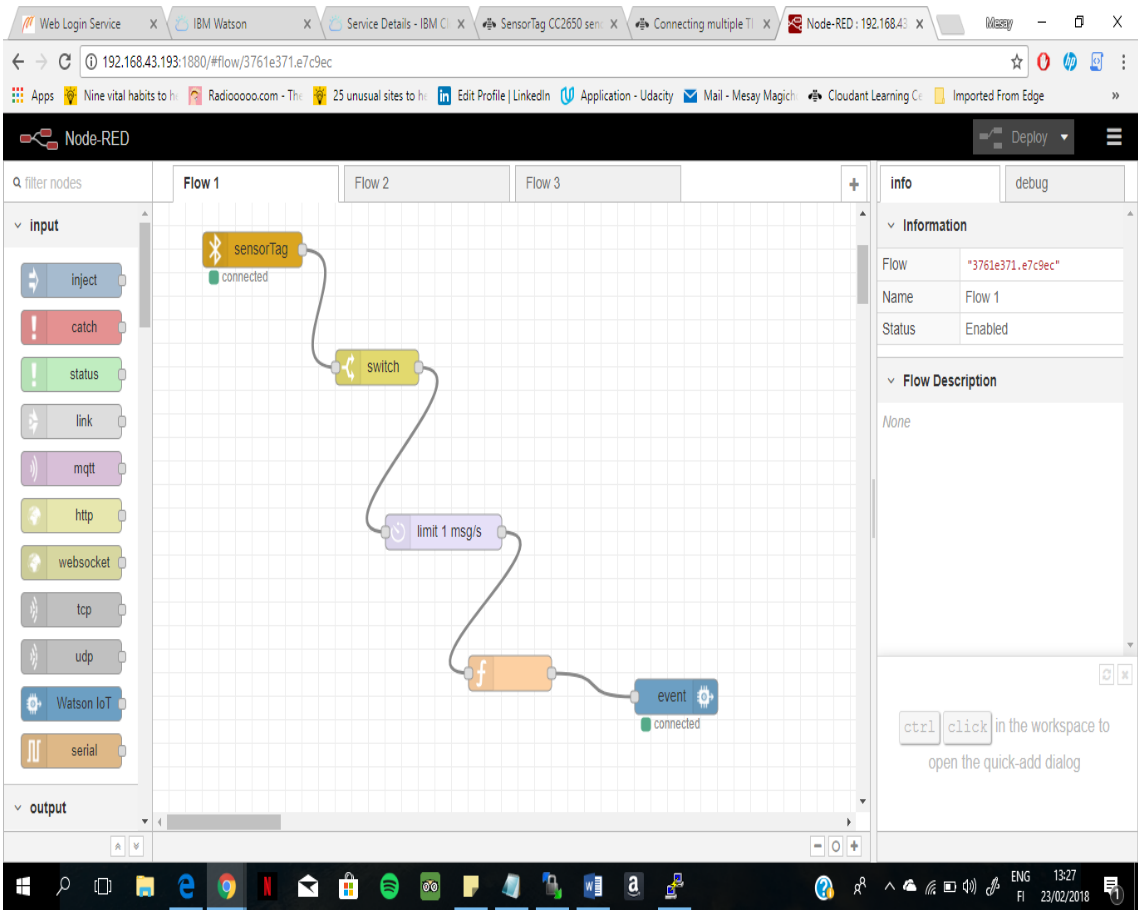
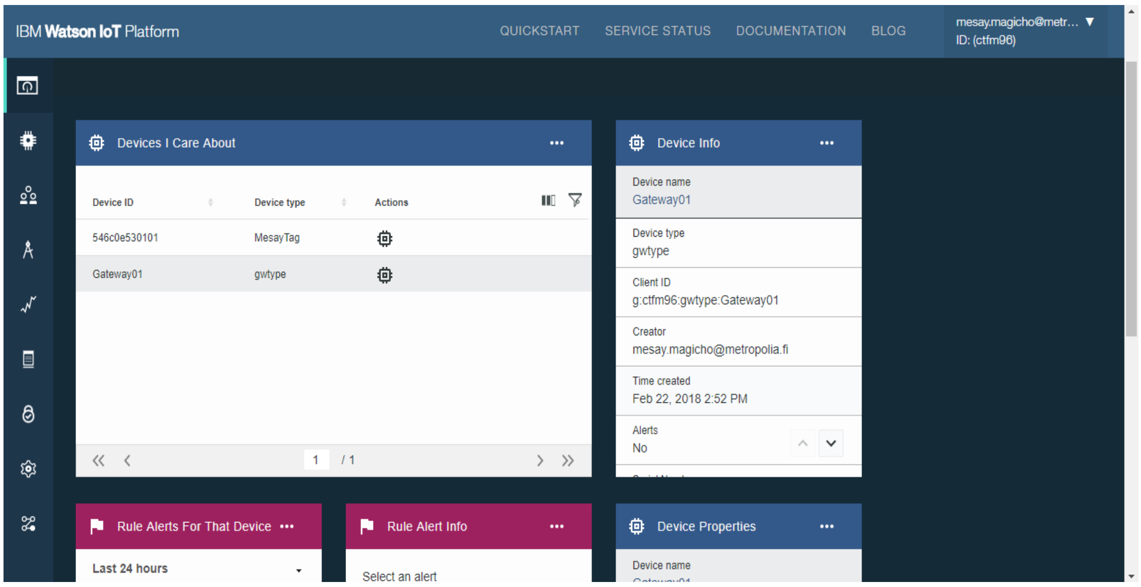


Figure 26. Connection flow of a Smart Home Device [screen shot]

We can see the devices are connected successfully!



IBM Watson IoT Platform   QUICKSTART   SERVICE STATUS   DOCUMENTATION   BLOG   mesay.magicho@metr... ID: (ctfm96)

Devices I Care About

Device ID	Device type	Actions
546c0e530101	MesayTag	
Gateway01	gwtype	

1 / 1

Device Info

Device name
Gateway01

Device type
gwtype

Client ID
g:ctfm96:gwtype:Gateway01

Creator
mesay.magicho@metropolia.fi

Time created
Feb 22, 2018 2:52 PM

Alerts
No

Rule Alerts For That Device

Last 24 hours

Rule Alert Info

Select an alert

Device Properties

Device name

Figure 27. Screen shot of devices on IBM Cloud

During last result execution on IBM cloud the cloud service client sent a later to notify about critical issues that affect output results (figure 29). Sensor data sent to cloud and can be seen on Node-Red that devices are connected with the cloud but doesn’t post the readings on IoT platform dashboard. This was happening sometimes for lite free service users. Here is previous records of Temperature and humidity(figure 28).

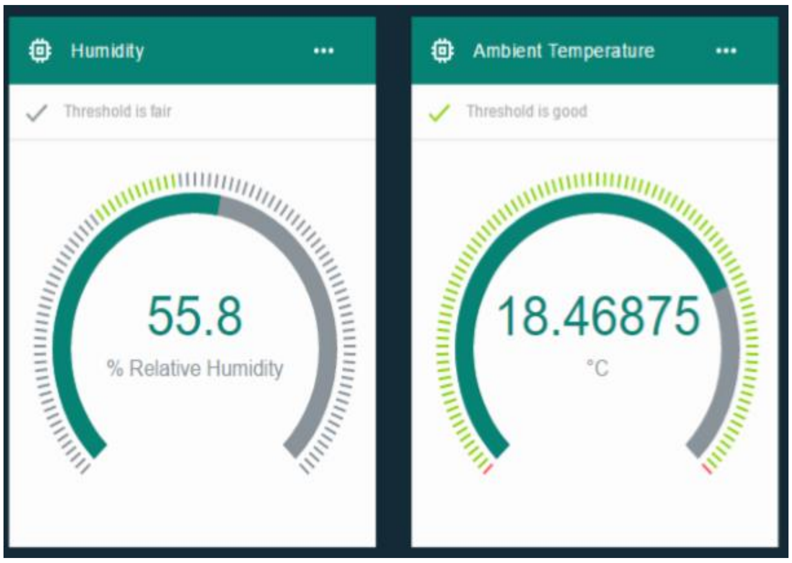
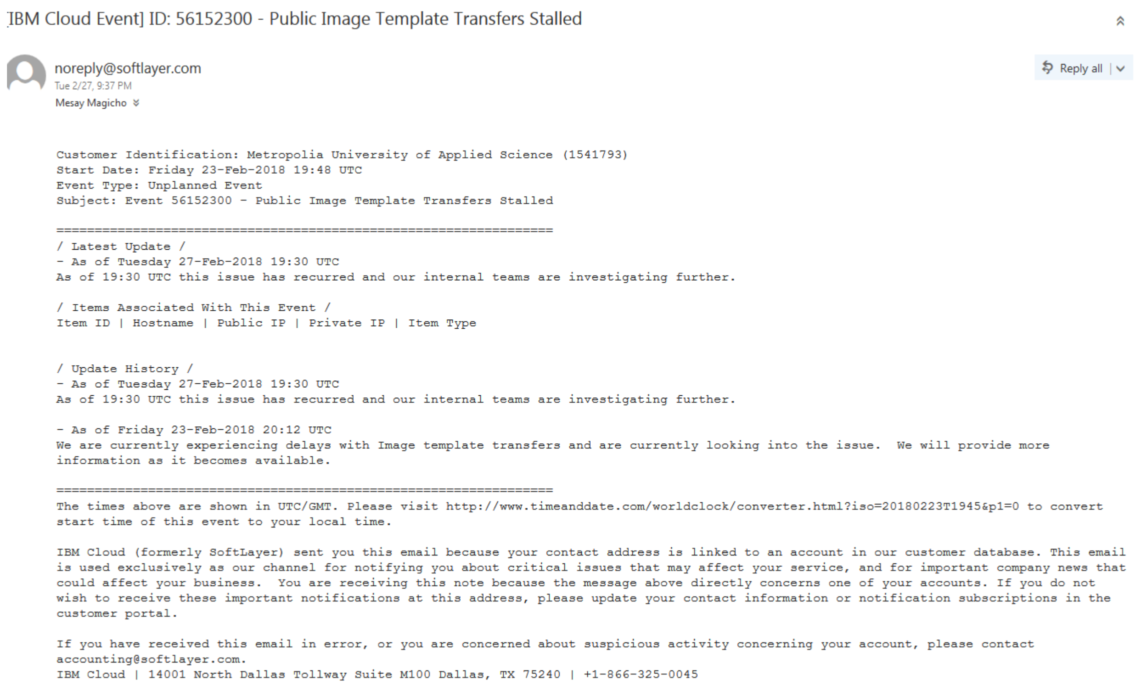


Figure 28. Temperature and Humidity



IBM Cloud Event] ID: 56152300 - Public Image Template Transfers Stalled

noreply@softlayer.com
Tue 2/27, 9:37 PM
Mesay Magicho

```
Customer Identification: Metropolia University of Applied Science (1541793)
Start Date: Friday 23-Feb-2018 19:48 UTC
Event Type: Unplanned Event
Subject: Event 56152300 - Public Image Template Transfers Stalled

=================================================================
/ Latest Update /
- As of Tuesday 27-Feb-2018 19:30 UTC
As of 19:30 UTC this issue has recurred and our internal teams are investigating further.

/ Items Associated With This Event /
Item ID | Hostname | Public IP | Private IP | Item Type


/ Update History /
- As of Tuesday 27-Feb-2018 19:30 UTC
As of 19:30 UTC this issue has recurred and our internal teams are investigating further.

- As of Friday 23-Feb-2018 20:12 UTC
We are currently experiencing delays with Image template transfers and are currently looking into the issue.  We will provide more
information as it becomes available.

=================================================================
The times above are shown in UTC/GMT. Please visit http://www.timeanddate.com/worldclock/converter.html?iso=20180223T1945&p1=0 to convert
start time of this event to your local time.

IBM Cloud (formerly SoftLayer) sent you this email because your contact address is linked to an account in our customer database. This email
is used exclusively as our channel for notifying you about critical issues that may affect your service, and for important company news that
could affect your business.  You are receiving this note because the message above directly concerns one of your accounts. If you do not
wish to receive these important notifications at this address, please update your contact information or notification subscriptions in the
customer portal.

If you have received this email in error, or you are concerned about suspicious activity concerning your account, please contact
accounting@softlayer.com.
IBM Cloud | 14001 North Dallas Tollway Suite M100 Dallas, TX 75240 | +1-866-325-0045
```

Figure 29. IBM Cloud Issues messages

5 Conclusion

The aim of this thesis project was to read the level of temperature and humidity in a room using Sensortag CC2650 and send the data to Raspberry Pi 3 as a gateway to IBM Cloud. A real time measurement is sent to the cloud over a network. I demonstrate how to set up and create a simple Smart Home Device.

This project focus on the learning process of IoT device design using sensortag, Raspberry Pi and IBM cloud. The project was a success. This project follows similar previous recipes from IBM open source projects and doesn’t cover other aspects of the smart home device such as controlling thermostat and humidity equipment’s in the room. The user have to do that manually.

This thesis project can further develop and can also be used for further development of similar open source projects. However, because of the fast development nature of IoT products the chance of using similar devices are limited.

References

1 David Mercer, Connected World: Smart Home Is Key To Tomorrow's Internet Of Things. [ONLINE] Available at: https://www.strategyanalytics.com/access-services/devices/connected-home/smart-home/ [Accessed 29 January 2018].

2 Texas Instruments Users Guide. 2016. [ONLINE] Available at: http://www.ti.com/lit/ug/tidu862/tidu862.pdf [Accessed 12 February 2018].

3 Multistandard Wireless MCU datasheet. 2015. [ONLINE] Available at: www.ti.com/lit/ds/symlink/cc2650.pdf

4 CC2650 SensorTag User's Guide - TI Wiki. 2018. [ONLINE] Available at: *processors.wiki.ti.com/index.php/CC2650_SensorTag_User%27s_Guide*

5 Raspberry Pi 3 wiki page. [ONLINE] Available at: : https://en.wikipedia.org/wiki/Raspberry_Pi

6 VNC Connect and Raspberry Pi Real VNC. 2015. [ONLINE] Available at: https://www.realvnc.com/docs/raspberry-pi.html

7 IBM Cloud Recipes open source. [ONLINE] Available at: https://www.redbooks.ibm.com/redbooks/pdfs/sg248374.pdf [Accessed 16 February 2018]

8 Hands on Workshop Watson IOT.pdf - File - IBM. [ONLINE] Available at: *https://www.ibm.com/.../d3b16283-e026-4229-a0ee-988f0cf1fda8* [Accessed 17 February 2018]

9 TI SensorTag and Raspberry Pi - developersWorks Recipes. 2015. [ONLINE] Available at: https://developer.ibm.com/recipes/tutorials/ti-sensor-tag-and-raspberry-pi/

10 Node-Red Users guide. [ONLINE] Available at: http://noderedguide.com/nr-lecture1/ [Accessed on 27 February 2018]

Source Code To Deploy Node-Red flow on Raspberry

/*
*Original author: IBM
Edited: Mesay
*/

```
    [{“id”:”fc916843.2edb28″,”type”:”sen-
sorTag”,”z”:”ee3e8f00.10579″,”name”:”tag”,”topic”:”sen-
sorTag”,”uuid”:”a0e6f8c21684″,”temperature”:true,”humid-
ity”:true,”pressure”:true,”magnetometer”:true,”accelerome-
ter”:true,”gyroscope”:true,”keys”:true,”luxo-
meter”:false,”x”:115.5,”y”:58,”wires”:[[“546008af.f1b3a8”]]},{“i
d”:”fdb01a2b.833a68″,”type”:”de-
bug”,”z”:”ee3e8f00.10579″,”name”:””,”active”:true,”con-
sole”:”false”,”com-
plete”:”true”,”x”:660,”y”:77,”wires”:[]},{“id”:”546008af.f1b3a8″
,”type”:”switch”,”z”:”ee3e8f00.10579″,”name”:””,”prop-
erty”:”topic”,”property-
Type”:”msg”,”rules”:[{“t”:”cont”,”v”:”tempera-
ture”,”vt”:”str”}],”checkall”:”true”,”out-
puts”:1,”x”:221,”y”:146,”wires”:[[“aa8718c3.6315f8”]]},{“id”:”aa
8718c3.6315f8″,”type”:”de-
lay”,”z”:”ee3e8f00.10579″,”name”:””,”pauseType”:”rate”,”timeout”
:”5″,”timeoutUnits”:”seconds”,”rate”:”1″,”nbRateUnits”:”1″,”ra-
teUnits”:”minute”,”randomFirst”:”1″,”randomLast”:”5″,”ran-
domUnits”:”sec-
onds”,”drop”:true,”x”:354,”y”:61,”wires”:[[“d68c43b5.dce2f”]]},{
“id”:”d68c43b5.dce2f”,”type”:”func-
tion”,”z”:”ee3e8f00.10579″,”name”:””,”func”:”var  temp  =  parse-
```

Float(msg.payload.ambient);nmsg.payload = {'d':{'tempera-ture':temp}};nreturn msg;”,”out-puts”:1,”noerr”:0,”x”:493,”y”:149,”wires”:[[“fdb01a2b.833a68",”daddaa20.7ffdd8”]]},{“id”:”daddaa20.7ffdd8",”type”:”wiotp out”,”z”:”ee3e8f00.10579",”authType”:”d”,”qs”:”false”,”qsDe-viceId”:””,”deviceKey”:”4d195373.2056ec”,”deviceType”:””,”de-viceId”:””,”event”:”event”,”for-mat”:”json”,”name”:””,”x”:665,”y”:197,”wires”:[]},{“id”:”4d195373.2056ec”,”type”:”wiotp-credentials”,”z”:””,”name”:””,”org”:”Me-sayTag”,”devType”:”gateway”,”devId”:”546C0E530101”}]