

# INTRODUCCIÓN

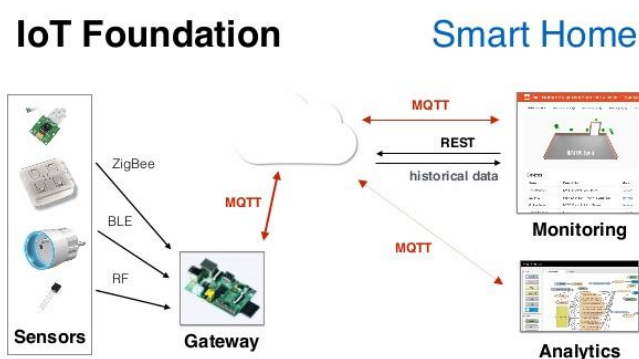
Nuestro objetivo en este proyecto es utilizar los sensores de un dispositivo, extraer los datos para analizarlos y posteriormente poder hacer predicciones futuras. Este proyecto está orientado a futuras aplicaciones relacionadas con el internet de las cosas (IoT). En nuestro caso el dispositivo será una Raspberry Pi 3 al que le conectaremos el módulo SenseHAT que dispone de diversos sensores.

Antes de nada, me gustaría decir que soy un iniciado en este mundo y mi objetivo haciendo este tutorial es poder asentar los conocimientos que he obtenido recientemente, así como ayudar a posibles personas que quieran introducirse en este mundo. Evitaré explicar conceptos que aún no conozco completamente. Las palabras que rotule en negro se darán por hecho que el lector conocerá su significado y en caso contrario acudirá a “San Google” a obtener información acerca de ellas.

Aunque está explicado paso a paso como realizar el proyecto, es importante tener nociones básicas de Node-RED, Python, JavaScript, JSON y SQL para un correcto entendimiento del tutorial.

Como he dicho anteriormente este proyecto es un escenario “end to end” que puede ser aplicado a números casos del IoT. Un dispositivo que está instalado en un vehículo o una casa puede estar obteniendo numerosos datos acerca del entorno que le rodea como puede ser la presión, la orientación, la posición etc. Un posible ejemplo más concreto puede ser un reloj que analiza las pulsaciones del individuo que lo lleva puesto y este puede avisar a otras personas acerca de si se encuentra durmiendo, despierto, haciendo deporte, nervioso, relajado... ¡Y todo en tiempo real! Para que esto pueda ocurrir es necesario que los dispositivos estén configurados correctamente tanto en su programa interno como en nuestra plataforma de administración de dispositivos. Esto hace que haya una correcta organización pues, aunque en este proyecto solo trabajemos con un dispositivo, en la vida real podemos hablar de miles de ellos mandando datos a nuestra plataforma. Así, el primer paso sería saber obtener los datos de los sensores de nuestro dispositivo y enviarlos al **bróker** que se encuentra en la nube. Hoy en día existen plataformas orientadas a este tipo de escenarios y a cambio de un módico precio nos permiten trabajar fácilmente. En nuestro caso trabajaremos con la nube de IBM llamada “**Bluemix**”, en especial con la herramienta orientada a este tipo de aplicaciones llamada “**Watson IoT Platform**”.

Cuando un dispositivo conecta con la nube para enviar eventos utiliza el protocolo **MQTT**. Una vez que tenemos los datos alojados en la nube el siguiente paso es extraerlos para su futuro análisis. En nuestro caso no haremos un análisis en tiempo real salvo un pequeño programa que controlará los leds de la Raspberry en función de la temperatura a la que se encuentre. Utilizaremos Apache Spark para extraer los datos y trabajaremos con Python para mostrar las gráficas y hacer peticiones SQL a la base de datos. ¡Así que sin más dilación comenzamos!



## PASO 1: REGISTRAR NUESTROS DISPOSITIVOS

Antes de comenzar es importante saber la diferencia entre “Gateway”, “Device” y “API”. Como he dicho anteriormente se deja al lector el trabajo de informarse acerca de estas definiciones.

En nuestro caso disponemos de una Raspberry Pi 3 y su módulo “SenseHAT”. Este módulo lleva integrado diversos sensores como puede ser el sensor de presión, temperatura, acelerómetro etc. En nuestro caso solo utilizaremos el sensor de temperatura y de humedad que serán suficiente para poder aplicarlos a cualquier otro ejemplo. Nuestra Raspberry Pi 3 debería tener instalado el sistema operativo “**Raspbian**”. Se da por hecho que el lector sabe instalarlo correctamente en su Raspberry.

A la hora de configurar nuestra Raspberry definiremos el módulo SenseHAT como “Device” y la Raspberry Pi 3 como “Gateway”.

Para configurarlos accedemos a nuestra cuenta de Bluemix en <https://console.bluemix.net> . La primera vez que la creamos tenemos una versión de prueba de 30 días. A continuación creamos un **espacio** (importante que se encuentre en US South), en este añadimos la plantilla “**Internet of Things Platform Starter**” configurada como “Lite” (versión gratuita) y rellenamos los campos necesarios para poder continuar. Una vez creada nuestra aplicación podremos observar que tenemos dos conexiones con ella. Una conexión con “**Watson IoT Platform**” y otra conexión llamada “**Cloudant NoSQL DB**”. **Watson IoT platform** nos servirá para administrar y configurar nuestros dispositivos y **Cloudant** nos servirá para almacenar los datos que nos lleguen.



Accedemos a la plataforma de Watson IoT que se encuentra señalada con la flecha azul en la imagen anterior. Nos debería parecer una ventana como la siguiente:



Una vez lanzada la plataforma lo primero que tenemos que hacer es registrar los **tipos de dispositivos** y a continuación registrar nuestra Raspberry Pi 3 y el SenseHAT con su tipo de dispositivo registrado anteriormente. Cuando registramos los dispositivos nos debería salir una ventana de este estilo:

## Credenciales de dispositivo

Ha registrado el dispositivo en la organización. Añada estas credenciales al dispositivo para conectarlo a la plataforma. Una vez conectado el dispositivo, puede navegar para ver los detalles de la conexión y los sucesos.

ID de organización	[REDACTED]
Tipo de dispositivo	SenseHat
ID de dispositivo	SH001
Método de autenticación	use-token-auth
Señal de autenticación	[REDACTED]

 Las señales de autenticación no son recuperables. Si pierde esta señal, deberá volver a registrar el dispositivo para generar una señal de autenticación nueva.

Es **importante guardar los datos de esta ventana** ya que la señal de autenticación no se nos volverá a mostrar nunca más y si la perdemos deberemos volver a registrar otra vez el dispositivo. Estos datos nos servirán para poder enviar los eventos a nuestra plataforma.

El resultado final deberá quedar parecido a este:

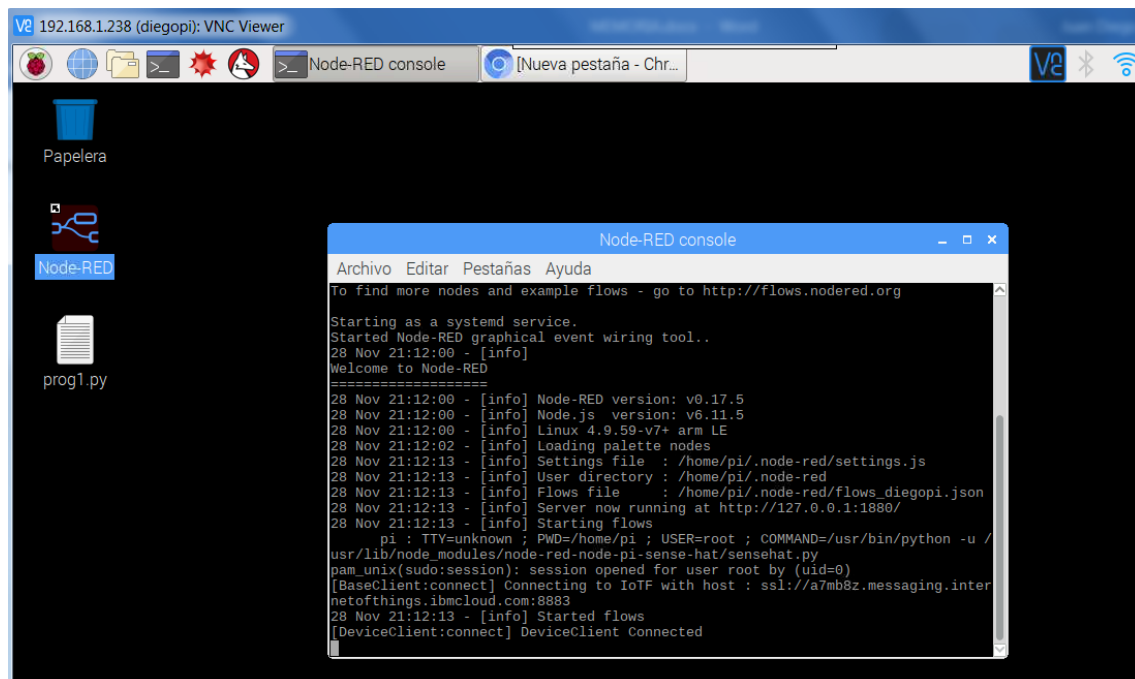


En este caso no importa si el tipo de dispositivo es Pasarela ("Gateway") o Dispositivo ("Device"). Ambos pueden enviar información al servidor, pero para que esto se produzca **deben estar registrados en nuestra plataforma.**

## PASO 2: ENVIAR DATOS A LA PLATAFORMA

Nuestro primer objetivo es obtener la temperatura y la humedad de la Raspberry. Para controlar la Raspberry puedes utilizar un teclado y un ratón conectados a esta o instalar algún programa "open source" de control remoto, como por ejemplo "VNC Viewer", que es el que utilizo yo, el cual te permite controlar la Raspberry desde tu propio ordenador lo que hace que sea mucho más cómodo.

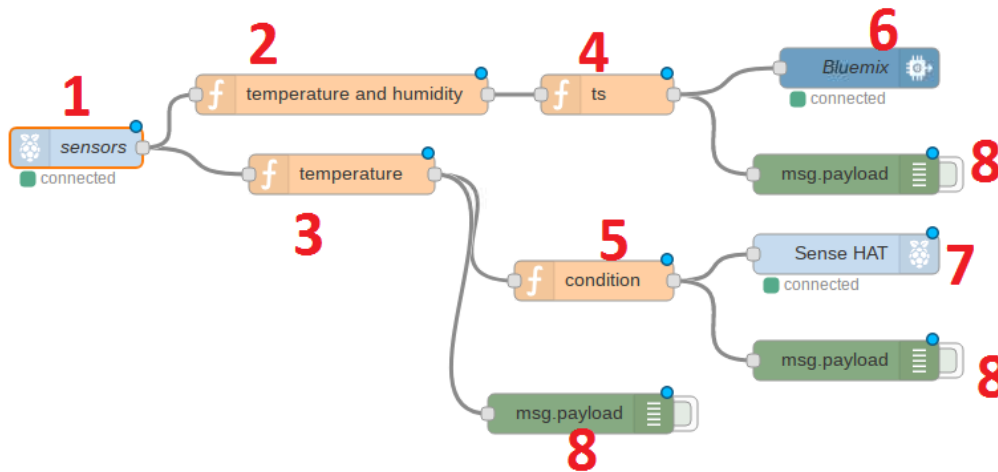
Una vez que tenemos el control de nuestra Raspberry iniciamos Node-RED que viene preinstalado con Raspbian.



Abrimos un explorador, accedemos a la dirección <http://127.0.0.1:1880/> e importamos el siguiente flujo (también lo puedes encontrar en el repositorio como flujo1.txt):

```
[{"id":"a5fded00.c4b12","type":"debug","z":"9c350e61.78843","name":"","active":true,"console":"false","complete":"false","x":650,"y":180,"wires":[]},{id:"6cf9fa04.8d6fe4","type":"rpi-sensehat\nin","z":"9c350e61.78843","name":"sensors","motion":false,"env":true,"stick":false,"x":50,"y":180,"wires":[{"id":"859a55c1.5c4678","35da13a9.f40a8c"}]},{id:"7ff8b0b6.8619c","type":"rpi-sensehat\nout","z":"9c350e61.78843","name":"","x":670,"y":280,"wires":[]},{id:"21178f23.9649a","type":"function","z":"9c350e61.78843","name":"condition","func":"var temperature = msg.payload.temperature;\n\nif (temperature >= 37) msg.payload = \"1-7,*maroon,0,*black\";\n\nelse if (temperature < 37 & temperature >= 36) msg.payload = \"1-7,*yellow,0-1,*black\";\n\nelse if (temperature < 36 & temperature >= 35) msg.payload = \"1-7,*olive,0-2,*black\";\n\nelse if (temperature < 35 & temperature >= 34) msg.payload = \"1-7,*lime,0-3,*black\";\n\nelse if (temperature < 34 & temperature >= 33) msg.payload = \"1-7,*green,0-4,*black\";\n\nelse if (temperature < 33 & temperature >= 32) msg.payload = \"1-7,*aqua,0-5,*black\";\n\nelse if (temperature < 31) msg.payload = \"1-7,*teal,0-6,*black\";\n\nelse msg.payload = \"ERROR\";\n\nreturn msg;","outputs":"1","noerr":0,"x":500,"y":280,"wires":[{"id":"7ff8b0b6.8619c","3edaa64b.d2076a"}]},{id:"3edaa64b.d2076a","type":"debug","z":"9c350e61.78843","name":"","active":false,"console":"false","complete":"false","x":670,"y":340,"wires":[]},{id:"622c17c1.aaa4f8","type":"wiotp\nout","z":"9c350e61.78843","authType":"d","qs":"false","qsDeviceId":"","deviceKey":"f59c71db.1540c","deviceType":"Raspberry-Pi-3","deviceId":"RBP001","event":"event","format":"json","qos":"","name":"Bluemix","x":660,"y":120,"wires":[]},{id:"859a55c1.5c4678","type":"function","z":"9c350e61.78843","name":"temperature","func":"var temperature = msg.payload.temperature;\n\nmsg.payload = {'temperature': temperature};\n\nreturn msg;","outputs":"1","noerr":0,"x":250,"y":220,"wires":[{"id":"21178f23.9649a"}]},{id:"35da13a9.f40a8c","type":"function","z":"9c350e61.78843","name":"temperature and humidity","func":"var humidity = msg.payload.humidity;\n\nvar temperature = msg.payload.temperature;\n\nmsg.payload = {'d': {'temperature': temperature, 'humidity': humidity}};\n\nreturn msg;","outputs":"1","noerr":0,"x":270,"y":140,"wires":[{"id":"323bdeea.4b1ff2"}]},{id:"323bdeea.4b1ff2","type":"function","z":"9c350e61.78843","name":"ts","func":"msg.payload.d.ts = new Date().getTime();\n\nreturn msg;","outputs":1,"noerr":0,"x":490,"y":140,"wires":[{"id":"a5fded00.c4b12","622c17c1.aaa4f8"}]},{id:"f59c71db.1540c","type":"wiotp-credentials","z":"","name":"raspberrypi","org":"a7mb8z","serverName":"","devType":"raspberrypi","devId":"RBP001","keepalive":"60","cleansession":true,"tls":"","usetls":false}]
```

Lo que nos da el siguiente resultado:



Explicaré brevemente lo que hace cada nodo. Las funciones se escriben en JavaScript y los mensajes están en formato JSON.

#### Nodo 1:

Este nodo nos envía los datos de los sensores de nuestro SenseHat. Si queremos más información acerca de como nos llegan estos datos podemos acceder a la información que se nos facilita en la ventana que nos aparece a la derecha de Node-RED.

info debug

**Node**

Name	sensors
Type	rpi-sensehat in
ID	"6cf9fa04.8d6fe4"

show more

**Information**

Raspberry Pi Sense HAT input node.

This node sends readings from the various sensors on the Sense HAT, grouped into three sets; motion events, environment events and joystick events.

**Motion events**

Motion events include readings from the accelerometer, gyroscope and magnetometer, as well as the current compass heading. They are sent at a rate of approximately 10 per second. The `topic` is set to `motion` and the `payload` is an object with the following values:

- `acceleration.x/y/z` : the acceleration intensity in Gs
- `gyroscope.x/y/z` : the rotational intensity in radians/s
- `orientation.roll/pitch/yaw` : the angle of the axis in degrees

### **Nodo 2:**

Utilizando JavaScript, este nodo extrae la temperatura y la humedad del mensaje obtenido del SenseHat. Haciendo doble click en él podemos observar como lo hacemos.

```
var humidity = msg.payload.humidity;

var temperature = msg.payload.temperature;

msg.payload = {'d' : {'temperature' : temperature,
                    'humidity' : humidity}};

return msg;
```

### **Nodo 3:**

Este nodo a diferencia del nodo 2, solo extrae la temperatura.

```
var temperature = msg.payload.temperature;

msg.payload = {'d' : {'temperature' : temperature}};

return msg;
```

### **Nodo 4:**

Este nodo nos sirve para añadir el TimeStamp al paquete de información. Nos servirá mas adelante para poder analizar datos en un intervalo de tiempo concreto.

```
msg.payload.d.ts = new Date().getTime();

return msg;
```

### **Nodo 5:**

En este nodo analizamos la temperatura y en función del valor de esta encendemos los leds de nuestro SenseHat de un color u otro. Cuanto más alto sea el valor, observaremos más leds encendidos y el color de estos serán más cálidos.

```
var temperature = msg.payload.d.temperature;

if (temperature >= 37) msg.payload = "*,*,red";

else if (temperature < 37 & temperature >= 36) msg.payload = "1-7,*,maroon,0,*,black";

else if (temperature < 36 & temperature >= 35) msg.payload = "2-7,*,yellow,0-1,*,black";

else if (temperature < 35 & temperature >= 34) msg.payload = "3-7,*,olive,0-2,*,black";

else if (temperature < 34 & temperature >= 33) msg.payload = "4-7,*,lime,0-3,*,black";
```

```

else if (temperature < 33 & temperature >= 32) msg.payload = "5-7,*,green,0-4,*,black";
else if (temperature < 32 & temperature >= 31) msg.payload = "6-7,*,aqua,0-5,*,black";
else if (temperature < 31) msg.payload = "1-7,*,teal,0-6,*,black"

else msg.payload = "ERROR";

return msg;

```

### Nodo 6:

Se envía la información a nuestro bróker de Bluemix. Este nodo es muy importante que esté bien configurado. Aquí **habrá que introducir los parámetros obtenidos al registrar nuestro dispositivo**: ID del dispositivo, ID de la organización, tipo de dispositivo y token de autenticación.

**node properties**

Connect as: Device

☐ Quickstart ☒ Registered

Credentials: raspberrypi

Event type: event

Format: json

QoS:

Name: Bluemix

Watson IoT > Edit wiotp-credentials node

Delete Cancel Update

Organization:

Server-Name: orgid.messaging.internetofthings.ibmcloud.com

Device Type: raspberrypi

Device ID: RBP001

Auth Token: \*\*\*\*\*

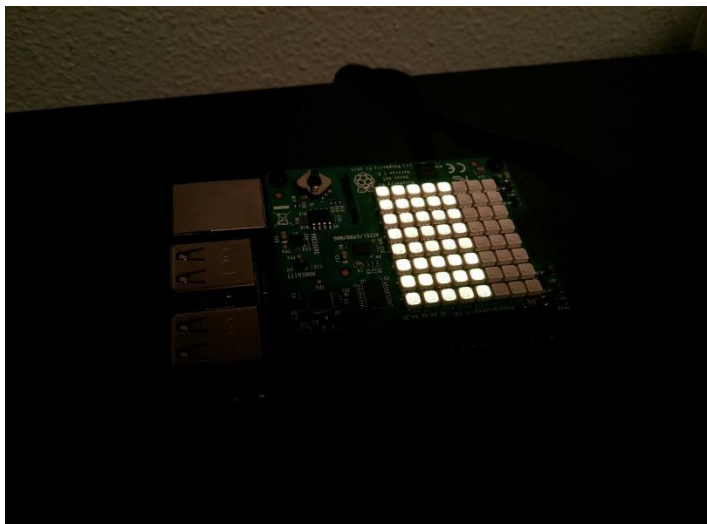
Keep Alive: 60 Seconds ☒ Use Clean Session

☐ Enable secure (SSL/TLS) connection

Name: raspberrypi

### Nodo 7:

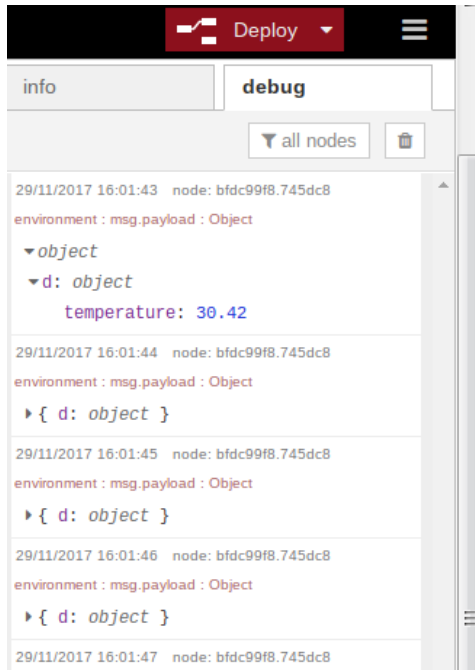
Este nodo recibe los mensajes que hemos creado en el nodo 5 y los transmite al SenseHAT consiguiendo encender los leds en función de la temperatura. Cuando hagamos deploy de nuestro programa los leds de nuestra Raspberry deberían tener un aspecto como este:





### Nodo 8:

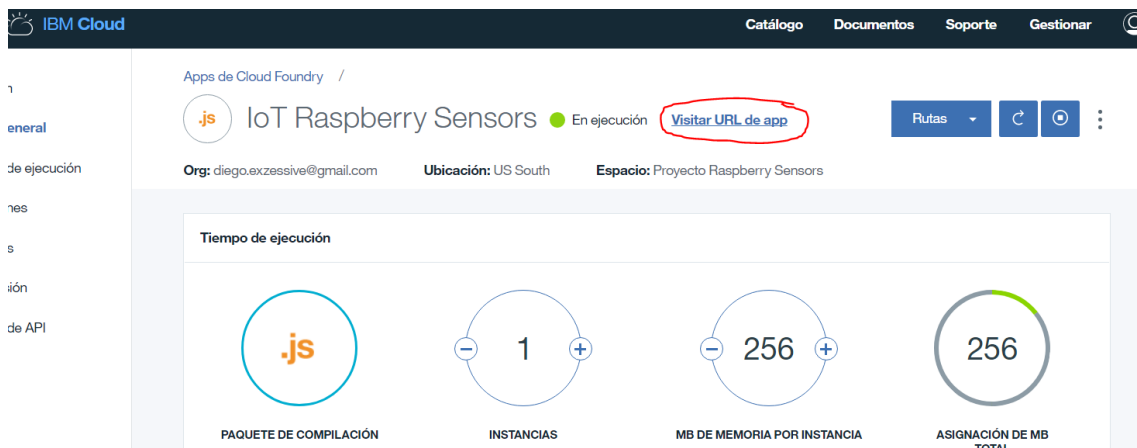
Estos nodos sirven como “debug”. Nos son útiles para observar el contenido de los mensajes y ver si estamos transmitiéndolos correctamente. No observaremos estos mensajes hasta que hayamos hecho clic en “Deploy”.



Hacemos clic en “Deploy” y nuestro programa automáticamente comienza a enviar información a la plataforma. Ahora podríamos acudir a nuestra plataforma y ver que nuestro dispositivo está conectado y que están llegando datos. Sin embargo, estos datos no se nos están almacenando.

## PASO 3: ALMACENAR DATOS

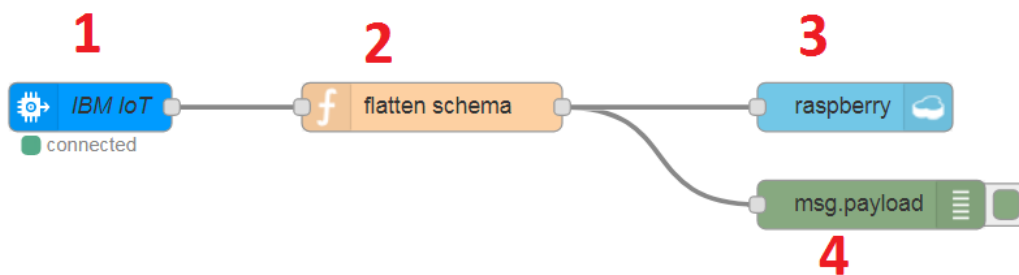
Dejamos la Raspberry a un lado, volvemos a nuestra aplicación de Bluemix y la ejecutamos clicando en el link que muestra la imagen siguiente:



Al clicar se nos abre una ventana que nos lleva a Node-RED. En un flujo importamos el siguiente código (también lo puedes encontrar en el repositorio como flujo2.txt):

```
[{"id":"c2e3b3ed.46576","type":"ibmiot
in","z":"deb0d57.1c46528","authentication":"boundService","apiKey":"","inputType":"evt","de
viceId":"","applicationId":"","deviceType":"+","eventType":"+","commandType":"","format":"j
son","name":"IBM
IoT","service":"registered","allDevices":"","allApplications":"","allDeviceTypes":true,"allEvents
":true,"allCommands":"","allFormats":"","qos":0,"x":90,"y":180,"wires":[["41f782b8.a74f5c"]]}
,{"id":"a291dbb.2cc2128","type":"cloudant
out","z":"deb0d57.1c46528","name":"","cloudant":"","database":"raspberry","service":"IoT
Raspberrry Sensors-
cloudantNoSQLDB","payonly":true,"operation":"insert","x":560,"y":180,"wires":[]},{"id":"41f78
2b8.a74f5c","type":"function","z":"deb0d57.1c46528","name":"flatten
schema","func":"msg.payload = msg.payload.d;\nreturn
msg;","outputs":1,"noerr":0,"x":300,"y":180,"wires":[["1ce8719b.1dceae","a291dbb.2cc2128"]
]},{"id":"1ce8719b.1dceae","type":"debug","z":"deb0d57.1c46528","name":"","active":true,"c
onsole":"false","complete":"false","x":570,"y":240,"wires":[]}]
```

Lo que nos debería dar algo como esto:



#### Nodo 1:

Este nodo extrae los datos que están llegando a nuestro bróker en Bluemix, es decir los paquetes que nos está enviando nuestro dispositivo Raspberry.

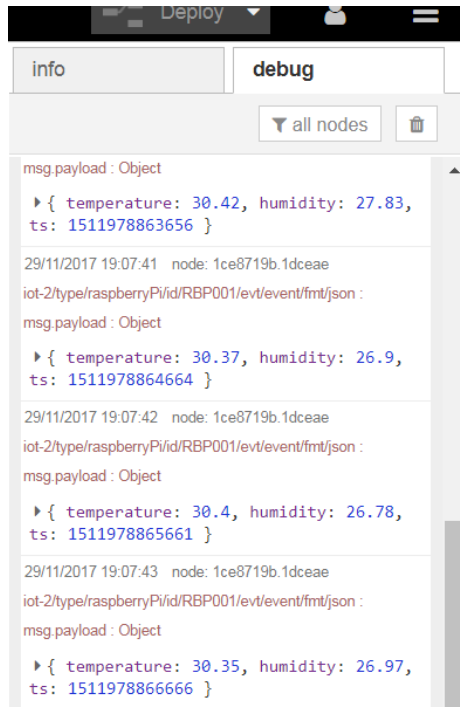
#### Nodo 2:

Este nodo adapta los mensajes recibidos para que se muestren correctamente en la base de datos. Puedes probar a quitar este nodo, conectar directamente el nodo "IBM IoT" con

“raspberry” y observar como se están almacenando en la base de datos (para hacer esto, avance un poco más en este tutorial).

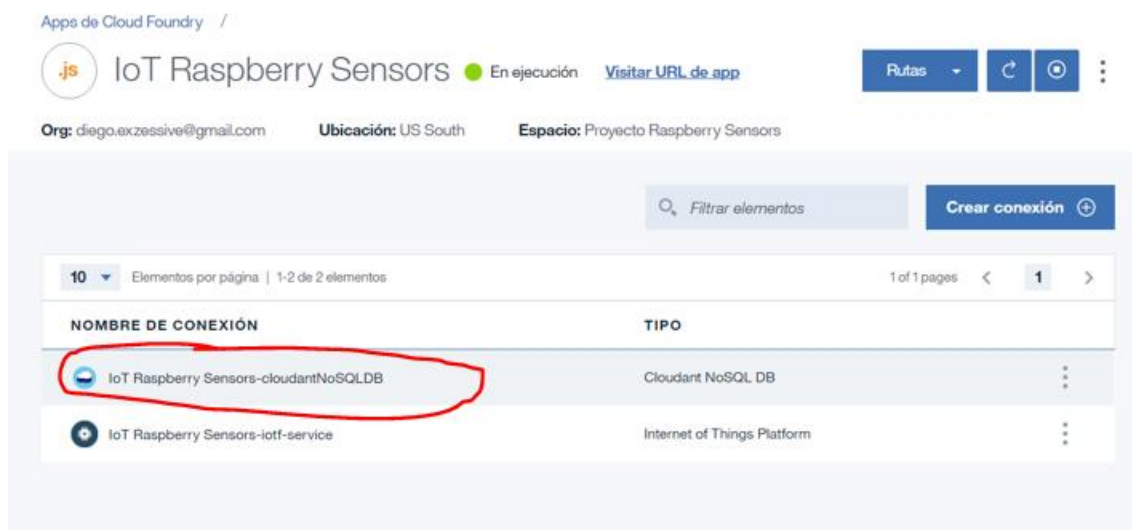
Y los envía a la base de datos “raspberry”. Esta base de datos se crea automáticamente y para comprobar que nos están llegando los datos vamos a nuestra aplicación

Hacemos clic en “Deploy” y automáticamente empezarán almacenarse los datos enviados desde la Raspberry. Podemos observar en la ventana “Debug” si nos están llegando.

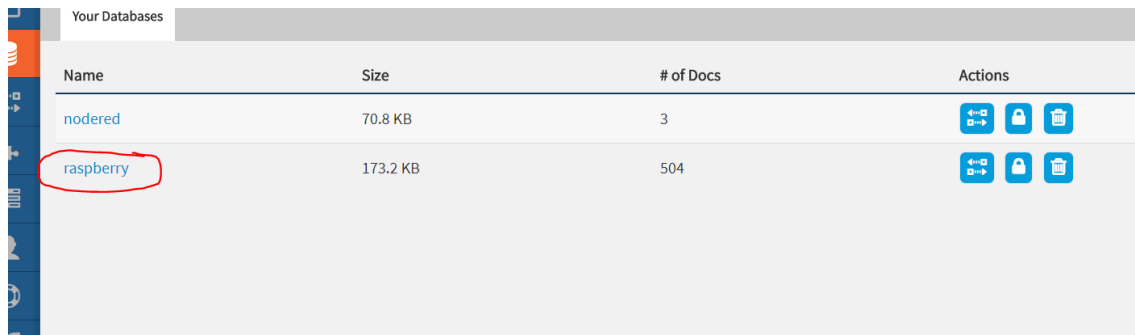








Ahora vamos a ver si realmente se están almacenando en nuestra BBDD.

Vamos a las conexiones de nuestra aplicación y hacemos clic en la conexión “cloudantNoSQLDB”.

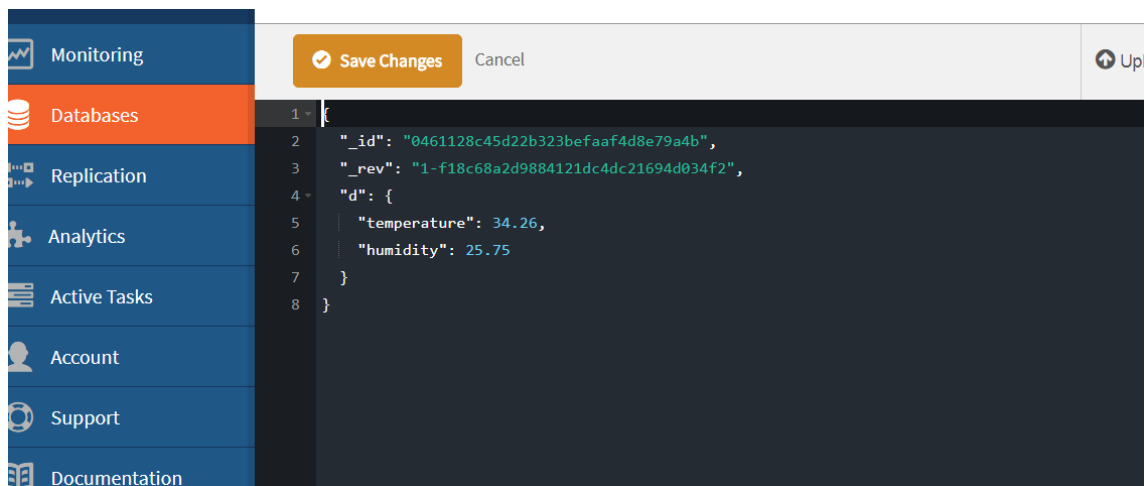



Se nos abrirá un nuevo entorno y en la sección “Data Bases” podremos ver que automáticamente se ha creado una base de datos llamada “raspberry”.





Name	Size	# of Docs	Actions
nodered	70.8 KB	3	  
raspberry	173.2 KB	504	  


Entramos en la base de datos y abrimos uno de los paquetes.





 Monitoring


 Databases


 Replication

 Analytics

 Active Tasks

 Account

 Support

 Documentation

Save Changes Cancel

```
1 {
2   "_id": "0461128c45d22b323befaaf4d8e79a4b",
3   "_rev": "1-f18c68a2d9884121dc4dc21694d034f2",
4   "d": {
5     "temperature": 34.26,
6     "humidity": 25.75
7   }
8 }
```

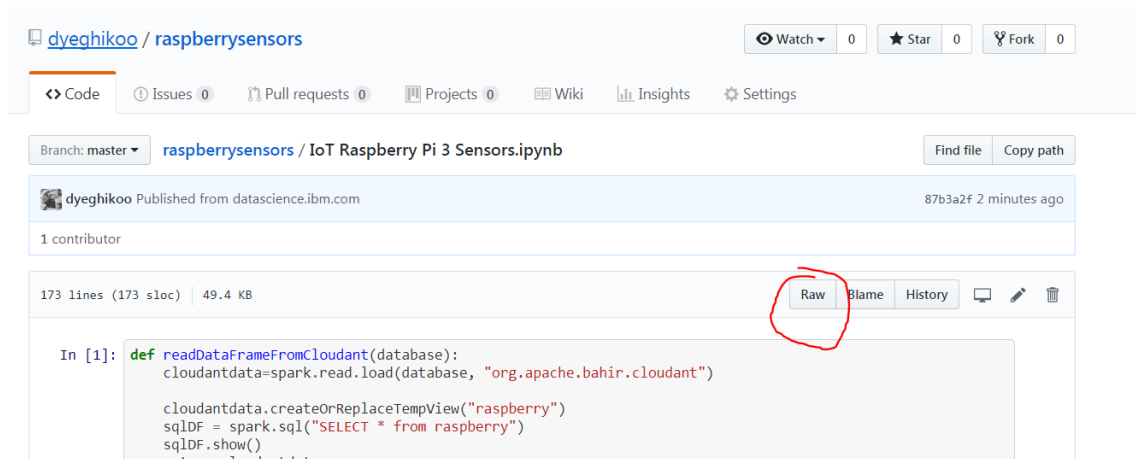
¡Ya queda menos! Ahora estaría bien que variases la temperatura de tu Raspberry con un secador, poniéndole en modo “aire ambiente” y “aire caliente”. Observarás que con el aire caliente los leds se pondrán rojos y al darle aire del ambiente tenderán hacia el color azul.

Cuando hayas experimentado un rato acuérdate de cortar la transmisión con la base de datos para que no tengamos una gran cantidad de ellos y más tarde al analizar los datos vaya más fluido el proceso.

## PASO 4: EXTRAYENDO Y ANALIZANDO LOS DATOS

Lo primero que deberemos hacer es acceder a <https://datascience.ibm.com/> y entrar con el mismo usuario que hemos estado utilizando en la nube de Bluemix.

Esta plataforma funciona con Apache Spark, con notebooks de Jupiter en los que programaremos en python. Crea un nuevo notebook con la opción crear a partir de una URL y añade el link del notebook que se encuentra en mi repositorio clicando en "Raw" y copiando la URL.



Explicaré paso a paso pero de forma resumida lo que hace este notebook:

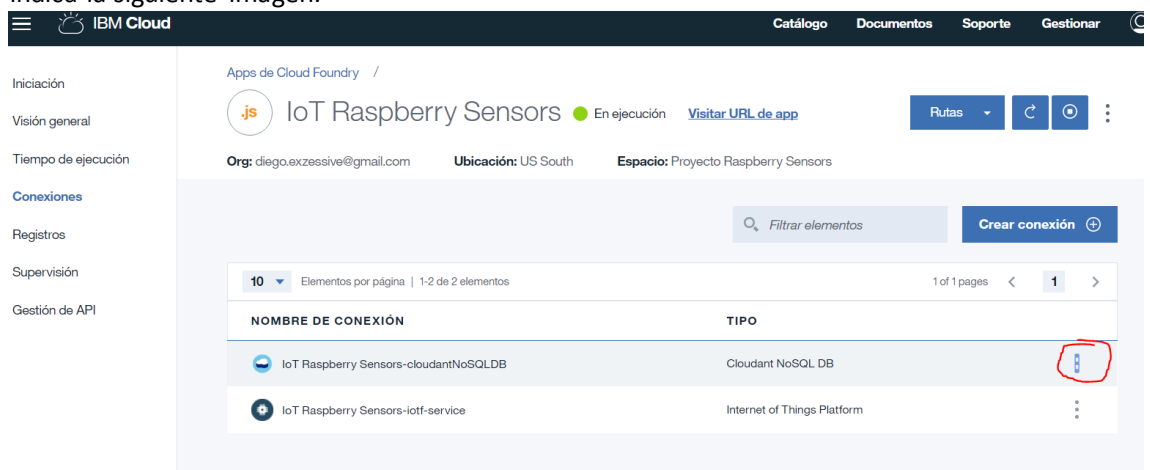
1. Definimos una función que nos mostrara la base de datos que estamos leyendo.



2. Introducimos los credenciales de nuestra base de datos.

```
In [2]: hostname = ""
user = ""
pw = ""
database = "raspberry"
```

Para ello vamos a las conexiones de nuestra aplicación en Bluemix y clicamos donde indica la siguiente imagen:



Copiamos las credenciales que nos aparecen:



Y los ponemos en su variable correspondiente:

```
In [2]: hostname = ""
user = ""
pw = ""
database = "raspberry"
```

3. Conectamos con la base de datos y la mostramos por pantalla la base de datos llamando a la función que hemos definido anteriormente. Si todo está bien no nos debería dar ningún error al ejecutar el programa.

```
In [4]: spark = SparkSession\
.builder\
.appName("Cloudant Spark SQL Example in Python using temp tables")\
.config("cloudant.host",hostname)\
.config("cloudant.username", user)\
.config("cloudant.password",pw)\
.getOrCreate()
cloudantdata=readDataFrameFromCloudant(database)
```

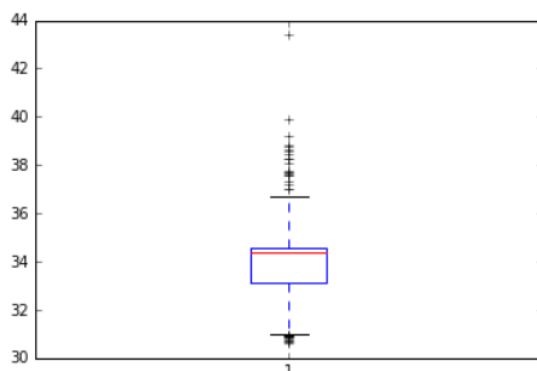
4. Mostramos los 15 primeros valores de la columna "temperatura"

```
In [5]: result = spark.sql("SELECT temperature from raspberry")
rdd = result.rdd.map(lambda row : row.temperature).collect()
rdd[:15]
```

5. Importamos la librería "matplotlib" y mostramos los datos de la temperatura. En esta gráfica podemos ver gran cantidad de información como el mínimo, máximo, media, desviación típica, etc.

```
In [6]: %matplotlib inline
```

```
In [7]: import matplotlib.pyplot as plt
plt.boxplot(rdd)
plt.show()
```

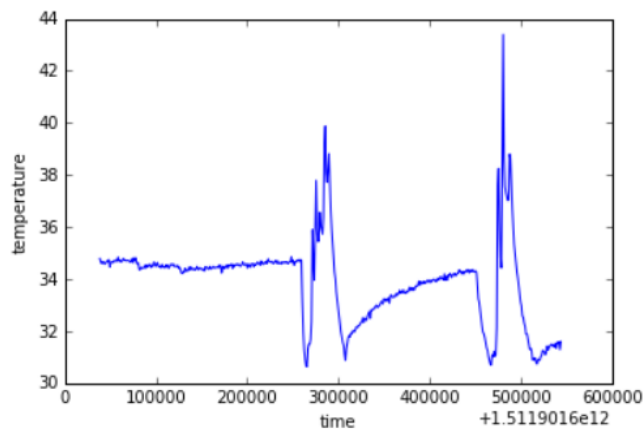


- Ordenamos los datos en orden ascendente con respecto al tiempo por si algún paquete hubiese entrado antes que otro en la base de datos.

```
In [8]: result = spark.sql("SELECT temperature,ts from raspberry order by ts asc")
rdd = result.rdd.map(lambda row : (row.ts,row.temperature))
rdd_temperature = rdd.map(lambda (ts,temperature): temperature).collect()
rdd_ts = rdd.map(lambda (ts,temperature): ts).collect()
print rdd_temperature[:15]
print rdd_ts[:15]
```

- Pintamos una gráfica de la temperatura con respecto al tiempo. Como puedes observar, los picos que aparecen corresponden a dos momentos en los que le he dado calor al SenseHAT.

```
In [9]: plt.plot(rdd_ts,rdd_temperature)
plt.xlabel("time")
plt.ylabel("temperature")
plt.show()
```



Si quisiésemos observar un periodo de tiempo en concreto haríamos lo siguiente.

- Obtenemos mínimo y máximo del tiempo.

```
In [10]: spark.sql("select min(ts),max(ts) from raspberry").show()
```

SPARK JOB PROGRESS

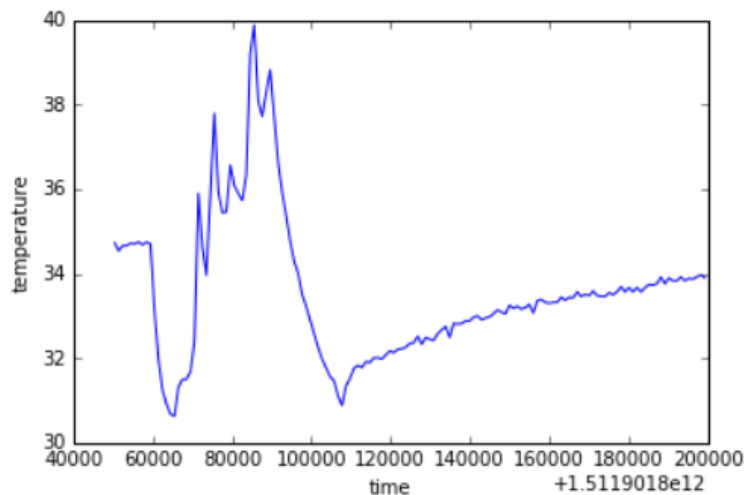
Hide All ▲

JOB	PROGRESS	DURATION	STATUS
6	2 stages	2.21 sec	▼

```
+-----+-----+
| min(ts) | max(ts) |
+-----+-----+
|1511901638144|1511902144191|
+-----+-----+
```

## 9. Mostramos la gráfica en un periodo de tiempo concreto.

```
In [11]: result = spark.sql("SELECT temperature,ts from raspberry where ts > 1511901850000 and ts <= 1511902000000 order by ts asc")
rdd = result.rdd.map(lambda row : (row.ts,row.temperature))
rdd_temperature = rdd.map(lambda (ts,temperature): temperature).collect()
rdd_ts = rdd.map(lambda (ts,temperature): ts).collect()
plt.plot(rdd_ts,rdd_temperature)
plt.xlabel("time")
plt.ylabel("temperature")
plt.show()
```



Ya hemos llegado al final. Te recomiendo que hagas lo mismo con la humedad. Si has trabajado anteriormente con Python sabrás que se pueden representar los datos de otras formas. En caso de no saberlo te recomiendo que busques información acerca de ello.

También te recomiendo que busques información acerca del protocolo MQTT y como aplicarlo para extraer los datos de una API. Esto te servirá para poder trabajar con los datos en tiempo real.

¡Muchas gracias por leer este tutorial! Puedes ver otros proyectos aquí

<https://github.com/dyeghikoo>

Si quieres contactar conmigo puedes escribirme a [diegosier28@hotmail.com](mailto:diegosier28@hotmail.com).

