Demostrador de Internet of Things con la tecnología IEEE 802.15.4e utilizando el sistema operativo *OpenWSN, OpenSim y theThings.io*

Manuel Márquez Salas

Máster Universitario en Ingeniería de Telecomunicación (UOC-URL)

JUN/2017

UOC
Universitat Oberta de Catalunya

Índice

Objetivos y motivación

- Desarrollo de un demostrador de IoT basado en la herramienta OpenWSN sobre Rpi, simulando una situación doméstica que permita la gestión y almacenado de datos en una base de datos local, subiendo la información compilada y resumida a la nube (theThings.iO).

- Comparar resultados entre la solución clásica de procesado en la nube y en el Edge con la presente aplicación.

Otros objetivos

- Estudiar las tecnologías IoT actuales.
- Entender la importancia del Edge Computing.
- Conocer y utilizar la herramienta de software libre *OpenWSN*.
- Conocer cómo funciona la plataforma theThings.iO y qué servicios ofrece.

Internet of Things (IoT)

- Conectividad para *anything, anytime, any place and for everyone. Idea principal: conectar a Internet cualquier objeto con cualidades heterogéneas.*

- *Protocolos: IEEE802.15.4-2006, IEEE802.15.4e, IETF 6LoWPAN, IETF CoAP….*

Cloud Computing

- Tecnología que ofrece servicios y aplicaciones de cálculo a través de Internet.

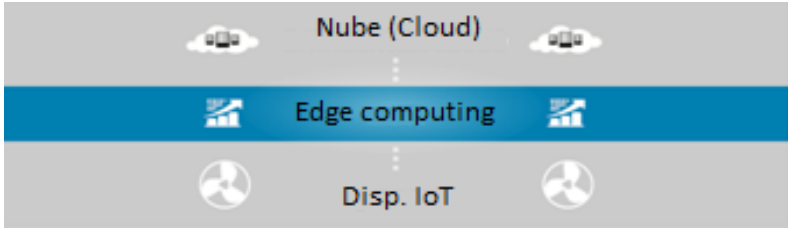
→ Para liberar **recursos de usuario** ←

- Objetivos:

  - Eficiencia → HW / SW bajo demanda.
  - Potencia escalable → según necesidad.
  - Flexibilidad → independencia de localización.
  - Abstracción → no depende infraestructura.

Edge Computing

- Acercamiento de nube a los dispositivos ioT que necesitan una acción en un tiempo de reacción corto.



- Objetivo:

  - Minimizar latencia de procesado.
  - Gestionar Ancho de banda de manera eficiente.
  - Aumentar la seguridad de los datos.

- ¿¿ Como ?? → **Acercar el procesado a los dispositivos que generan los datos**.

Cloud vs Edge computing

Parámetros	*Cloud Computing*	*Edge Computing*
Localización Servidor/es	Internet	Dentro de red local (Edge)
Distancia (cliente --> servidor)	Multiples Hops	Un único Hop
Latencia	Alta	Baja
Variación de Latencia	Alta fluctuación	Baja fluctuación
Seguridad	- Segura, indefinida	+ Segura, se puede definir
Información de localización	No	Si
Vulnerabilidad	Alta probabilidad	Baja probabilidad
Distribución	Centralizada	Densa y distribuida
Nº nodos servidores	Pocos	Elevado
Respuesta en Tiempo real	Si	Si
Conexión a Internet	Dedicada	Inalhámbrica
Soporte Mobilidad	Limitada	Si

OpenWSN

Implementación *open-source* en lenguaje C de la pila de protocolos estándar del Internet de las cosas (*IoT*) como son el *6LoWPAN*, *IEEE802.15.4e* y *CoAP*.

application	CoAP, HTTP
transport	UDP, TCP
IP/routing	IETF RPL
adaptation	IETF 6LoWPAN
medium access	IEEE802.15.4e
phy	IEEE802.15.4-2006

Pila de protocolos:

→ OpenOS: es el planificador principal.

→ OpenVisualizer: Programa de visualización y depuración (Gui, Cli y Web)

→ OpenSim: simulador de nodos (motes), utilizando misma pila de protocolos que nodos físicos .

→Librería CoAP para Python.

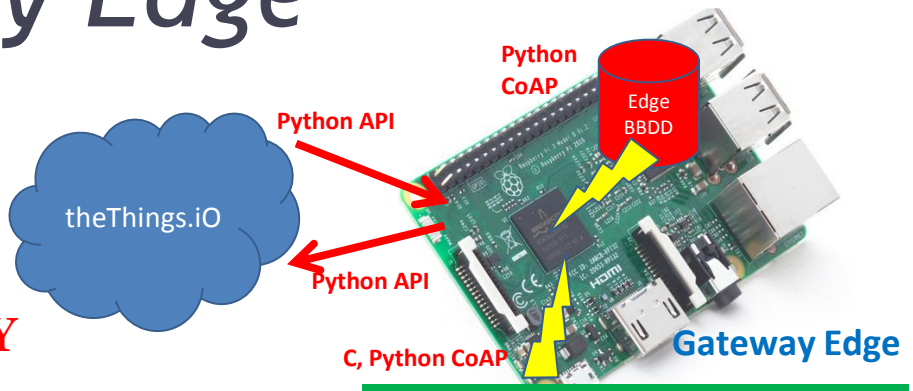
Thethings.iO

Plataforma de Cloud computing para los dispositivos ioT.

→ Código de activación compuesto por un Token (thingToken).
→ 15 días de acceso gratuito.
→ Librería de Python para implementar llamadas/métodos hacia su API.

- addVar(key, value, dt=None): Añade la pareja de datos {key : value} al buffer de datos pendiente de escribir.
- write(): petición de escritura del buffer , par {key : value}.
- read(key, limit=1, startDate=None, endDate=None, to=10): petición a la API del par {key : value}.

Desarrollo: Gateway Edge



- Rpi 3 (B) con SO OpenWSN → GATEWAY

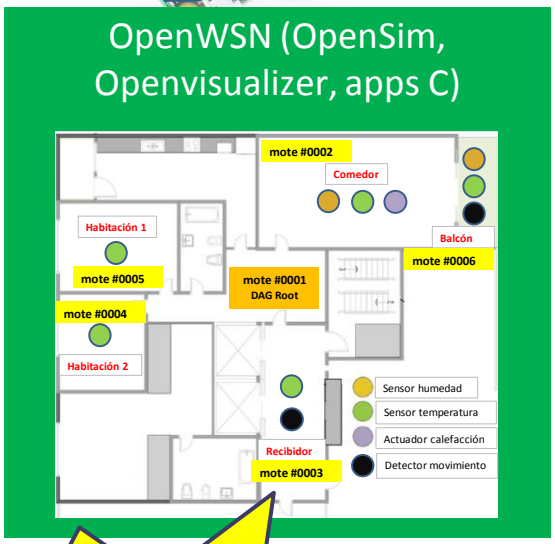
- Desarrollo de 4 apps dentro del firmware de OpenWSN (ctempo, chumidity, ccalef y cmotion), escritas en C.

- Utilización de OpenSim (Openvisualizer).

- Interactuación usando protocolo CoAP, métodos GET y PUT.

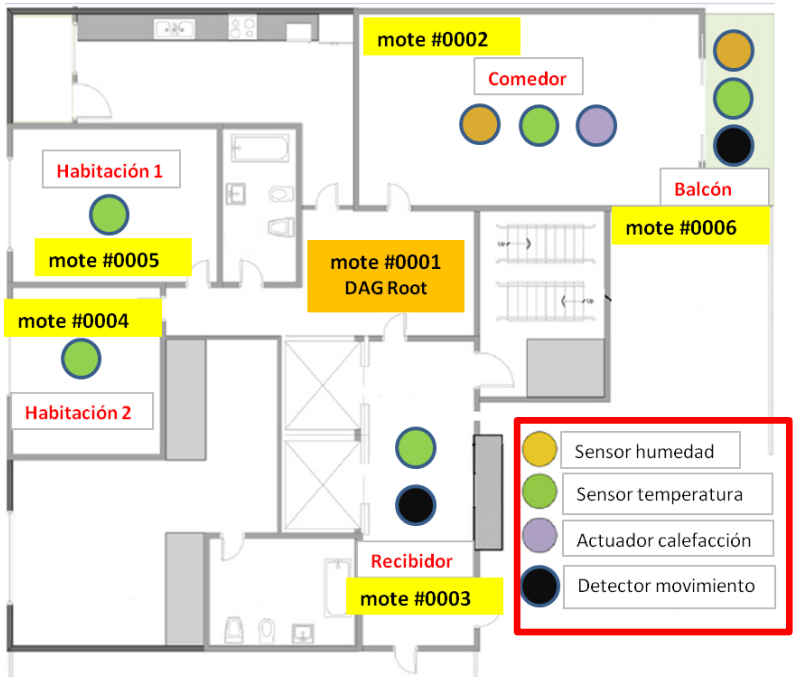
- Grabación de datos en la bbdd en MySQL.

- Subida de datos resumen al Cloud (theThings.iO) usando Python API.

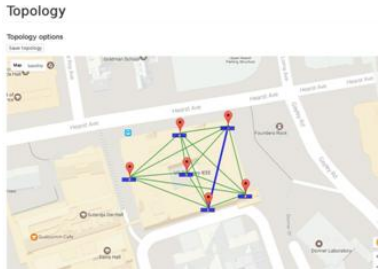


Sin HW adicional

Desarrollo: despliegue de sensores



- Layout de despliegue sensores y funciones dentro de OpenWSN (openwsn-fw).



- 5 sensores, 1 DAG-root.

- Implementado dentro del openwsn-fw en lenguaje C.

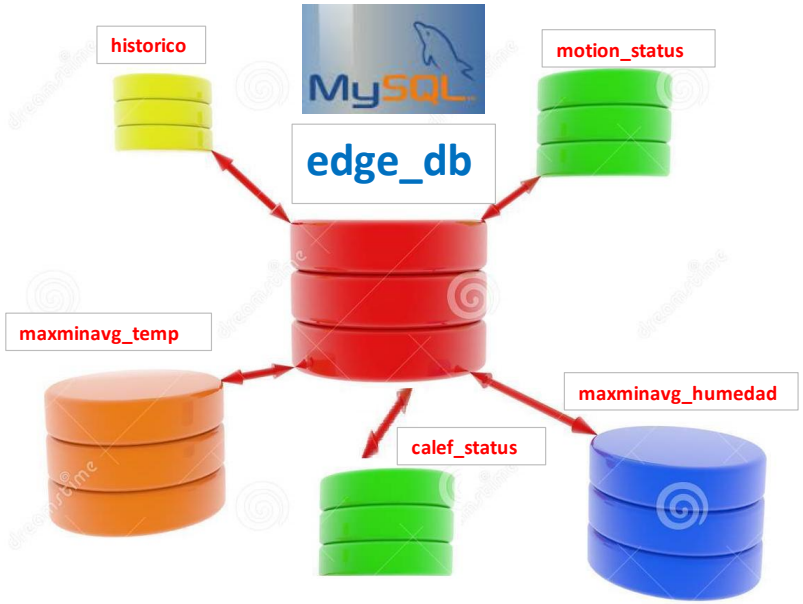
- Necesidad de función que genere valores aleatorios de temperatura, humedad y movimiento .

mote	*alias*	DAG root	Temperatura	Humedad	Movimiento	Calefacción
#0001	DAG	X	-	-	-	-
#0002	Comedor	-	X	X	-	X
#0003	Recibidor	-	X	-	X	-
#0004	Hab. 2	-	X	-	-	-
#0005	Hab. 1	-	X	-	-	-
#0006	Balcón	-	X	X	X	-

```
// GENERACION DE VALORES RANDOM DE TEMPERATURA DENTRO DE UN RANGO
// DEFINIDO

int ctempo openrandomtemp (){

    int resultado = 0, low num = 0, hi num = 0;
    min num=15;
    max num=25;

    if (min num < max num)
    {
        low num = min num;
        hi num = max num + 1; // include max num in output
    } else {
        low num = max num + 1; // include max num in output
        hi num = min num;
    }

    srand(time(NULL));
    resultado = (rand() % (hi num - low num)) + low num;
    printf("Min=15 Max=25 TEMP=%d\n",resultado);
    return resultado;
```

Función *openrandomtemp()* encargada de generar una temperatura aleatoria

Desarrollo: estructura de base de datos



- Base de datos basada en MySQL (edge_db).

→ 5 tablas:

- Datos históricos (todos los datos generados).
- Estado de actuador calefacción (“0” Des, “1” Act).
- Estado de sensor movimiento (“0” sin mov., “1” mov. Detectado).
- Datos resumidos (cada 3 horas), valores MAX, MIN y AVG.

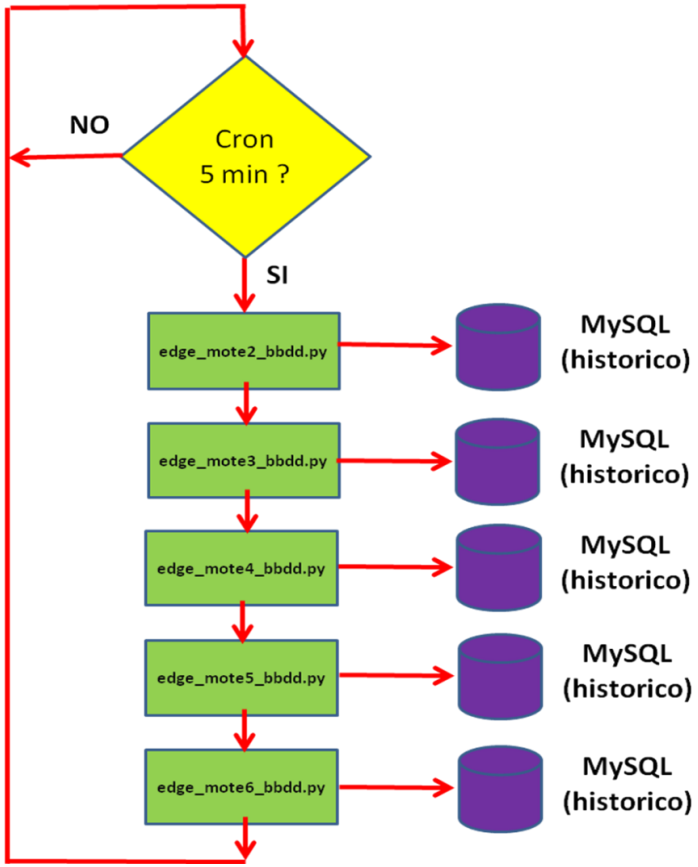
```
mysql> SELECT * FROM maxminavg_temp;
+--------+------------+----------+----------+----------+----------+
| sensor | dia        | hora     | temp_max | temp_min | temp_avg |
+--------+------------+----------+----------+----------+----------+
| 02     | 2017-05-14 | 17:18:50 | 25       | 25       |       20 |
| 02     | 2017-05-14 | 17:19:34 | 25       | 25       |       20 |
| 02     | 2017-05-14 | 17:20:39 | 25       | 25       |       20 |
| 02     | 2017-05-14 | 17:23:30 | 25       | 25       |       20 |
| 02     | 2017-05-14 | 17:24:39 | 25       | 25       |       20 |
| 02     | 2017-05-14 | 17:25:23 | 25       | 25       |       20 |
| 02     | 2017-05-14 | 17:25:57 | 25       | 25       |       20 |
| 02     | 2017-05-14 | 17:26:23 | 25       | 25       |       20 |
| 02     | 2017-05-14 | 17:27:06 | 25       | 25       |       21 |
| 02     | 2017-05-14 | 17:28:08 | 25       | 25       |       21 |
| 02     | 2017-05-14 | 17:30:58 | 25       | 25       |       20 |
| 02     | 2017-05-14 | 17:32:01 | 25       | 25       |       21 |
| 02     | 2017-05-14 | 17:34:02 | 25       | 25       |  20.5455 |
| 02     | 2017-05-14 | 17:34:40 | 25       | 25       |  20.5455 |
| 02     | 2017-05-14 | 17:35:10 | 25       | 25       |  20.5455 |
| 02     | 2017-05-14 | 17:36:16 | 25       | 25       |  20.5455 |
| 02     | 2017-05-14 | 17:37:14 | 25       | 16       |  20.5455 |
| 02     | 2017-05-14 | 17:40:44 | 25       | 16       |  20.5455 |
| 02     | 2017-05-14 | 17:45:36 | 25       | 16       |  20.5455 |
```

→ tabla *maxminavg_temp (timestamp, valor temp_max, temp_min y temp_avg)*

Desarrollo: secuenciación procesado Edge

→ Script en crontab:

- Cada 5 min inicio secuencia de lectura de datos.
- Actuación sobre calefacción si es necesario (según temperatura asignada).
- Activación de Alarma si existe movimiento.
- Grabación en bbdd local.



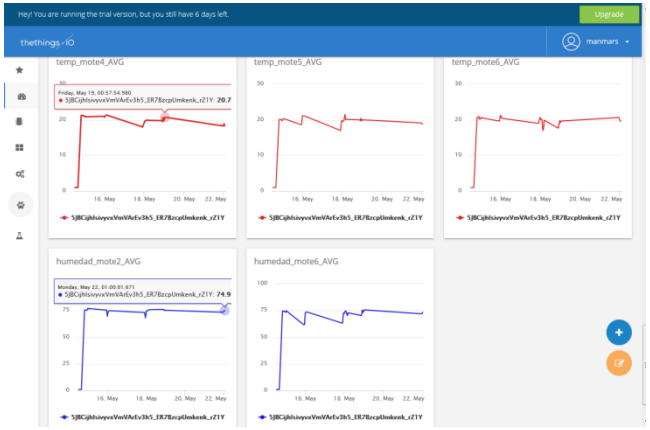

```
*/5 * * * *     /home/pi/openwsn/scripts/mote2cloud.sh && /home/pi/openwsn/scripts/delay.sh &&
 /home/pi/openwsn/scripts/mote3cloud.sh && /home/pi/openwsn/scripts/delay.sh &&
/home/pi/openwsn/scripts/mote4cloud.sh && /home/pi/openwsn/scripts/delay.sh &&
/home/pi/openwsn/scripts/mote5cloud.sh && /home/pi/openwsn/scripts/delay.sh &&
/home/pi/openwsn/scripts/mote6cloud.sh

*/180 * * * *   /home/pi/openwsn/scripts/edge_maxminavg_theThingsIO.sh
```

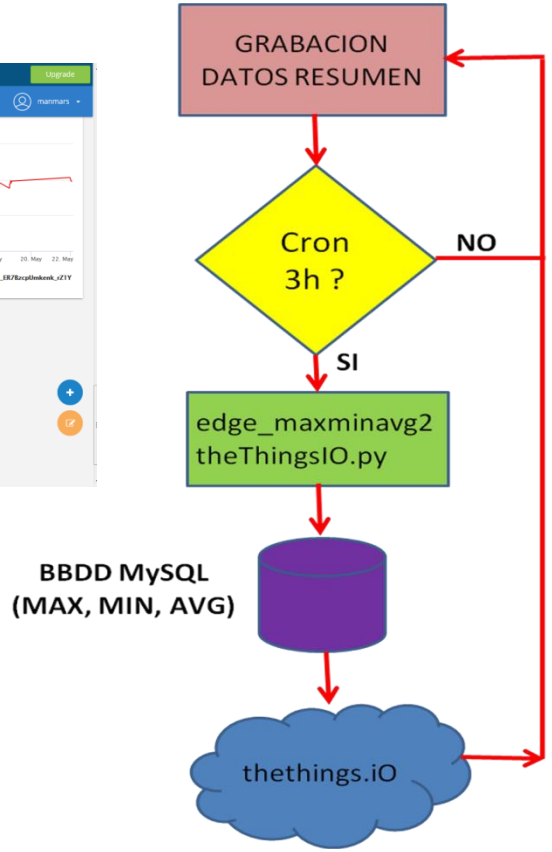
Secuenciación mediante script usando *Crontab*

Desarrollo: secuenciación procesado Edge

→ Script en crontab:

- Cada 3 horas inicio script de escritura de datos resumidos (valores máximos, mínimos y medias)
- Grabación en bbdd local.
- Subir datos resumen al Cloud (theThings.iO).





Thing write
5JBCijhIsivyvxVmVArEv3h5_ER7BzcpUmkenk_rZ1Y

[{"value":"17/05/22","key":"fecha"},{"value":"21:49:45","key":"hora"},{"value":25,"key":"temp_mote2_MAX"},
{"value":25,"key":"temp_mote3_MAX"},{"value":25,"key":"temp_mote4_MAX"},{"value":25,"key":"temp_mote5_MAX"},
{"value":25,"key":"temp_mote6_MAX"},{"value":15,"key":"temp_mote2_MIN"},{"value":16,"key":"temp_mote3_MIN"},
{"value":16,"key":"temp_mote4_MIN"},{"value":15,"key":"temp_mote5_MIN"},{"value":16,"key":"temp_mote6_MIN"},
{"value":19.523809523809526,"key":"temp_mote2_AVG"},{"value":21.1875,"key":"temp_mote3_AVG"},
{"value":19.625,"key":"temp_mote4_AVG"},{"value":20.705882352941178,"key":"temp_mote5_AVG"},
{"value":21.266666666666666,"key":"temp_mote6_AVG"},{"value":93,"key":"humedad_mote2_MAX"},
{"value":90,"key":"humedad_mote6_MAX"},{"value":55,"key":"humedad_mote2_MIN"},{"value":58,"key":"humedad_mote6_MIN"},
{"value":75.04761904761905,"key":"humedad_mote2_AVG"},{"value":73.6,"key":"humedad_mote6_AVG"}]

Desarrollo: secuenciación procesado Cloud

→ Script en crontab:

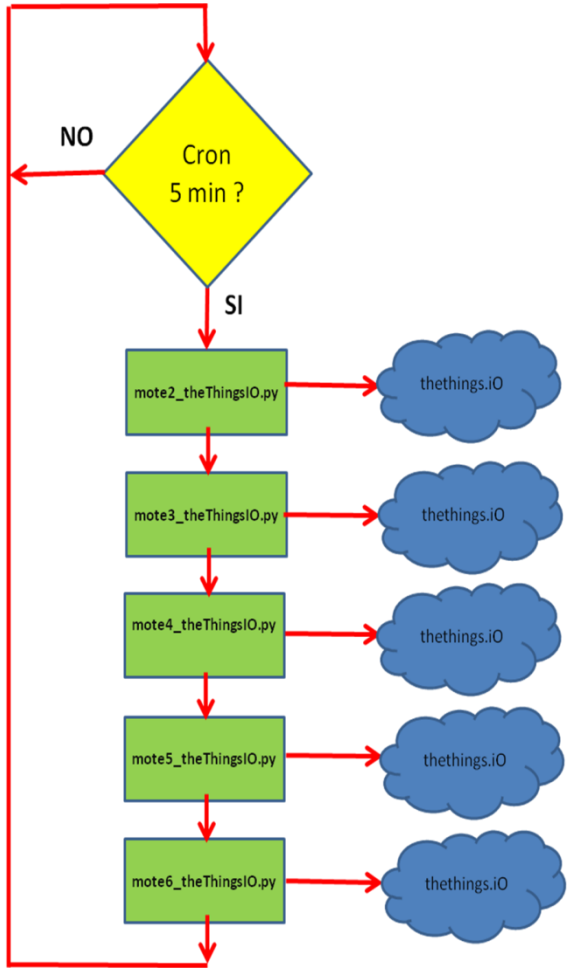
- Cada 5 min inicio secuencia de lectura de datos.
- Grabación directa en el Cloud (cada sensor).

```
79  ##########################################################################
80
81  tt.addVar("fecha", fecha);
82  tt.addVar("hora", hora);
83  tt.addVar("temp_mote2", temp, dt=None);
84  tt.addVar("humedad_mote2", humedad);
85  tt.addVar("calef_mote2", calef);
86
87  tt.write();
88  tt.clear();
89
```

- Cloudcode: actuar sobre la calefacción, Procesado en la nube.

encender_Calefaccion

```
function trigger(params, callback){
  console.log('¡¡ Se ha activado la alarma !!')
  //ignore non write events
  if(params.action !== 'write') return callback()

  var values = params.values
  var thingToken = params.thingToken

  //iterate over the values of the write
  for(var i=0; i<values.length; ++i){
    if(values[i].key === 'temp_mote2' && values[i].value <= 21){
      console.log('hace mucho frio %d/n',values[i].value)
      email(
        {
          service : 'SendGrid',
          auth: {
            api_user: 'manmarsal',
            api_key: 'YOUR API KEY'
          }
        },
        {
          from: 'manmarsal@gmail.com',
          to: 'manmarsal@gmail.com',
          subject : 'La temperatura es muy baja',
          text : 'HACE MUCHO FRIO! \n\n Saludos,\n '+ thingToken
        }
      )
    }
  }
  //end the trigger
  callback()
}
```



Desarrollo: secuenciación procesado Cloud

→Script en crontab:

- Job en el Cloud, cálculo diario (valores máximos, mínimos y medias).

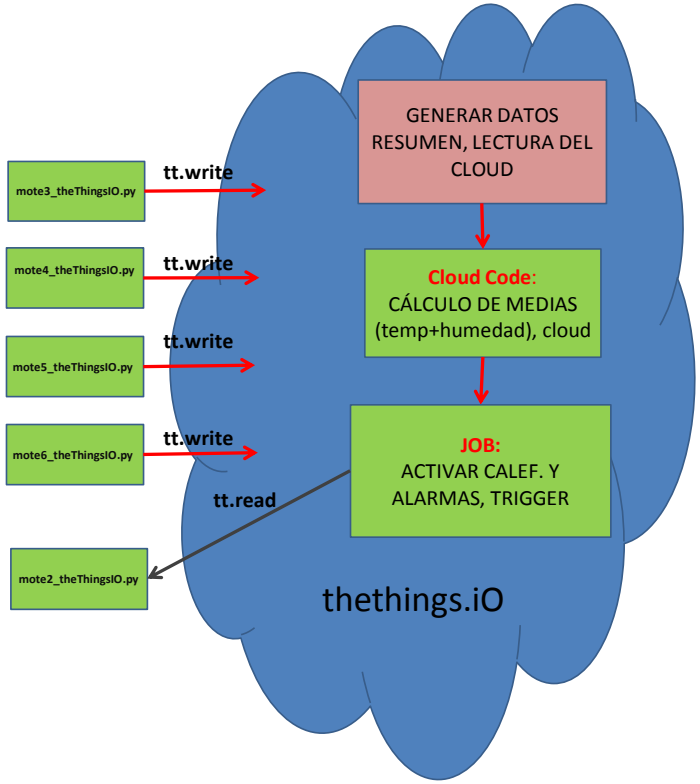
Job

Nombre: mote2_MaxMinAVG

Frecuencia: Diaria

Código:

```
function job(params, callback){
    analytics.events.getValuesByName('temp_mote2',function(error, data){
        var high0 = data.filter(function(val){
            return val > 0;
        }).avg();
        var low0 = data.filter(function(val){
            return val < 0;
        }).avg();
        analytics.kpis.create('MaxMinAVG_Temperatura',{ high: high0, low: low0 });
        callback();
```



Resultados

	theThings.iO	Rpi Edge
Tiempo ejecución x Script (mote)/seg	15	13
Tiempo ejecución x Script general (mote)/seg	29	34
Tiempo ejecución x Script (datos resumen)/seg	3,13	3,2
Tiempo de ocupación red (mote)/seg	2	0
Tiempo de ocupación red (datos resumen)/seg	3	3

→Tiempos de ejecución parecidos pero uso de red intensivo en el caso del Cloud por el contrario del procesado Edge.

	theThings.iO	Rpi Edge	Amazon AWS
Llamadas a la API (máx)	50000	n/a	n/a
Cloud code (máx)	10000	n/a	n/a
Llamadas sensor (cadencia/minutos)	2	2	2
Sensores (aplicación)	5	5	5
Llamadas diárias	14400	4800	
Llamadas mensuales	432000	144000	ilimitado
Cloud code diários (valores MAX, MIN, AVG), una cada hora	24	8	24h/dia
Cloud code mensual = instancias (AMAZON)	720	240	720
COSTE (€/mes)	29	-	2906,64

→Diferencia de coste notables.

Edge	
bytes datos resumen	1.500
bytes resumen/dia	36.000
bytes resumen/mes	1.080.000
bytes resumen/anuales	12.960.000
Cloud	
bytes/subida sensor /paquete	547
sensores	5
llamadas / hora	30
llamadas / hora / 5 sensores	150
bytes / hora / red 5 sensores	82.050
bytes / dia	1.969.200
bytes /mes	59.076.000
bytes / año	708.912.000

→Diferencia en la cantidad de datos generados y subidos al Cloud.

Conclusiones

- Se ha realizado un demostrador de ioT utilizando el sistema operativo *OpenWSN* simulando una situación doméstica → Al ser una simulación el HW (Rpi) es importante para determinar el nº de motes final soportados.

- Se ha comprobado la diferencia a nivel de uso de red entre la solución Cloud y Edge debida a la diferente cantidad de datos a subir, siendo la primera la que necesita una conexión a Internet continua.

- Se ha visto la importancia de la latencia en un aplicación domestica donde se han de tomar decisiones rápidamente.

- Se han calculado los tiempos de ejecución de cada script y se han comparado.

- Se han comparado varias opciones en términos de coste siendo la opción Rpi Edge mucho más económica para una aplicación doméstica.

- Aumenta la seguridad de los datos por el hecho de procesarlos en el Edge (misma red local).

Gracias por su atención.

Manuel Márquez Salas
manmarsal@uoc.edu