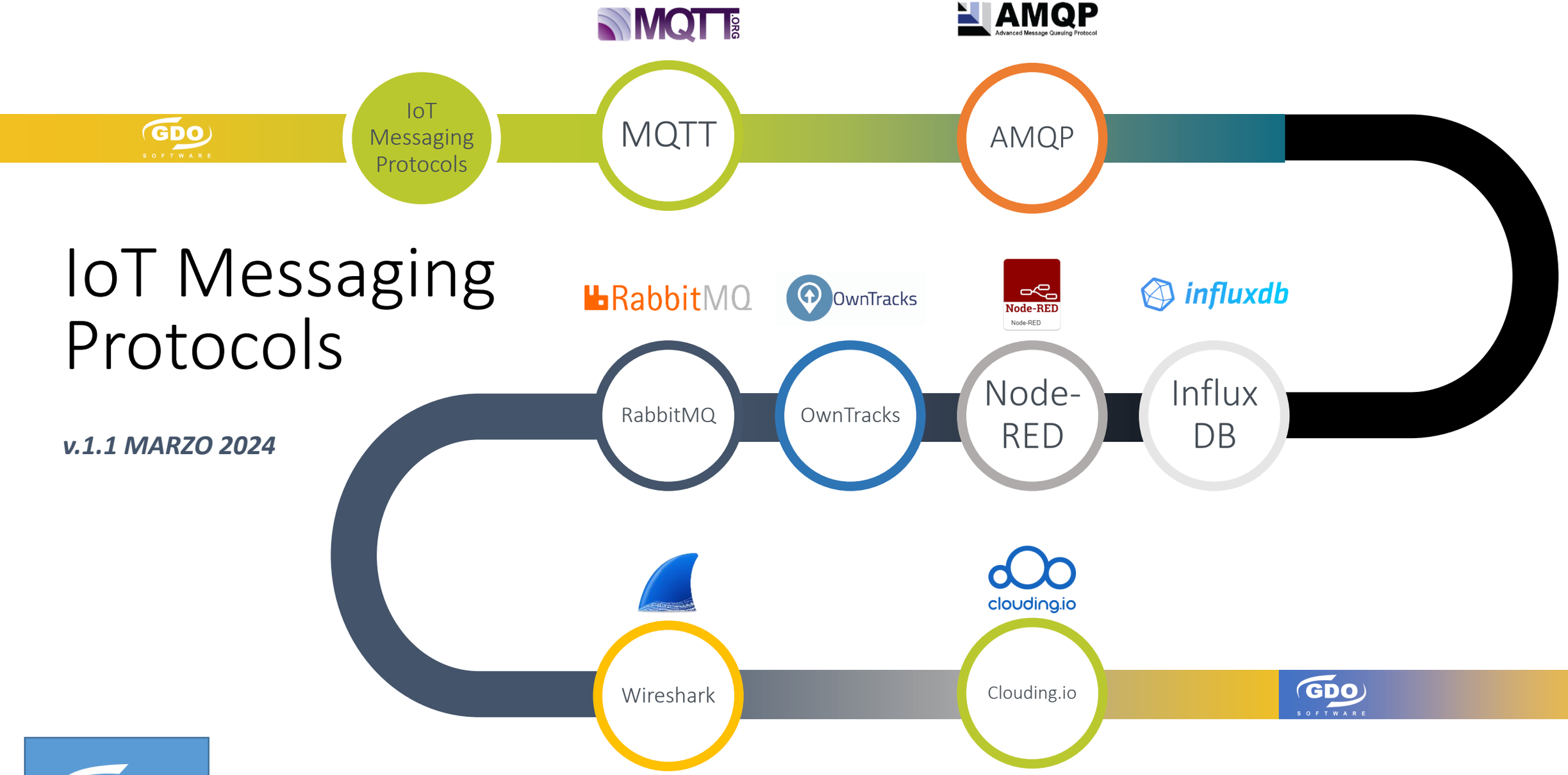
CAPÍTULO 10. IoT Messaging Protocols

v.1.1 MARZO 2024

Ricardo Moraleda Gareta

[Director departamento de software de GDO Software]



IoT Messaging Protocols

v.1.1 MARZO 2024

GDO SOFTWARE

Protocols

Protocolos mensajería

Vamos a repasar los 2 protocolos de mensajería más extendidos en el mundo IoT (sobre TCP/IP).

AMQP 0-9-1: Advanced Message Queuing Protocol

MQTT 3.1.1: MQ Telemetry Transport

http://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf

http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html

Realizaremos una breve comparativa y analizaremos sus tramas con el analizador de protocolos de red "Wireshark" en un ejemplo con Node-RED.

Similitudes: Se basan en un middleware/**broker** y usan el patrón publicador/suscriptor (pub/sub) ó broker pattern.

-Espacio desacoplado: Publicador y suscriptor no tienen por qué saber el uno del otro (IP, puerto, etc.).

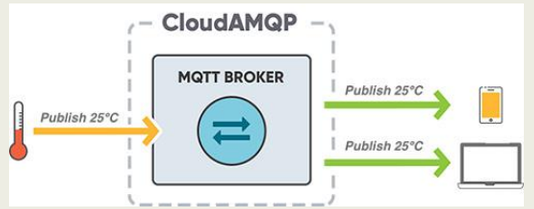
-Tiempo desacoplado: Publicador y suscriptor no tienen por qué estar en ejecución al mismo tiempo (asíncronos).

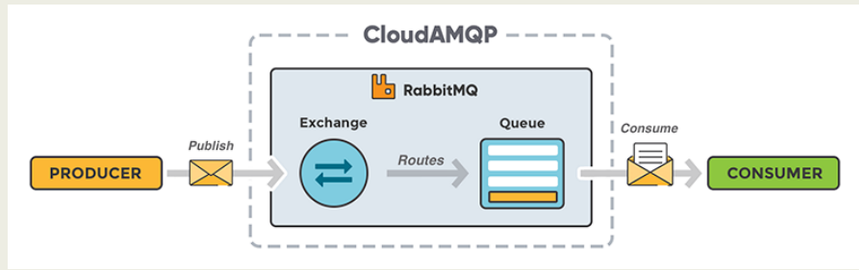
-Sincronización desacoplada: Las operaciones de los 2 componentes no se detienen durante envío o recepción.

Diferencias:

AMQP permite el encolado de mensajes mientras que MQTT sólo puedes tener el último mensaje por cada tópico, no una cola de mensajes.

El uso de MQTT es para tiempo real y que si se pierde un mensaje o varios no es del todo importante (envío temperatura) y AMQP para cerciorarse de que todos los mensajes llegan a destino encolándose si es necesario (pedidos online de compra). Incluso una muy buena opción es utilizar ambos de manera combinada.



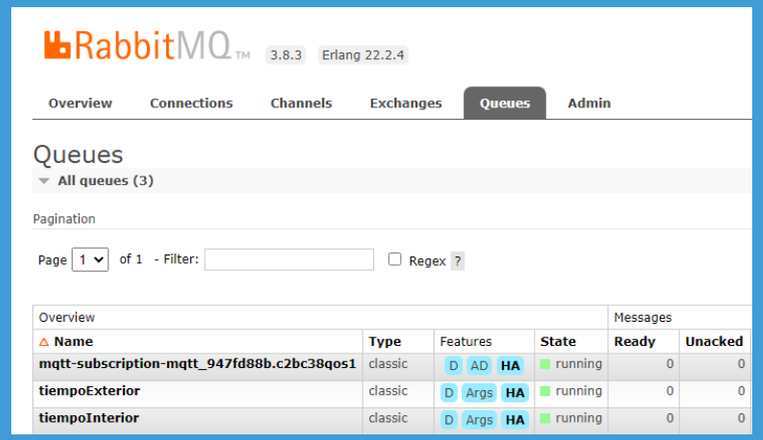


Broker AMQP/MQTT

RabbitMQ

En otros capítulos hemos utilizado Ubidots y Mosquitto como brokers MQTT. En este caso utilizaremos uno para ambos protocolos llamado RabbitMQ y más concretamente su versión "as a service" en cloud (https://www.cloudamqp.com/)

El sistema de colas nos permite soportar picos de tráfico, ya que si los servicios o las bases de datos no son capaces de soportar una carga elevada los mensajes simplemente se encolarán y no serán descartados.



AMQP 0-9-1

Los mensajes se publican en **exchanges**, que se podrían ver como un buzón. Los exchanges, al recibir mensajes, los distribuyen a las colas (**queues**) siguiendo unas reglas llamadas **bindings**. A continuación, el broker entrega los mensajes a los consumidores que están suscritos a las colas, o los propios consumidores consultan la cola y leen los mensajes.

Las redes pueden tener problemas y las aplicaciones pueden fallar al procesar los mensajes, por eso mismo, el modelo AMQP hace uso de ACKs. Cuando un mensaje se entrega a un consumidor, éste debe notificar al broker que ha procesado el mensaje, ya sea de forma automática o cuando el desarrollador de la aplicación lo decida. El broker solamente eliminará el mensaje de la cola cuando éste haya sido confirmado.

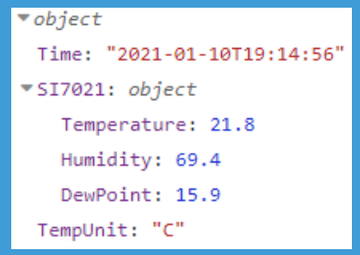
Patrón arquitectónico STORE and FORWARD

https://reactiveprogramming.io/blog/es/patrones-arquitectonicos/store-and-forward

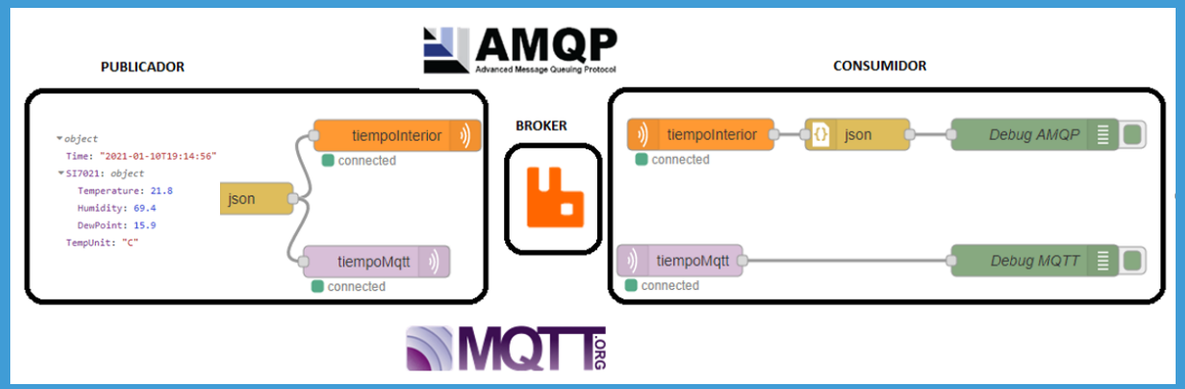
Ejemplo Node-RED

Node-RED

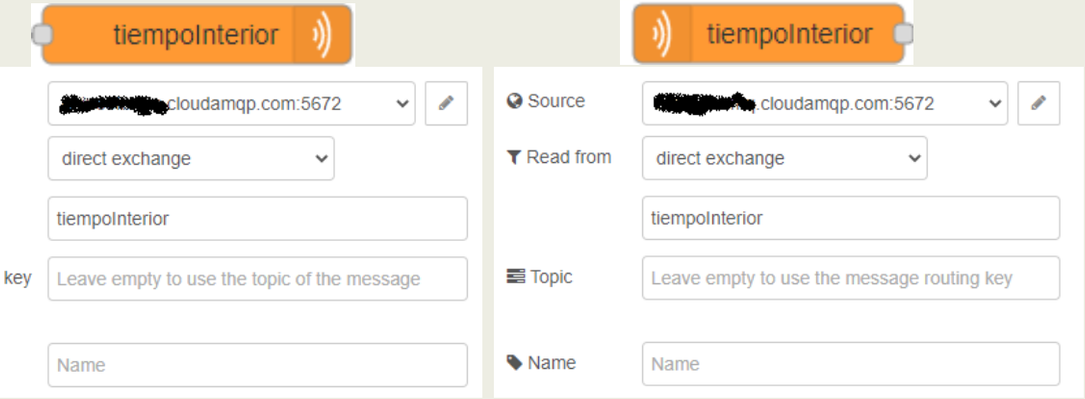
El ejemplo se basa en el envío y recepción de un JSON con info del sensor de temperatura y humedad modelo SI7021 de Sonoff para la placa TH16 (ver Capítulo 8. Domótica IoT).



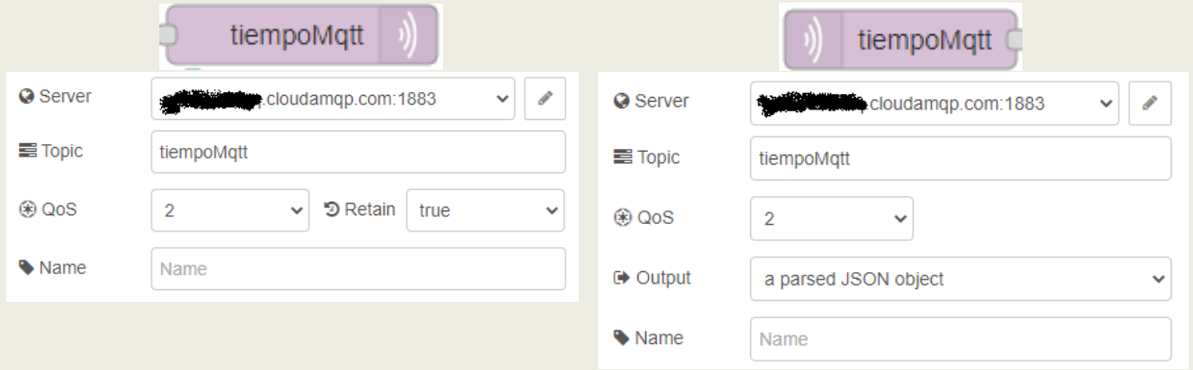
Mensaje (formato JSON) representado por Debug de Node-RED



AMQP → Puerto **5672** y Exchange: "*tiempoInterior*".



MQTT → Puerto **1883** y topic: "*tiempoMqtt*".



RabbitMQ

Broker

Conexiones: una para AMQP y otra para MQTT

Overview			Details			Network		+/-
Name	User name	State	SSL / TLS	Protocol	Channels	From client	To client	
[redacted]:57400	kdxibemi	running	○	AMQP 0-9-1	4	34iB/s	42iB/s	
[redacted]:57401	kdxibemi	running	○	MQTT 3.1.1	1	25iB/s	25iB/s	

Canales: Cada conexión abre 2 canales (publicar y consumir / confirmar). MQTT si es QoS > 0, sino sería uno.

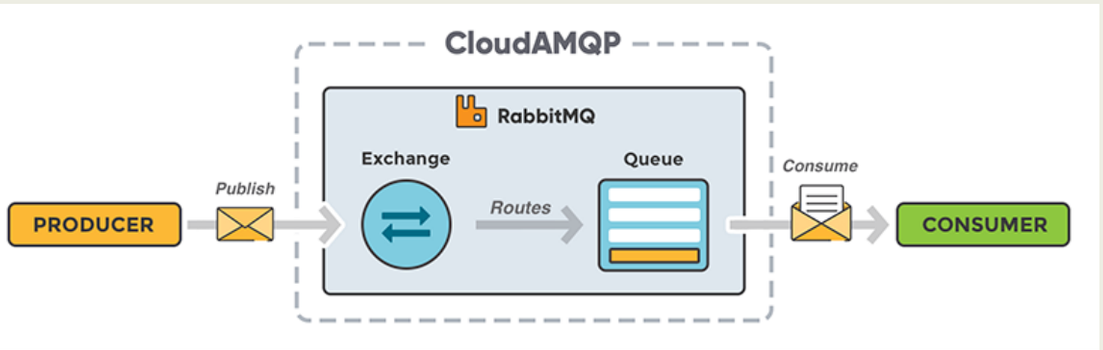
:57400 (3)	kdxibemi		running	AMQP
:57400 (4)	kdxibemi		running	
:57401 (1)	kdxibemi		running	MQTT
:57401 (2)	kdxibemi	C	running	

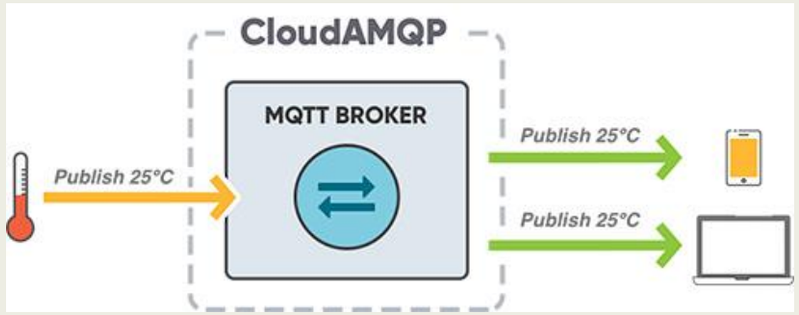
Exchange: sólo para AMQP. Hace binding a la cola tiempoInterior.

tiempoInterior	direct	D	0.20/s	0.20/s

Cola: Para AMQP es una cola. Para MQTT una suscripción (last message).

mqtt-subscription-mqtt_397dec95.232ac4qos1	classic	D AD HA	running
tiempoInterior	classic	D Args HA	running





Wireshark

Análisis de tramas

AMQP 0-9-1

Mensaje Publish (azul) y mensaje Deliver (rojo).

amqp

No.	Time	Source	Destination	Protocol	Length	Info
1179	9.971868			AMQP	224	Basic.Publish x=tiempoInterior rk= Content-Header Content-Body
1188	9.994313			AMQP	262	Basic.Deliver x=tiempoInterior rk= Content-Header Content-Body
2496	21.416541			AMQP	226	Basic.Publish x=tiempoInterior rk= Content-Header Content-Body
2503	21.438542			AMQP	264	Basic.Deliver x=tiempoInterior rk= Content-Header Content-Body
3390	31.558517			AMQP	226	Basic.Publish x=tiempoInterior rk= Content-Header Content-Body
3396	31.580521			AMQP	264	Basic.Deliver x=tiempoInterior rk= Content-Header Content-Body

```
Advanced Message Queueing Protocol
    Type: Method (1)
    Channel: 3
    Length: 23
    Class: Basic (60)
    Method: Publish (40)
  > Arguments
Advanced Message Queueing Protocol
    Type: Content header (2)
    Channel: 3
    Length: 18
    Class ID: Basic (60)
    Weight: 0
    Body size: 105
    Property flags: 0x2000
  v Properties
      Headers
Advanced Message Queueing Protocol
    Type: Content body (3)
    Channel: 3
    Length: 105
    Payload: 7b2254696d65223a22323032312d3
```

Publish: 146 bytes de datos AMQP

Deliver: 184 bytes de datos AMQP

```
Payload: 7b2254696d65223a22323032312d30312d31305432303a32353a3436222c225349373032…

0030  03 fc 22 f0 00 00 01 00  03 00 00 00 17 00 3c 00   ··"····· ······<·
0040  28 00 00 0e 74 69 65 6d  70 6f 49 6e 74 65 72 69   (···tiem poInteri
0050  6f 72 00 00 ce 02 00 03  00 00 00 12 00 3c 00 00   or····· ·····<··
0060  00 00 00 00 00 00 00 69  20 00 00 00 00 00 ce 03   ·······i ·· ·····
0070  00 03 00 00 00 69 7b 22  54 69 6d 65 22 3a 22 32   ·····i{" Time":"2
0080  30 32 31 2d 30 31 2d 31  30 54 32 30 3a 32 35 3a   021-01-1 0T20:25:
0090  34 36 22 2c 22 53 49 37  30 32 31 22 3a 7b 22 54   46","SI7 021":{"T
00a0  65 6d 70 65 72 61 74 75  72 65 22 3a 32 32 2e 33   emperatu re":22.3
00b0  2c 22 48 75 6d 69 64 69  74 79 22 3a 36 39 2c 22   ,"Humidi ty":69,"
00c0  44 65 77 50 6f 69 6e 74  22 3a 31 36 2e 33 7d 2c   DewPoint ":16.3},
00d0  22 54 65 6d 70 55 6e 69  74 22 3a 22 43 22 7d ce   "TempUni t":"C"}
```

MQTT 3.1.1 QoS = 2 (assured delivery)

Mensaje Publish (azul), Ack del broker al publicador y envío del mensaje del broker al suscriptor (rojo).

mqtt

No.	Time	Source	Destination	Protocol	Length	Info
1177	9.971315			MQTT	175	Publish Message (id=65102) [tiempoMqtt]
1189	9.997952			MQTT	60	Publish Ack (id=65102)
1190	9.998112			MQTT	175	Publish Message (id=65101) [tiempoMqtt]

Publish (pub→ broker): 119 bytes de datos MQTT

Publish (broker→ sub): 119 bytes de datos MQTT

```
MQ Telemetry Transport Protocol, Publish Message
  > [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]
  v Header Flags: 0x35, Message Type: Publish Message, QoS Level: Exactly once delivery (Assured Delivery), Retain
      0011 .... = Message Type: Publish Message (3)
      .... 0... = DUP Flag: Not set
      .... .10. = QoS Level: Exactly once delivery (Assured Delivery) (2)
      .... ...1 = Retain: Set
    Msg Len: 119
    Topic Length: 10
    Topic: tiempoMqtt
    Message Identifier: 65102
    Message: 7b2254696d65223a22323032312d30312d31305432303a32353a3436222c225349373032…
```

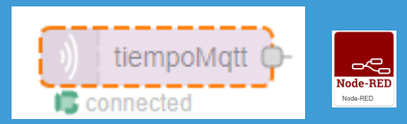
```
Message: 7b2254696d65223a22323032312d30312d31305432303a32353a3436222c225349373032…

0000  90 e2 ba 0a 45 c0 fa 16  3e 46 80 2e 08 00 45 00   ····E··· >F····E·
0010  00 a1 db f3 40 00 80 06  00 00 b9 fd 98 b6 34 2f   ····@··· ·····4/
0020  9b 48 e0 7b 07 5b 50 cf  9a 1a 2c 7d 3c e0 50 18   ·H·{·[P· ··,}<·P·
0030  03 fe 22 bf 00 00 35 77  00 0a 74 69 65 6d 70 6f   ··"···5w ··tiempo
0040  4d 71 74 74 fe 4e 7b 22  54 69 6d 65 22 3a 22 32   Mqtt·N{" Time":"2
0050  30 32 31 2d 30 31 2d 31  30 54 32 30 3a 32 35 3a   021-01-1 0T20:25:
0060  34 36 22 2c 22 53 49 37  30 32 31 22 3a 7b 22 54   46","SI7 021":{"T
0070  65 6d 70 65 72 61 74 75  72 65 22 3a 32 32 2e 33   emperatu re":22.3
0080  2c 22 48 75 6d 69 64 69  74 79 22 3a 36 39 2c 22   ,"Humidi ty":69,"
0090  44 65 77 50 6f 69 6e 74  22 3a 31 36 2e 33 7d 2c   DewPoint ":16.3},
00a0  22 54 65 6d 70 55 6e 69  74 22 3a 22 43 22 7d      "TempUni t":"C"}
```

Ejemplo Node-RED

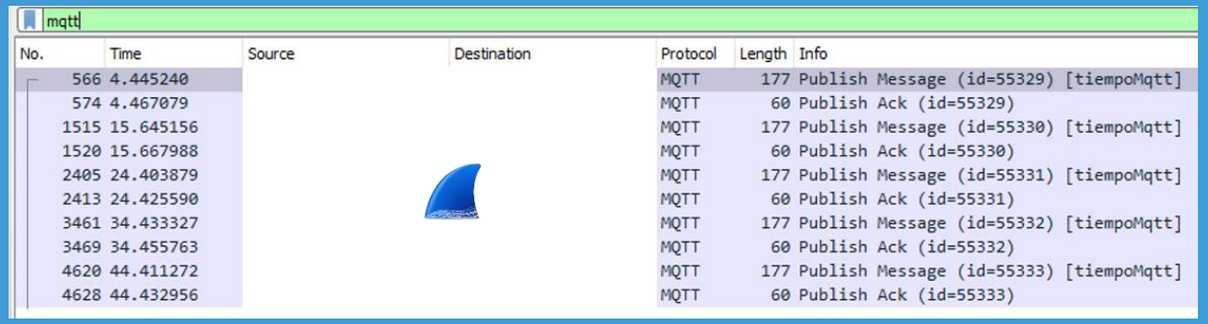
MQTT

¿Qué pasa si el receptor MQTT no está activo? Ponemos disabled el nodo mqtt in.



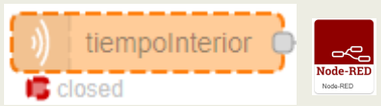
¿Qué ve el broker? Pierde el suscriptor MQTT y no encola nada.

Con wireshark se ve la publicación de datos MQTT y los ACK del broker al publicador pero no el envío del mensaje a los suscriptores (no hay).

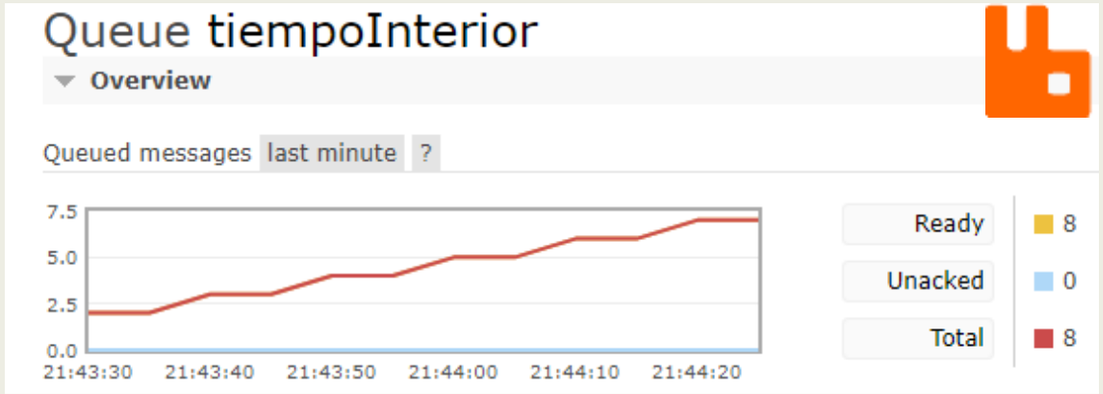


AMQP

¿Qué pasa si el receptor AMQP no está activo? Ponemos disabled el nodo amqp in.



¿Qué hace el broker? Encola los mensajes publicados ya que no son consumidos.

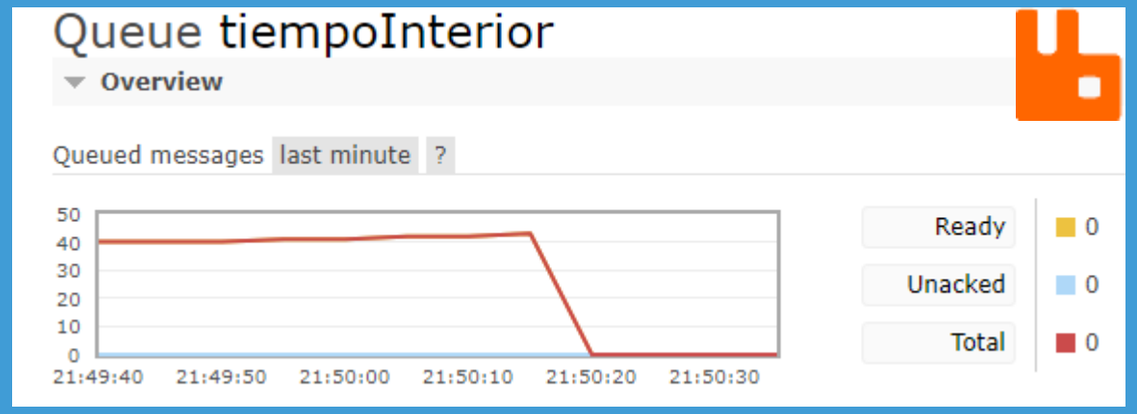


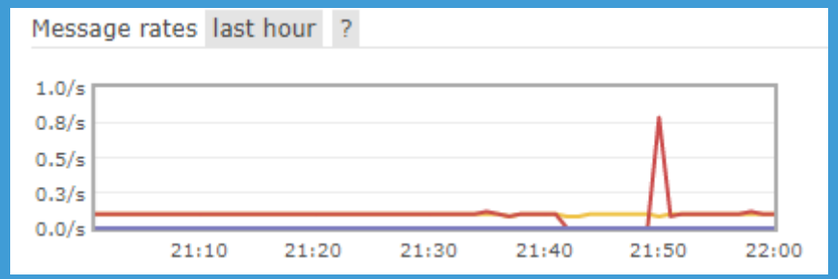
Ejemplo Node-RED

AMQP

¿Qué pasa si el receptor AMQP vuelve a estar activo?

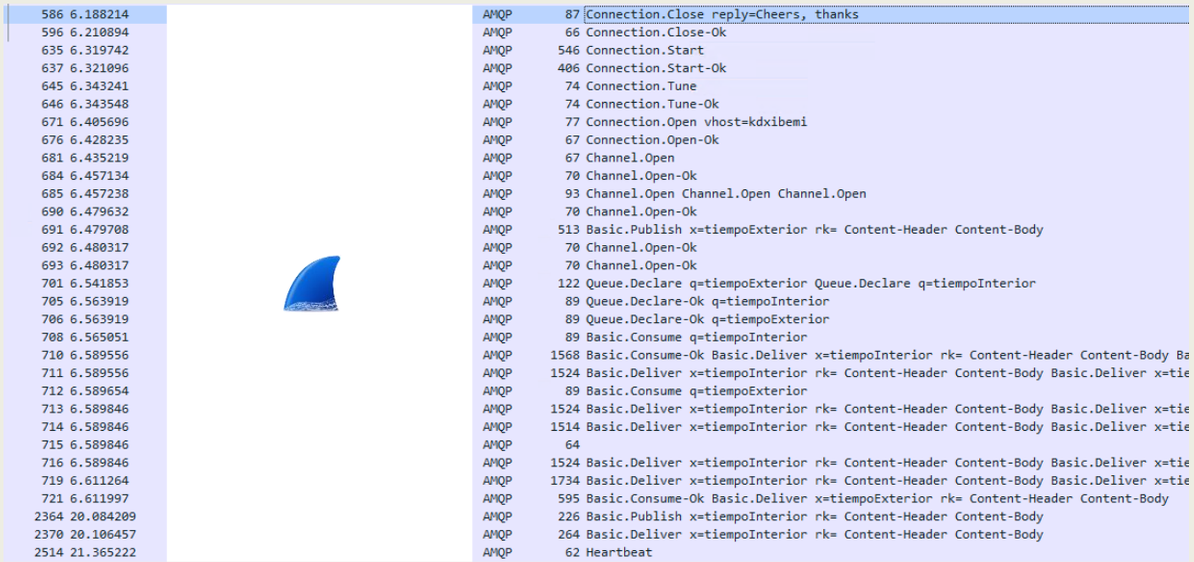
Vacía la cola consumiendo los mensajes acumulados (40 mensajes en este caso).





AMQP

¿Qué se ve en la red?



A partir del mensaje 711. Manda 6 mensajes de Basic.Deliver (tiempoInterior) de más tamaño con el contenido de la cola . En cada trama Ethernet manda 6-8 mensajes para enviar los 40 mensajes encolados.

IoT Messaging Protocols

Resumen

- El protocolo MQTT es algo más ligero y aunque puede garantizar que el mensaje se entregue, no realiza ningún encolado. Protocolo pub/sub para Machine to Machine (M2M).
- Si queremos historizar datos (historian – influxdb) tras recibirlos por MQTT podríamos perder algunos datos. Si el canal y/o el receptor fallan esos datos no serán historizados.
- ¿Solución? Almacenarlos en una cola previa con AMQP. De esta manera todo lo publicado pasa por la cola y el receptor, cuando el canal esté disponible o cuando disponga de recursos irá tratando esos mensajes (a su ritmo - asincronía).
- Imaginemos pedidos online y no queremos perder ni uno porque no puedo procesarlos. La solución será añadirlos en colas.
- Ejemplo de MQTT (Facebook Messenger) y de AMQP (Google / NASA).
- Existen otros protocolos como STOMP, CoAP, XMPP, DDS. Aquí dejo algún enlace para poder investigar más sobre ellos.

https://geeks.ms/jorge/2017/07/11/messaging-con-amqp-mqtt-y-stomp/

https://arxiv.org/pdf/1804.01747.pdf

Patrones arquitectónicos que cumplen estos protocolos:

MQTT	AMQP
Publish-Subscriber (pub/sub) / broker	Publish-Subscriber (pub/sub) / broker
-	Store&Forward (queues)

MQTT.ORG

AMQP Advanced Message Queuing Protocol

RabbitMQ

Node-RED

Node-RED

GPS TRACKING

GPS tracking

- Solución para control de flotas o para registrar tus rutas a pie, en coche, moto y/o bicicleta.
- Fuente de datos app OwnTracks en un Smartphone y GPS activado.
- Envío de datos (posicionamiento GPS) a un broker MQTT (RabbitMQ en cloud). Puede ser cualquiera (Mosquitto, Mosca, Aedes, Hbmqtt, Emqtt, Rabbitmq, Hivemq CE, Activemq, Ubidots, etc.) en cloud o en local la mayoría de ellos y open source.
- Recepción y registro de datos en un server con Node-RED e InfluxDB. Puede ser también una Raspberry Pi 4 o cualquier equipo con Node-RED y una base de datos (influxdb, mysql, sql server, mongodb, etc.)
- Acceso web mediante navegador para visualización en un mapa de la posición actual y del track realizado.

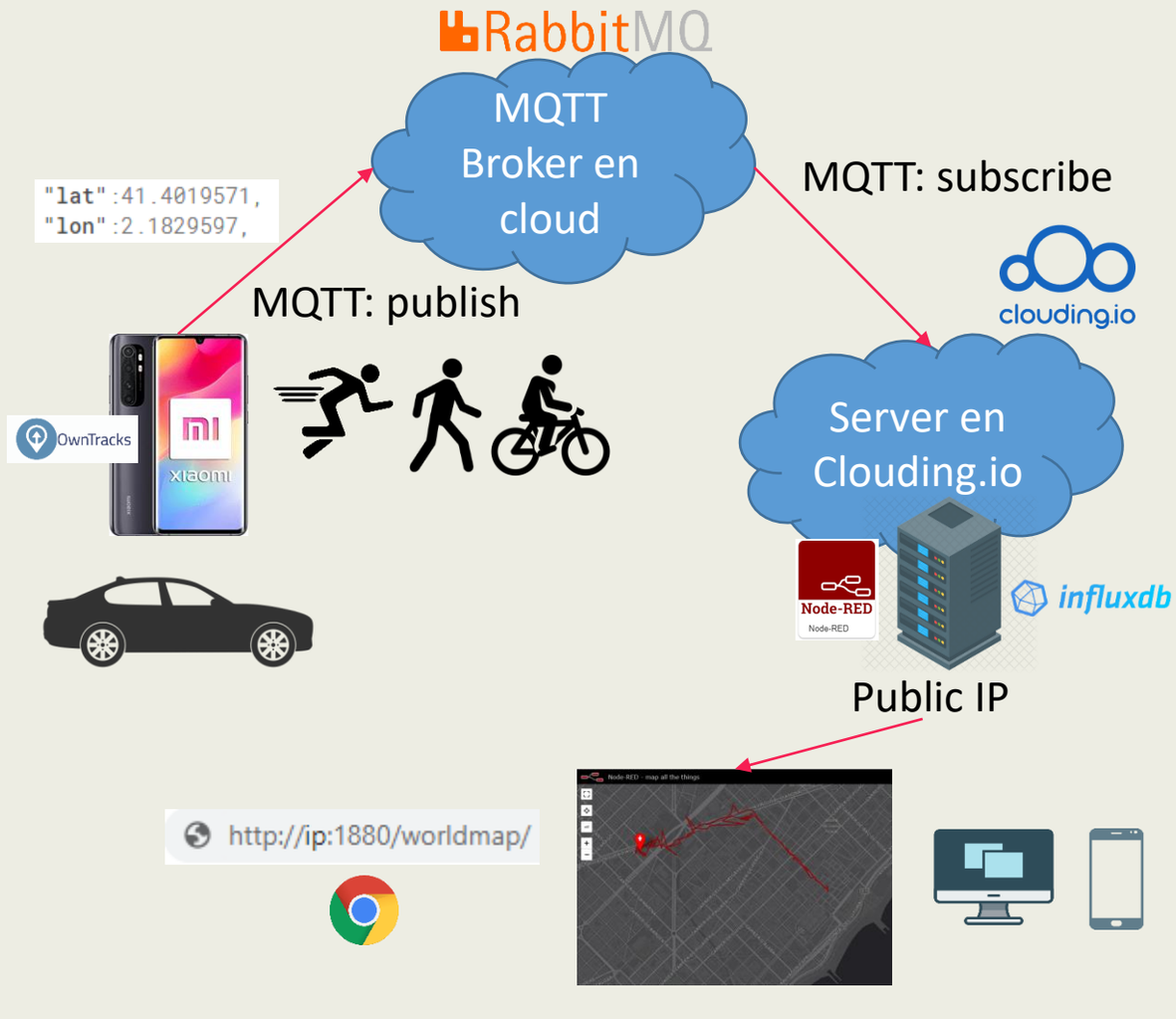
Links con recursos útiles:

https://flows.nodered.org/node/node-red-contrib-web-worldmap

https://discourse.nodered.org/t/worldmap-gps-tracking/13895/13

http://resources-area.co.uk/node-red-flows/tracking_a_mobile_device.pdf

https://dev.to/webhookrelay/fast--simple-location-tracking-with-node-red-and-owntracks-195o



GPS TRACKING

Fuente de datos

Como fuente de datos utilizaremos la app "Owntracks" para smartphone Android. https://owntracks.org/booklet/

Se debe activar el GPS (Global Positioning System)

Se puede configurar el envío de datos mediante HTTP o MQTT. En este caso está como MQTT (1883).

El topic será "**owntracks/kdxibemi:kdxibemi/xiaomimi10**"

Puede funcionar en modo:

- -Silencio
- -Modo movimiento
- -Modo manual
- -Modo cambio significante

Broker MQTT

El broker MQTT será RabbitMQ (cloudamqp)

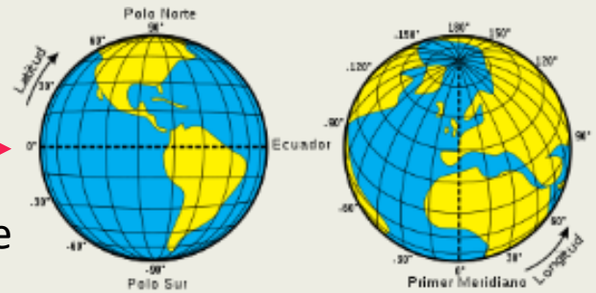
Los datos se publican en formato JSON


```
{
   "_type":"location",
   "acc":15,
   "alt":65,
   "batt":83,
   "bs":1,
   "conn":"w",
   "created_at":1610887485,
   "inregions":[
      "casa"
   ],
   "lat":41.4019571,
   "lon":2.1829597,
   "t":"u",
   "tid":"10",
   "tst":1610887453,
   "vac":4,
   "vel":0
}
```


JSON

Battery

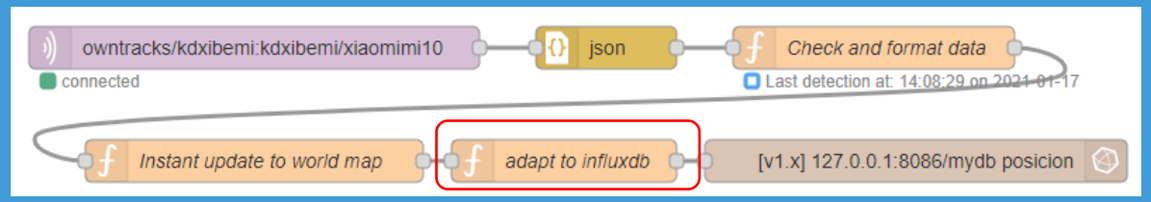
ID device



GPS TRACKING

Save data

Nos suscribiremos al broker MQTT con Node-RED y registraremos cada localización recibida en InfluxDB.



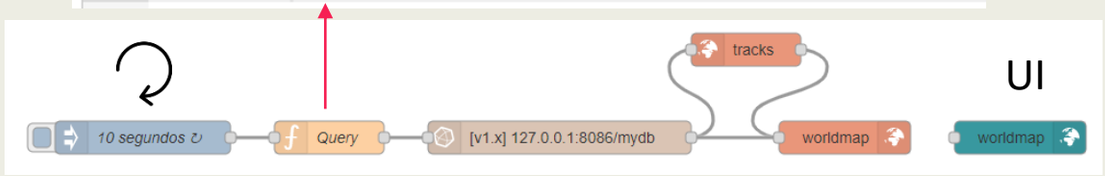

```
msg.payload = { lat: msg.lat,
                lon: msg.lon,
                name: "Ricardo",
                layer: "realtime",
                date: msg.date,
                device: msg.device_id,
                icon: "male",
                iconColor: "red",
                batt: msg.batt
              };
return msg;
```

```
> select * from posicion
name: posicion
time                           batt date       device icon iconColor lat       layer    lon       name
----                           ---- ----       ------ ---- --------- ---       -----    ---       ----
2021-01-17T09:11:14.1739054Z 92   2021-01-17 10     male red       41.4019268 realtime 2.1829688 Ricardo
2021-01-17T09:11:15.7473855Z 92   2021-01-17 10     male red       41.4019081 realtime 2.1828964 Ricardo
2021-01-17T09:11:53.6336674Z 92   2021-01-17 10     male red       41.4018582 realtime 2.1828682 Ricardo
2021-01-17T09:12:20.0999664Z 92   2021-01-17 10     male red       41.4017444 realtime 2.1830066 Ricardo
2021-01-17T09:12:50.1290631Z 92   2021-01-17 10     male red       41.4017105 realtime 2.1831949 Ricardo
2021-01-17T09:13:20.1266336Z 92   2021-01-17 10     male red       41.4016996 realtime 2.1832086 Ricardo
2021-01-17T09:13:59.767719Z  92   2021-01-17 10     male red       41.4018661 realtime 2.1828266 Ricardo
2021-01-17T09:14:19.1256374Z 92   2021-01-17 10     male red       41.4016682 realtime 2.183377  Ricardo
2021-01-17T09:14:51.135139Z  92   2021-01-17 10     male red       41.4016674 realtime 2.1833795 Ricardo
2021-01-17T09:15:21.0943314Z 91   2021-01-17 10     male red       41.4016676 realtime 2.1834159 Ricardo
```

Get data

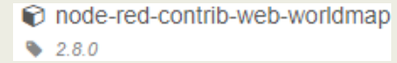
Cada 10 segundos haremos la siguiente query en InfluxDB (datos del último día).

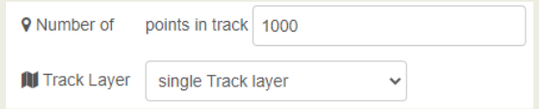
```
var selectQuery = "select * from posicion where time > now() - 1d";
msg.query = selectQuery;
return msg;
```

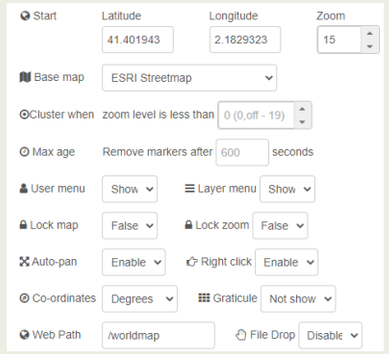


Obtendremos las N posiciones para representar en mapa un track y la última ubicación.

La configuración de tracks y worldmap



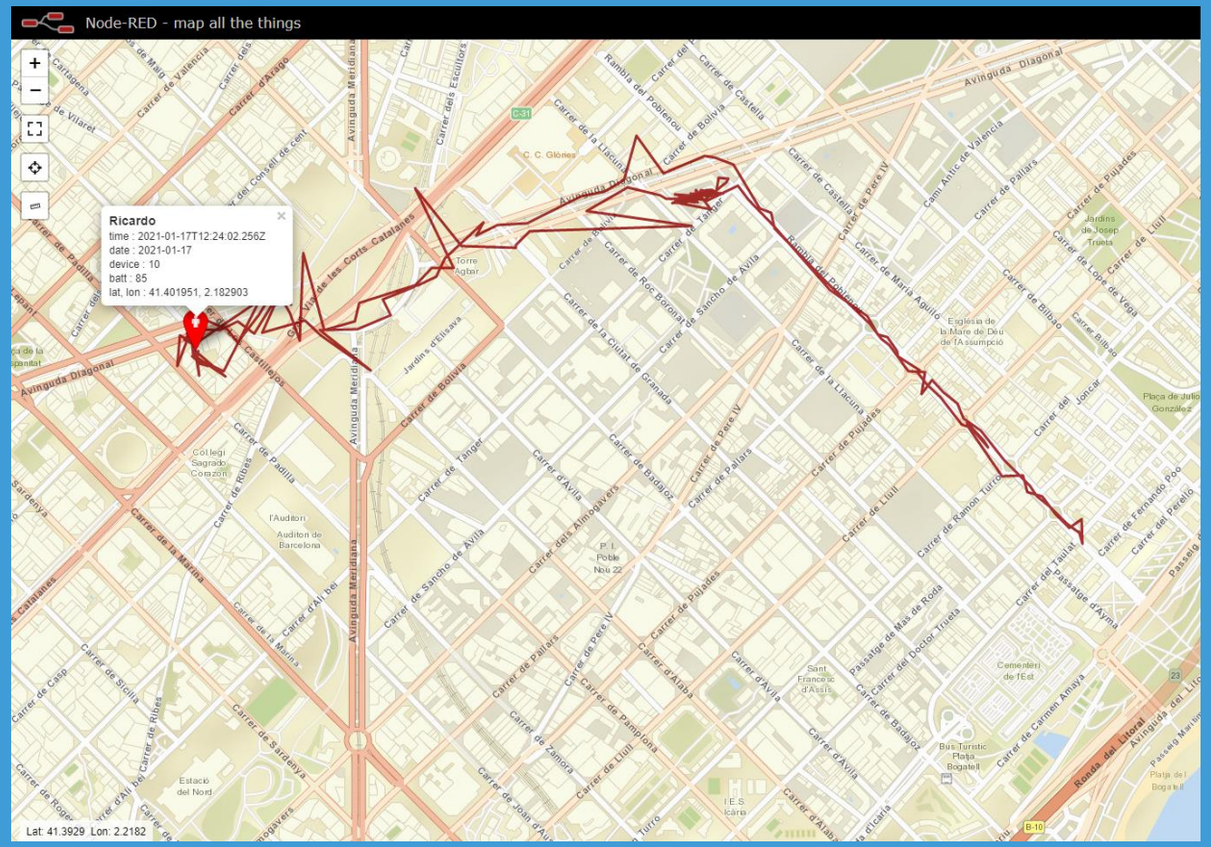




GPS TRACKING

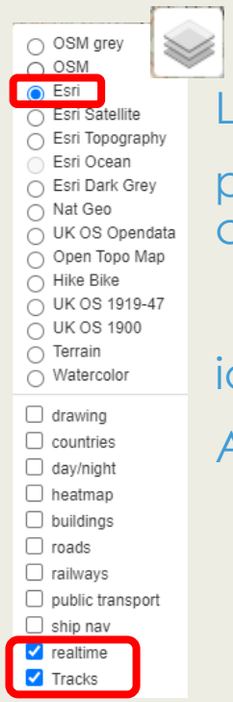
Representación en mapa

Node-RED map all the things



Ubicación y track

Trazamos en tiempo real el camino realizado y la posición actual con los datos registrados en la BBDD.



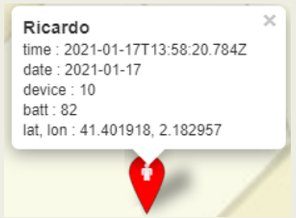
http://ip:1880/worldmap/

Los mapas servidos por streetmap de Esri

puedo seleccionar las capas (layers) creadas: “realtime” y “Tracks”.

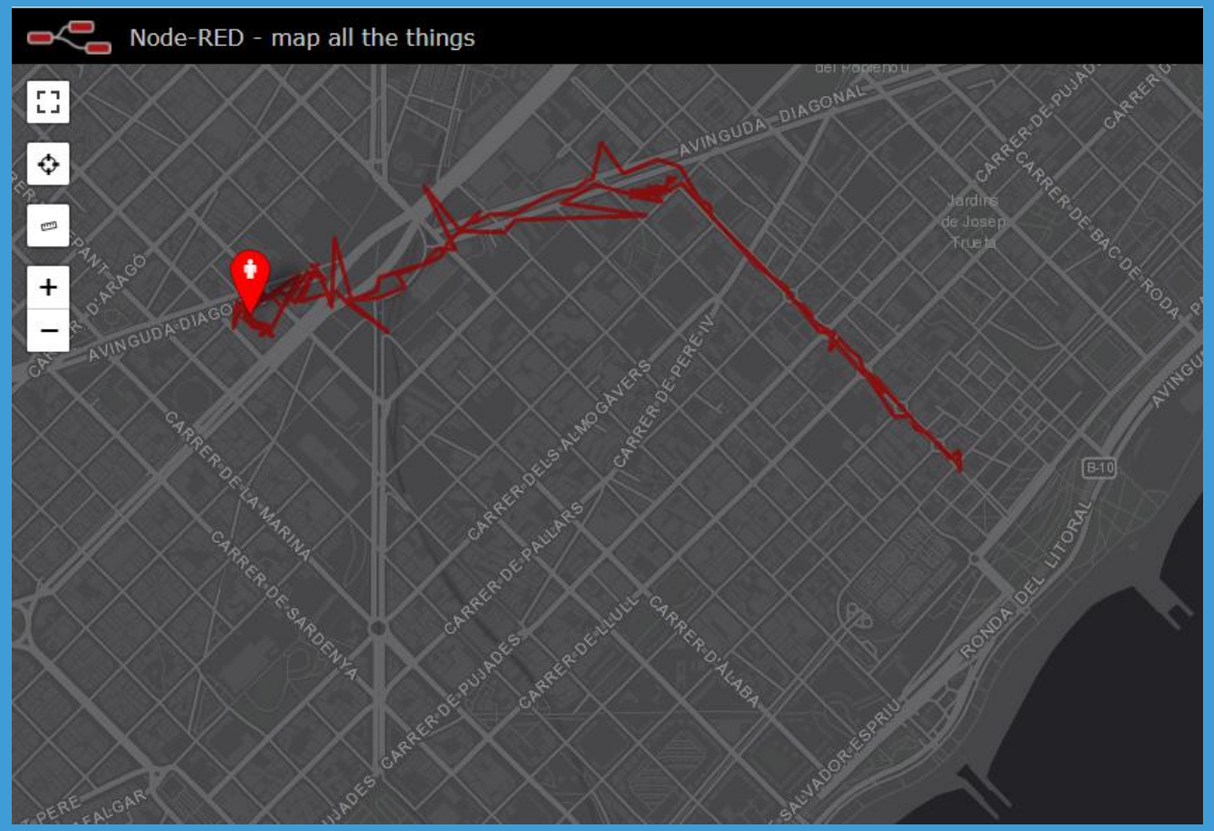
icon: male / iconColor: red

Al clicar en él aparece un tooltip con info



GPS TRACKING

Esri Dark Grey



Esri + heat map layer

IoT Messaging Protocols
v.1.1 MARZO 2024

https://www.linkedin.com/in/ricardo-moraleda-gareta-9421099

https://www.linkedin.com/company/gdo-electric1996/

RICARDO MORALEDA GARETA

GDO SOFTWARE