

Router Titan

Firmware V6

Sending data to
ThingsBoard platform

•Detalles del escenario

Los router Titan disponen de todas las funcionalidades típicas de un router 4G/3G/2G pero además cuentan con una serie de prestaciones adicionales que lo convierten en uno de los routers con más prestaciones del mercado.

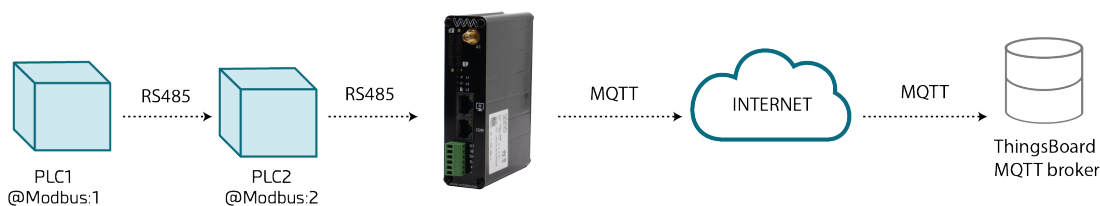
Una de las prestaciones adicionales es la capacidad de datalogger donde el router Titan almacena en su memoria no volátil distintos tipos de registros en un formato JSON. Estos registros almacenados pueden provenir de lecturas MODBUS, o de capturas de datos SERIE por sus puertos RS232 / RS485, de posiciones GPS, etc. Estos registros de tipo JSON son almacenados por el router Titan en su memoria interna no volátil para ser posteriormente enviados a una plataforma remota mediante comunicaciones tipo HTTP, HTTPS, MQTT, MQTTS, FTP, FTPS

Como se indica, el router Titan almacena en su memoria interna dichos registros JSON en un formato predeterminado y propietario. En ocasiones ésto puede ser un problema para la comunicación con aquellas plataformas que esperan recibir la información en un determinado formato (es decir, con un formato distinto de JSON al utilizado por defecto por el router Titan).

Mediante la funcionalidad “JSON Transformer Function Script” el router Titan permite realizar un formateo de cualquier objeto JSON antes de su almacenamiento en su memoria interna. Esta característica permite por tanto acomodar los objetos JSON con el formato adecuado para cada plataforma. Realmente no es necesario transformar los datos en un formato JSON, puede transformarse en cualquier formato String.

En esta nota de aplicación se desarrollará el ejemplo completo para el envío de datos a la conocida plataforma ThingsBoard (<https://www.thingsboard.cloud/>) la cual requiere de un formato especial de los JSON enviados.

En esta nota de aplicación en concreto vamos a suponer que se deben leer 2 registros de 2 PLCs con comunicaciones modbus conectados a un Webdyn-EasyTunnel por su puerto RS485



En concreto se pretende lo siguiente: cada 1 minuto, el dispositivo WebDyn-Easytunnel debe leer los registros modbus con dirección 40000 y 40001 del PLC1 y los registros 40000 y 40001 del PLC2. En ambos PLCs el registro 40000 corresponde con la temperatura medida y el registro 40001 con la humedad. Las lecturas realizadas deben almacenarse en la memoria interna no volátil del WebDyn-EasyTunnel (en su datalogger) y éste debe enviar dichos datos leídos a la plataforma ThingsBoard siempre que sea posible (que haya cobertura, conectividad IP, ...) La comunicación con los PLCs es a través de un bus RS485 con una configuración 9600,8,N,1

1. Configuración de la Mobile WAN.

El Webdyn EasyTunnel debe comunicarse con la plataforma ThingsBoard a través de comunicaciones 4G/3G/2G, por lo que la sección “Mobile-Basic Settings” debe configurarse apropiadamente en función de la tarjeta SIM utilizada.

2. Configuración del puerto serie RS485

La comunicación con los dos PLCs se realizará a través de puerto serie RS485 a 9600,8,N,1, por lo que debe configurarse la sección “Serial Settings - Serial Port2-RS485” ajustando los parámetros como se indica a continuación.

3. Configuración del Logger

El siguiente paso es configurar el datalogger interno del dispositivo. Para ello debe acudir al menú "External Devices - Logger configuration". La configuración debería ser similar a la que se muestra en la siguiente figura:

External Devices > Logger

ID: Optional. Device identification

Send mode: Send mode (normally FIFO)

Time format: Time format used in timestamp logger data

Use script: ☒ Check for customized json using 'Json Transformer Script' in [Script section](#). Check if you want to send more than one JSON per transimition.

Use array: ☐ Save data in Logger only if date has been set (check [Time Servers](#))

Check date: ☐

Communication mode: WEB PLATFORM (HTTP REST)

Enabled: ☐ Communication mode HTTP enabled

Mode: Method of sending data

Custom parameters: Optional. Ex: &a=1&b=2 only for "HTTP GET/PUT (PARAMETERS)" modes

En la parte inferior de la misma pantalla debe seleccionarse el modo de comunicación MQTT y en el topic de envío de telemetría debe introducirse: v1/devices/me/telemetry

External Devices > Logger

Enabled: ☐ Communication mode HTTP enabled

Mode: Method of sending data

Custom parameters: Optional. Ex: &a=1&b=2 only for "HTTP GET/PUT (PARAMETERS)" modes

Custom header1: Optional. Custom header1. For example: Content-type:application/json

Custom header2: Optional. Custom header2. For example: IDENTITY_KEY:YOUR_KEY

Custom header3: Optional. Custom header3.

Server: Destination URL. Example: www.mydomain.com/set.asp?data=

Server Username: Optional. Blank if no server authentication required

Server Password: Optional. Blank if no server authentication required

Communication mode: MQTT

Enabled: ☒ Communication mode MQTT enabled

MQTT Topic: MQTT Topic. Example: [IMEI]/logger

Note: Other>MQTT menu must be configured

Cabe destacar los siguientes parámetros:

- **“Time format”** seleccionamos formato de hora “unix”.
- **“Use script”** es muy importante seleccionar esta casilla, pues al activarla se ejecutará el script que incorporaremos más adelante y que transformará el JSON estándar creado por el dispositivo con los datos en un JSON con el formato necesario soportado por la plataforma ThingsBoard.
- **“MQTT Enabled”** El logger debe utilizar el método de envío por MQTT hacia la plataforma ThingsBoard, por lo que debe seleccionarse esta casilla.
- **“MQTT Topic”**. Indica el topic de envío de telemetrías “v1/devices/me/telemetry” que es el que necesita la plataforma ThingsBoard

4. Configuración de la sección Modbus

En esta sección de configuración “External Devices - Modbus Devices” debe configurarse las lecturas Modbus que debe realizar el dispositivo Titan sobre los 2 PLCs

External Devices ▶ ModBus RTU / TCP

Enabled: ☒ Enable Modbus Devices

Serial Port: Serial Port 2 Select the connected serial port if needed

Logger: ☒ Check if logger must be used
Please, configure logger before using this option

SAVE CONFIG VIEW LOG

Dev. name / ID	Addr.	Command	Start @	Num word/bit	Reg Type	Period		
PLC1	1	0x04	40000	2	WORD	1	Del	Test
PLC2	2	0x04	40000	2	WORD	1	Del	Test

Device name / ID: PLC2 Insert the device name or ID

Address: 2 Modbus RTU address or IP:port address

Command: 0x04 Modbus read command

Start: 40000 Address of the first register

Number Words / Bits: 2 Words for command 0x03/0x04. Bits for 0x01/0x02

Reg Type: WORD Type of registers for command 0x03/0x04

Period: 1 Read period (minutes)

Como puede observarse en la pantalla anterior, se debe habilitar el servicio Modbus activando la casilla “Enabled”. Debe seleccionarse el puerto serie “Serial Port 2”, pues la lectura se realizará a través del puerto RS485. Y también debe activarse la casilla “Logger”, pues los registros modbus leídos de los PLCs deben almacenarse en el datalogger interno del dispositivo Titan.

También deben crearse dos nuevos dispositivos que en este ejemplo se llaman PLC1 y PLC2. En “address” debe indicarse la dirección modbus de cada uno de los PLCs. PLC1 tendrá la dirección modbus 1 y PLC2 la dirección modbus 2. El comando modbus que se utilizará en este ejemplo para la lectura de los registros 40000 y 40001 será el 0x04, por lo que deberá seleccionar el comando 0x04 en el campo “Command”. El registro inicial a leer en ambos PLCs es el registro 40000, por lo que en el campo “Start” se indicará el valor 40000. Pretendemos leer 2 registros modbus de cada PLC (el 40000 y 40001) ,

lo que se traduce en que en el campo “Number Words” tendrá un valor “2”. El tipo de registro “Reg Type” debe seleccionarse tipo “WORD” y en “Period” seleccionaremos 1, pues queremos lecturas de los registros modbus cada 1 minuto.

5. Configuración del SCRIPT para adaptar el JSON con formato estándar generado por el dispositivo Titan al formato que queremos utilizar para comunicarnos con la plataforma ThingsBoard.

El formato de JSON estándar generado por el dispositivo Titan con los datos leídos y con el que de forma estándar (sin transformar) enviaría a una plataforma tendría un formato como el que sigue:

Para el PLC1:

```
{“IMEI”:”869101054286683”,”TYPE”:”MODB”,”TS”:”2022-11-11T12:17:00Z”,”ID”:”PLC1”,”A”:”1”,”ST”:”40000”,”N”:”2”,”V”:[225,62],”P”:”ID0001”}
```

Para el PLC2:

```
{“IMEI”:”869101054286683”,”TYPE”:”MODB”,”TS”:”2022-11-11T12:17:01Z”,”ID”:”PLC2”,”A”:”2”,”ST”:”40000”,”N”:”2”,”V”:[241,71],”P”:”ID0001”}
```

Pero por sencillez, queremos enviar los datos a la plataforma ThingsBoard así:

Para el PLC1:

```
{“TS1”:”2022-11-11T12:17:00Z”, “TEMP1”:22.5, “HUM1”:62}
```

Para el PLC2:

```
{“TS2”:”2022-11-11T12:17:00Z”, “TEMP2”:24.1, “HUM2”:71}
```

Es decir, se pretende asignar a los campos del JSON “TEMP1” y “TEMP2” la temperatura de cada PLC (dividiendo además cada valor por 10 para obtener la temperatura en grados). Los campos “HUM1” y “HUM2” tendrán asignados los valores de humedad obtenidos de la lectura Modbus.

Por lo tanto hay que crear un pequeño SCRIPT para adaptar los datos de la forma estándar a la transformada para el envío a ThingsBoard. Recuerde que este script de transformación se ejecutará porque anteriormente, en la sección “External Devices - Logger Configuration”, seleccionamos la casilla “Use script”

Use script: ☒

Use array: ☐

Check date: ☐

Communication mode: **WEB PLATFORM (HTTP REST)**

Enabled: ☐ Communication mode HTTP enabled

Mode: **HTTP GET (JSON)** Method of sending data

Check for customized json using 'Json Transformer Script' in **Script** section.
Check if you want to send more than one JSON per transmission.
Save data in Logger only if date has been set (check Time Servers)

Para introducir el código del script de transformación de datos debe acudir al menú “Other - Titan Scripts”. El código del script de transformación debe introducirse en la sección “JSON Transformer Function Script”

Other Titan Scripts v2 JSON Transformer Function Script

This function script allows to customize json sent by Logger and private DNS (File: 'script-transf.mtx')

```
function getTransformedJson (json)
{
    var objJson = JSON.parse(json);
    if (objJson.TYPE=='MODB')
    {
        var objThingsBoardJson=JSON.parse("{}");
        if (objJson.ID=="PLC1")
        {
            objThingsBoardJson.TS1=objJson.TS;
            objThingsBoardJson.TEMP1=objJson.V[0]/10;
            objThingsBoardJson.HUM1=objJson.V[1];
        }
        else if (objJson.ID=="PLC2")
        {
            objThingsBoardJson.TS2=objJson.TS;
            objThingsBoardJson.TEMP2=objJson.V[0]/10;
            objThingsBoardJson.HUM2=objJson.V[1];
        }
        objJson=objThingsBoardJson;
    }
    return JSON.stringify(objJson);
}
```

Save JSON Transformer Script Delete JSON Script Load Example Encrypt Script

El código del SCRIPT es el siguiente.

//Creamos una variable objeto de tipo JSON, pues la variable “json” que recibe la

//función como argumento es de tipo “String”.

```
var objJson = JSON.parse(json);
```

//Si en el json original recibido, se indica que es un JSON de datos modbus ...

```
if (objJson.TYPE=='MODB')
```

```
{
```

```
    //Creamos un nuevo objeto JSON, donde pondremos los datos
```

```
    //transformados
```

```

var objThingsBoardJson=JSON.parse("{}");

//Si los datos corresponden a lecturas del PLC1 ...
if (objJson.ID=="PLC1")
{
    //Añadimos la fecha de los datos.
    objThingsBoardJson.TS1=objJson.TS;

    //Añadimos la temperatura en el campo TEMP1 dividido por 10.
    objThingsBoardJson.TEMP1=objJson.V[0]/10;

    //Añadimos la humedad en el campo HUM1.
    objThingsBoardJson.HUM1=objJson.V[1];
}
//Si los datos corresponden a lecturas del PLC2 ...
else if (objJson.ID=="PLC2")
{
    //Añadimos la fecha de los datos.
    objThingsBoardJson.TS2=objJson.TS;

    //Añadimos la temperatura en el campo TEMP2 dividido por 10.
    objThingsBoardJson.TEMP2=objJson.V[0]/10;

    //Añadimos la humedad en el campo HUM2.
    objThingsBoardJson.HUM2=objJson.V[1];
}

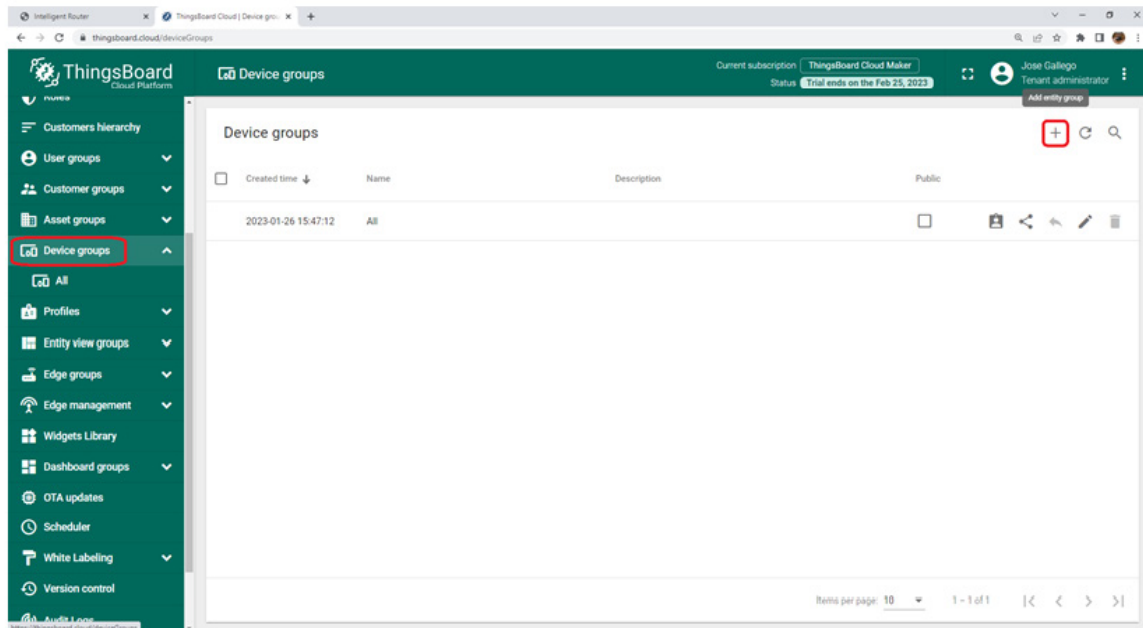
//Finalmente copiamos el JSON transformado, para devolverlo como resultado
//de la función de transformación.
objJson=objThingsBoardJson;
}

```

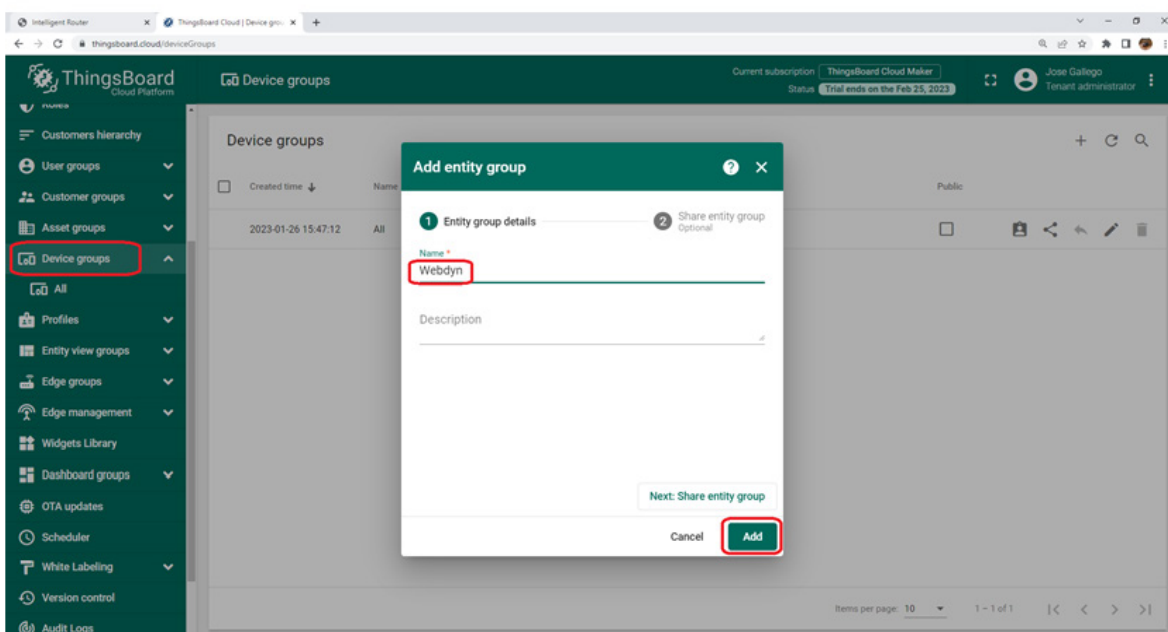
6. Configuración de la sección MQTT y plataforma ThingsBoard

Finalmente debemos configurar correctamente la sección “Other - Mqtt” del dispositivo Titan para que pueda conectarse a la plataforma ThingsBoard. Pero antes debemos acudir a la plataforma web para añadir el dispositivo Webdyn-Easy-tunnel y obtener los atributos de autenticación mqtt.

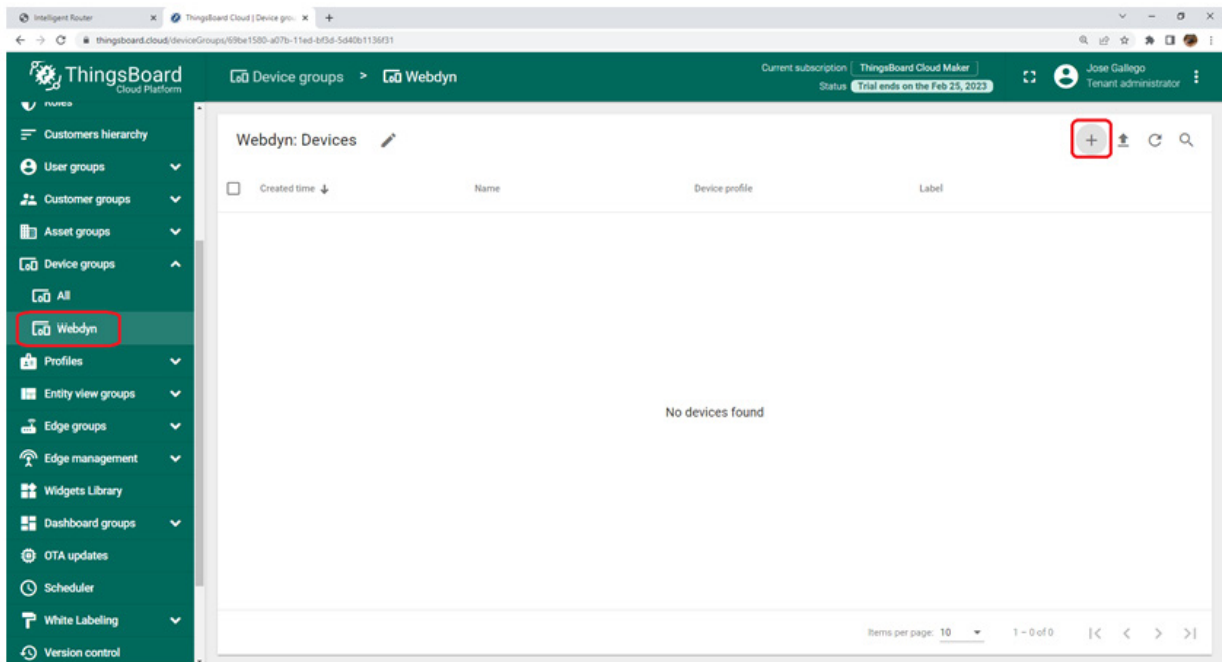
Ya en la cuenta de <http://thingsboard.cloud> debe acudirse a la sección “Device groups”. Allí debe pulsarse el icono “+” para crear un nuevo grupo.



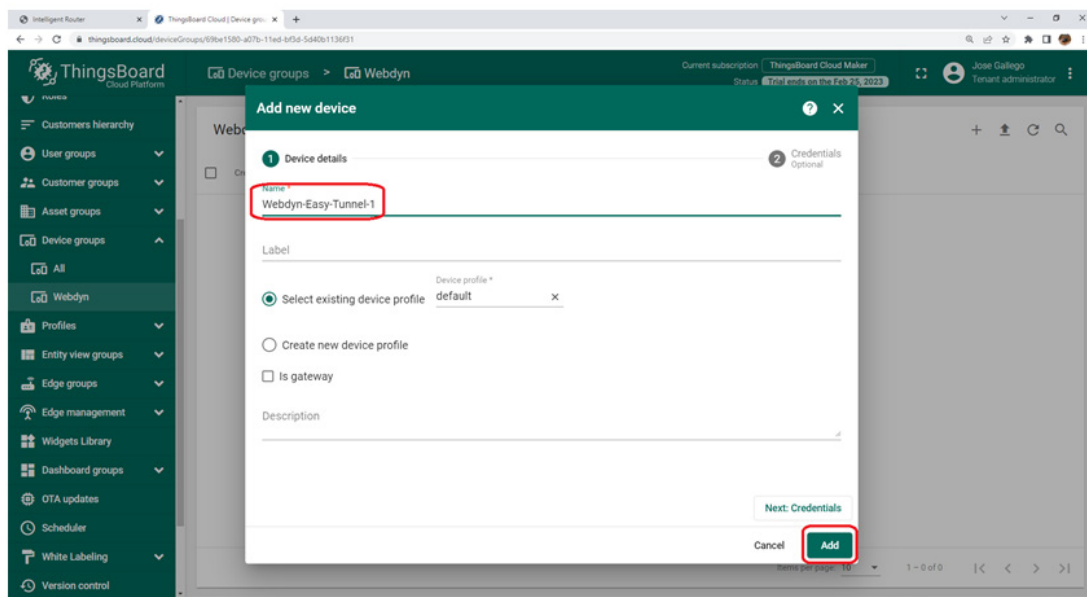
Por ejemplo, puede introducirse el nombre de grupo “Webdyn”. Tras ello debe pulsarse el botón “Add”.



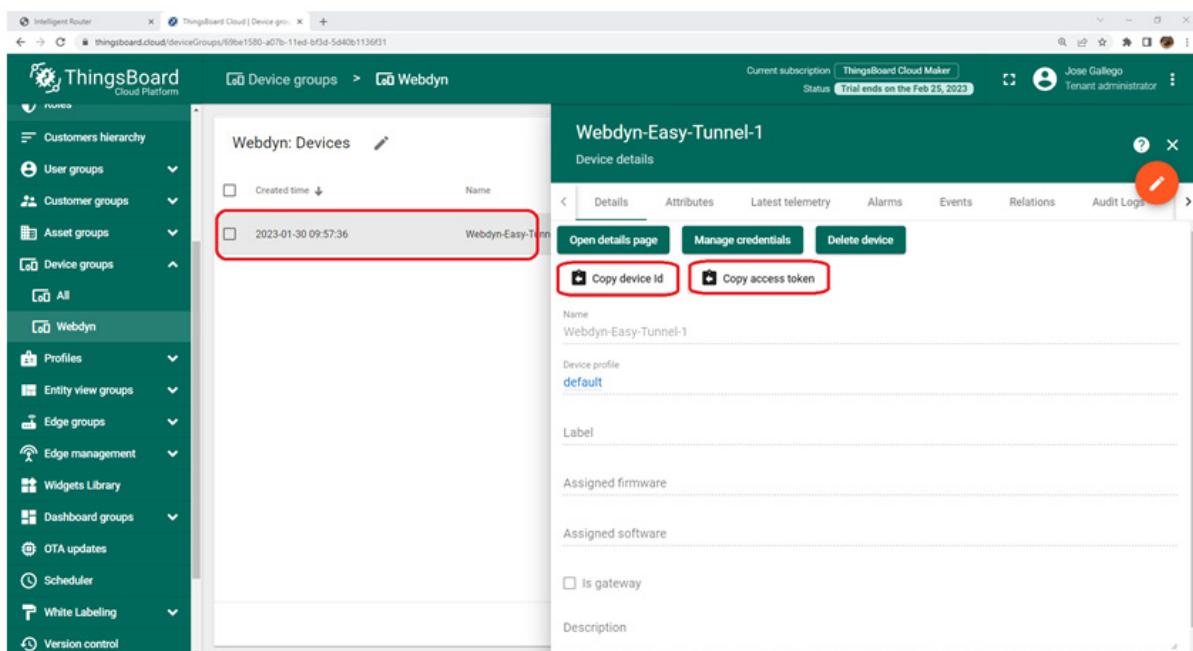
Una vez creado el grupo “Webdyn” debe entrarse en el grupo creado (haciendo click en la línea del grupo) y tras ello debe crearse un nuevo dispositivo pulsando de nuevo en el icono “+”.



Como nombre de dispositivo puede introducirse “Webdyn-Easy-Tunnel-1”, y tras ello debe pulsarse el botón “Add”.



Seguidamente debe hacerse click en la línea del nuevo dispositivo creado y pulsarse en el botón “Copy device Id” y “Copy access token”. Esos valores deben apuntarse, pues se necesitarán incorporar en la sección “Other - MQTT” del dispositivo Titan.



Ahora ya en la sección “Other-Mqtt” del dispositivo Titan debe activarse el servicio “MQTT Client” para que éste pueda conectarse con la plataforma ThingsBoard. En el campo “MQTT Broker” se introducirá el valor “tcp://thingsboard.cloud:1883”, en el campo “Mqtt Username” el “access token” y en el campo “MQTT ID” el “device ID”, ambos campos copiados en el paso anterior. Seguidamente debe pulsarse el botón “SAVE Config” y reiniciar el dispositivo Titan para que aplique la nueva configuración.

- ★ **Mobile**
 - Status
 - Basic Settings
 - Keep Online
- ★ **Ethernet**
 - Basic Settings
- ★ **Firewall**
 - Authorized IPs
- ★ **Serial Settings**
 - Serial Port1-RS232
 - Serial Port2-RS485
 - SSL Certificates
- ★ **External Devices**
 - Logger configuration
 - ModBus Devices
 - Generic Serial Device
 - Temperature Sensor
 - IEC102 Meter
 - GPS Receiver
- ★ **Other**
 - AT Command
 - DynDns
 - Private DynDns
 - Sms control
 - Periodic Autoreset
 - Time Servers
 - Remote Console
 - Snmp
 - Tacacs+
 - Mqtt
 - Http / Https
 - User Permissions

▶ Other ▶ MQTT Broker

Enabled:	☐	Enable MQTT broker service
TCP Port:	1883	Listening port (for example 1883)
Anonymous User:	☐	Allow anonymous user (no user / password is needed)
User:		Username (used if anonymous is not selected)
Password:		Password (used if anonymous is not selected)

▶ Other ▶ MQTT Client

Enabled:	☒	Enable MQTT client
MQTT Broker	tcp://thingsboard.cloud:1883	Destination MQTT Broker. Examples: tcp://test.mosquitto.org:1883 ssl://test.mosquitto.org:8883 (certificate needed) ssl://test.mosquitto.org:8884 (certificates needed)
MQTT Username	2X5PYTCx80KYNo1ftqwP	MQTT Username (blank if not used)
MQTT Password		MQTT Password (blank if not used)
MQTT ID	2474d9e0-a07c-11ed-a8d5-e	Device identification
MQTT Qos	1	MQTT Quality Of Service (0 ... 2)
MQTT Keepalive	60	Seconds for keepalive (30 ... 3600)
MQTT Persistence	☐	Data persistence

Tras el reinicio y pasados unos segundos para que el dispositivo Titan consiga conectividad IP, si todo es correcto en la sección “Other - MQTT” aparecerá el estado de la conexión mqtt como sigue:

- MQTT
- Http / Https
- User Permissions
- Passwords Web UI
- CA Certificates
- Email Config
- ModBus Slave
- Titan Scripts
- Connectivity tools
- Digital I/O
- Custom Skin
- Led Config
- Syslog
- Backup / Factory
- Firmware Upgrade
- Reboot
- Logout

MQTT AT Topic		This topic will be subscribed for receiving AT Commands (usefull for individual device)
MQTT AT Resp Topic		This topic will be used for publishing the AT Command Responses of AT Topic
MQTT AT Topic 2		This topic will be subscribed for receiving AT Commands (usefull for groups)
MQTT AT Resp Topic 2		This topic will be used for publishing the AT Command Responses of AT Topic 2
MQTT AT Topic 3		This topic will be subscribed for receiving AT Commands (usefull for all devices)
MQTT AT Resp Topic 3		This topic will be used for publishing the AT Command Responses of AT Topic 3

MQTT Script Topic 1	v1/devices/me/rpc/request+	When data is received in this topic the 'Topic Script' will be executed.
MQTT Script Topic 2		When data is received in this topic the 'Topic Script' will be executed.

SAVE CONFIG

Other ▶ MQTT Client ▶ Status

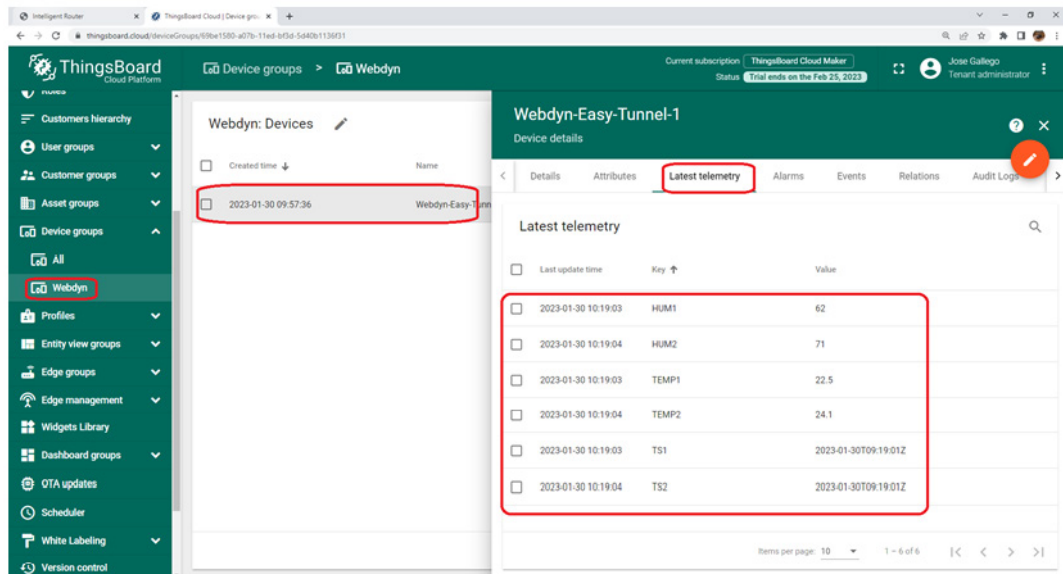
Internet status: Online

MQTT connection status: Connected

REFRESH

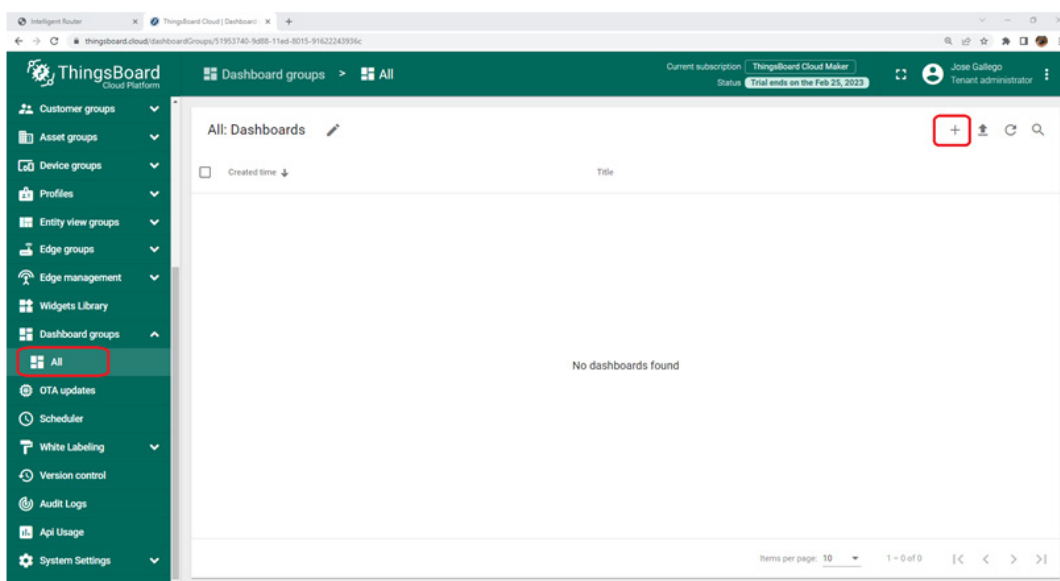
Checked every MQTT
Keepalive period

Ya en este punto queda por ver si la plataforma ThingsBoard está recibiendo los datos de los registros modbus correspondientes a la temperatura y humedad obtenidos leídos de ambos PLCs vía Modbus. Para ello, de nuevo haciendo click en la línea del dispositivo, debe hacerse click en la sección “Latest telemetry”. En esa sección de la plataforma ThingsBoard, si se están recibiendo correctamente los datos enviados por el dispositivo Titan, deberán aparecer como se muestra en la siguiente captura de pantalla:

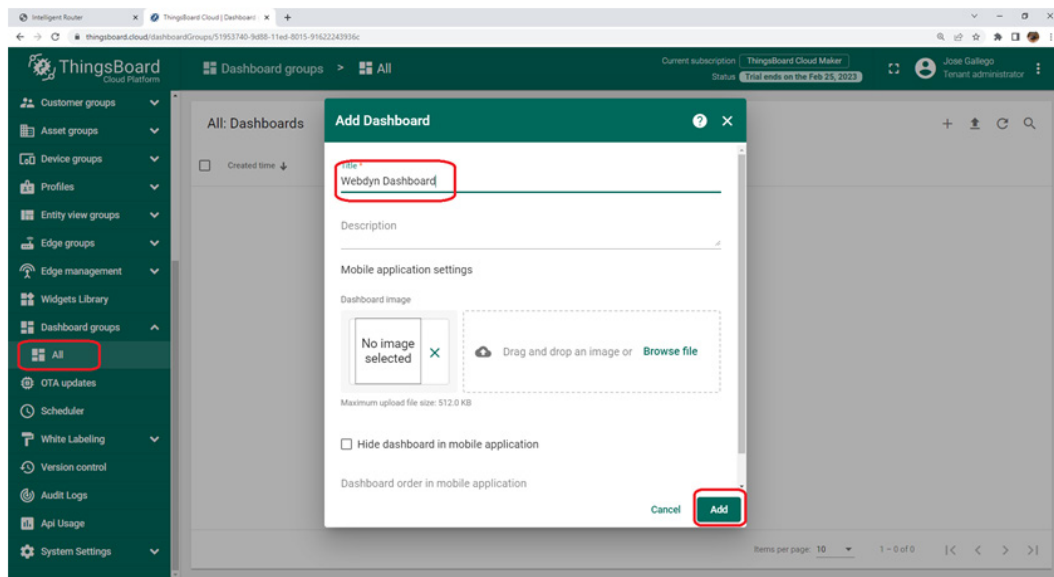


7.Construyendo un Dashbpard en ThingsBoard para visualizar los datos de temperatura y humedad.

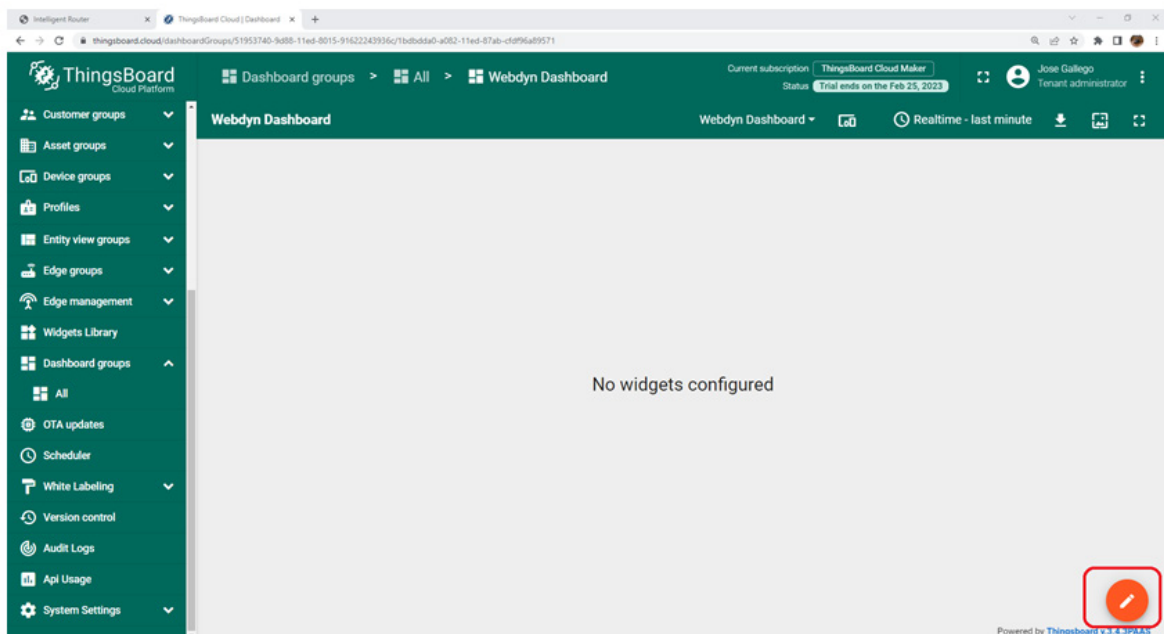
Para crear un dashboard donde representar gráficamente los datos, es posible hacer click en la sección “DashBoard groups - All” y después click en el icono “+” para crear un nuevo Dashboard.



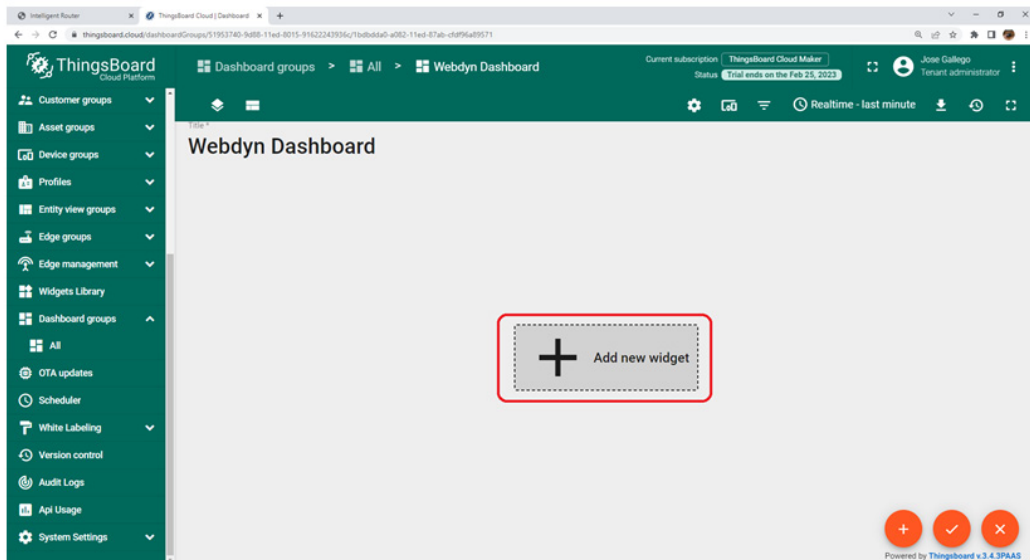
Como nombre del dashboard puede introducirse “Webdyn Dashboard” y tras ello pulsar el botón “Add



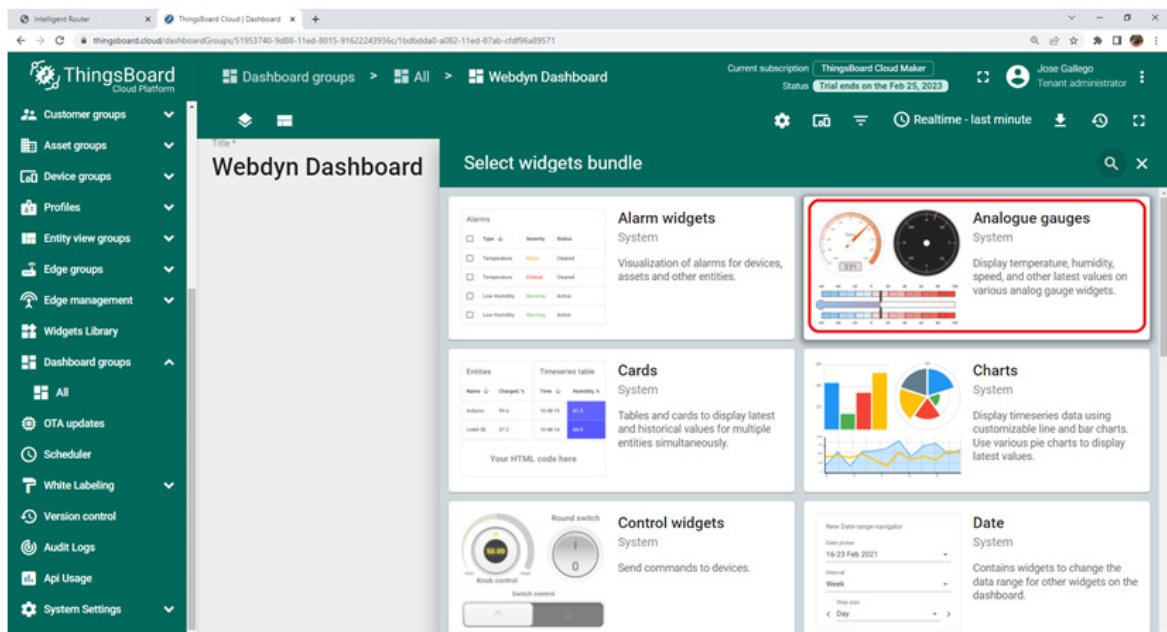
Seguidamente debe pulsarse sobre la línea del nuevo dashboard creado y hacer click en el icono de edición situado abajo a la derecha.



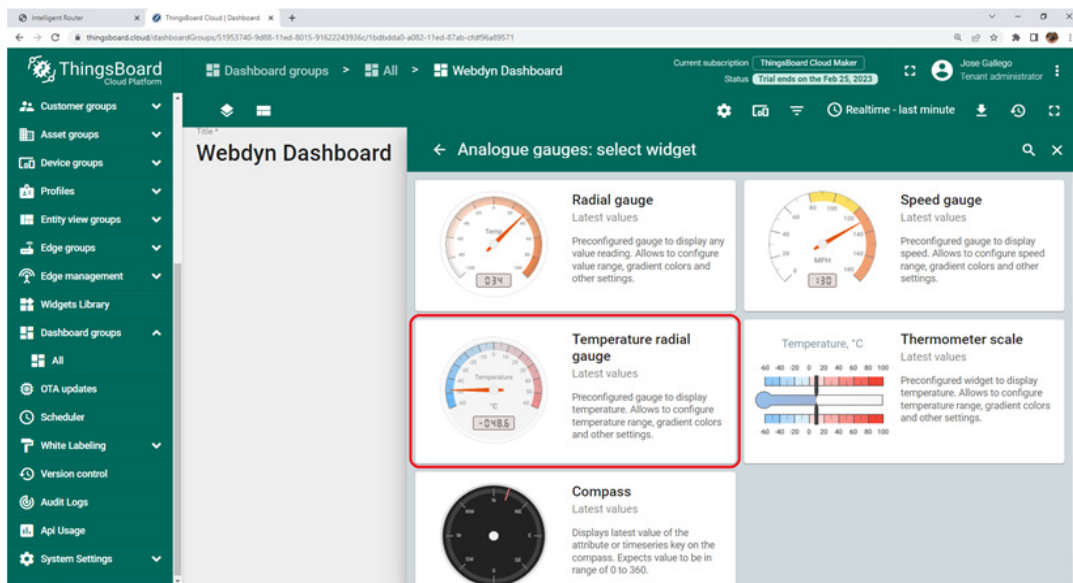
Pulsar de nuevo en el botón “Add new widget” para añadir un nuevo widget al dashboard



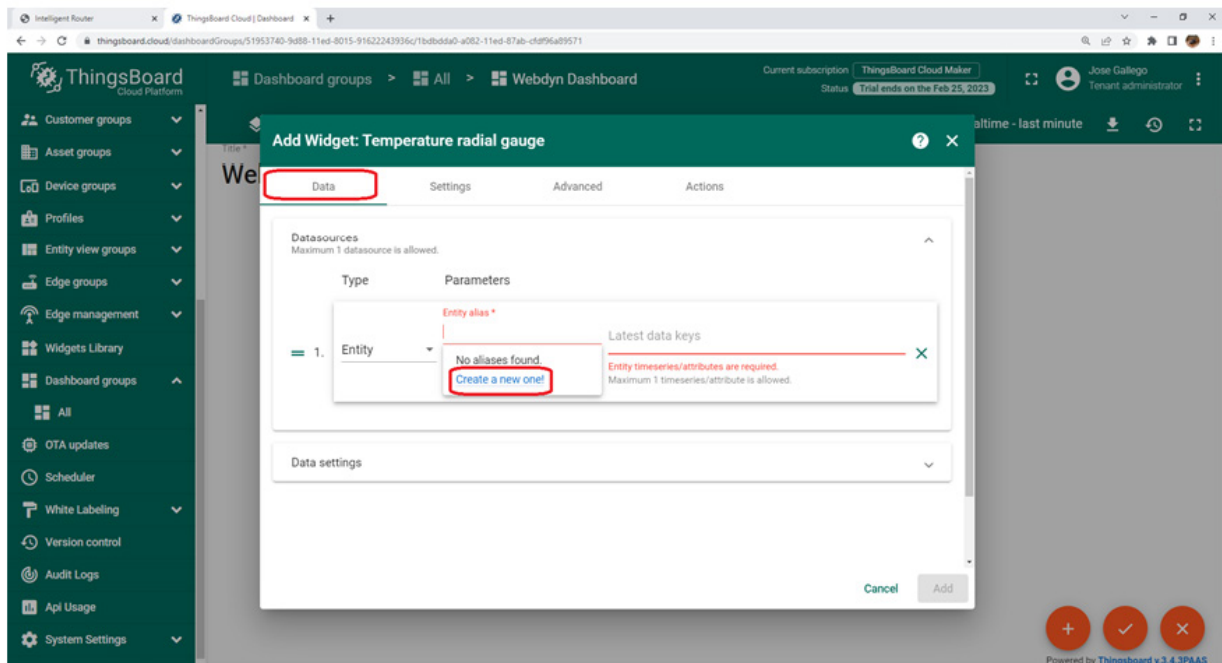
Seleccionaremos el tipo de gauge “Analogue gauges”



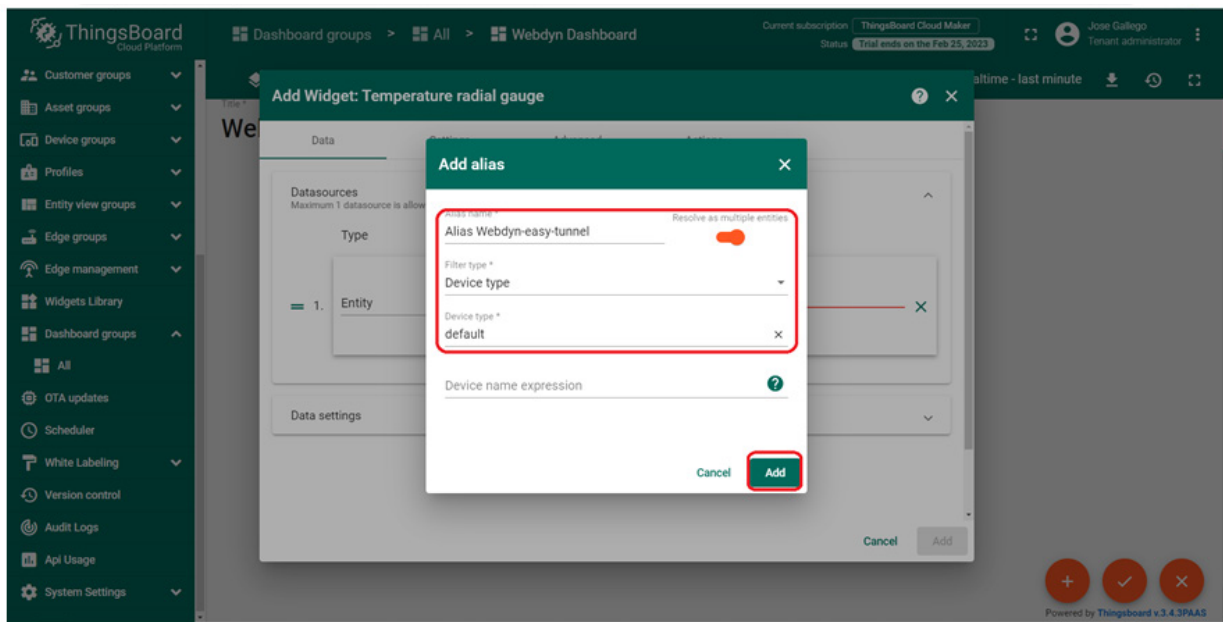
Y Finalmente se añadirá un “Temperature Radial gauge”



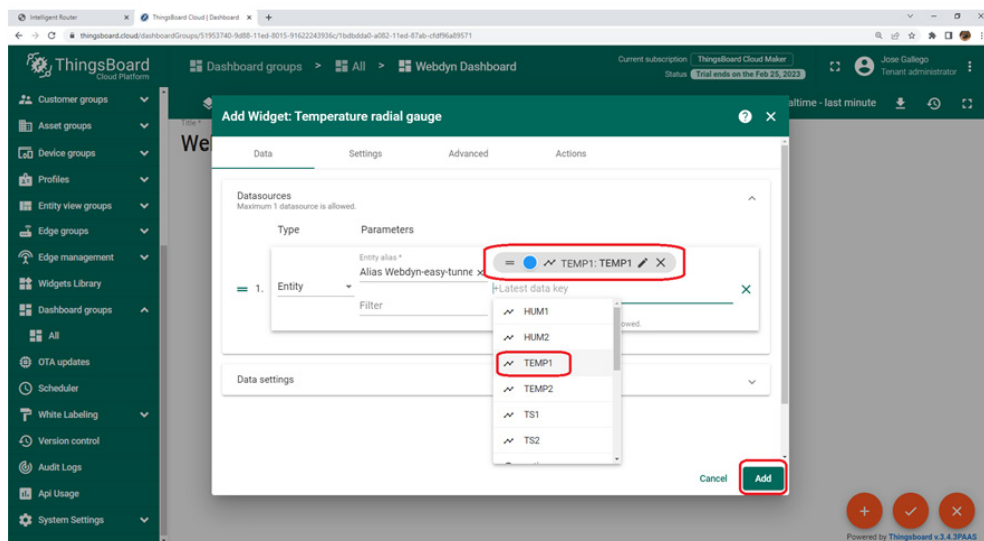
Una vez añadido, en las propiedades del “Temperature radial gauge”, en la sección “Data”, debe pulsarse en el link “Create a new one” para crear un nuevo Alias que haga referencia al dispositivo Titan Webdyn-Easy-Tunnel



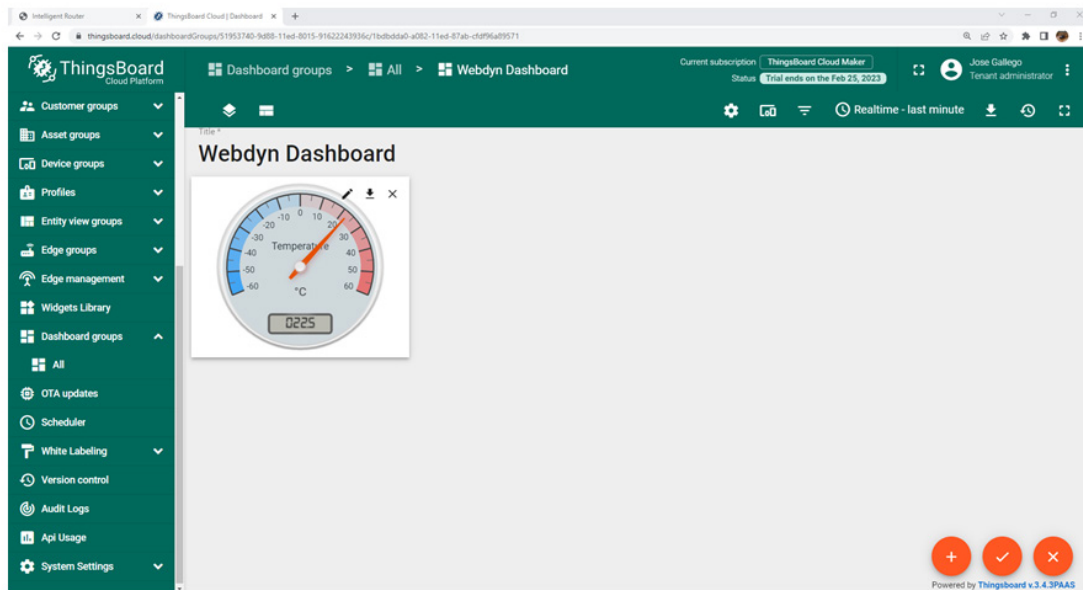
Creamos el Alias al dispositivo y pulsamos el botón “Add”.



Tras crear el Alias, en el desplegable de al lado, debe seleccionarse el campo “TEMP1”. Los campos del desplegable se mostrarán automáticamente si la conexión con el dispositivo Titan está establecida y el módem ya ha enviado datos. Después debemos pulsar el botón “Add” para finalizar.



En este punto el widget se añadirá en el dashboard y debería mostrar el valor actual de la temperatura leída por el PLC1, en el caso del ejemplo 22.5°C



Para añadir otros widgets con la temperatura del PLC2 y las humedades del PLC1 y PLC2 basta con repetir los últimos pasos, seleccionando para cada uno de ellos los campos correspondientes TEMP2, HUM1, HUM2.

8. Comunicación directa entre la plataforma ThingsBoard y el dispositivo Titan Webdyn-Easy-Tunnel para lectura de estados y cambio de configuración remota..

Lo primero que debe realizarse para poder enviar comandos desde la plataforma ThingsBoard al dispositivo Titan es añadir los topics MQTT adecuados. Ello debe hacerse desde la sección “Other ☐ Mqtt” añadiendo el topic “v1/devices/me/rpc/request/+” en el campo “MQTT Script Topic 1”. Esto hará que cualquier dato recibido por el dispositivo Titan en el topic “v1/devices/me/rpc/request/xxxxx” será reenviado internamente al script de “MQTT Topic Function Script”. Tenga en cuenta que esta funcionalidad (la de poder añadir el caracter “+” en el topic para subscribirse en cualquier topic que empiece por v1/devices/me/rpc/request/xxxxx donde xxxxx puede ser cualquier cosa) es soportada a partir de la versión de firmware 6.17. Tras el cambio de configuración, debe reiniciarse el dispositivo Titan.

Esto es necesario porque la plataforma ThingsBoard enviará los comandos al dispositivo Titan al topic “v1/devices/me/rpc/request/xxxxx” donde xxxxx será un número que irá incrementando en cada comando enviado. El dispositivo Titan deberá enviar la respuesta al topic “v1/devices/me/rpc/response/xxxxx”

MQTT Persistence ☐ Data persistence ☒

MQTT AT Topic This topic will be subscribed for receiving AT Commands (usefull for individual device)

MQTT AT Resp Topic This topic will be used for publishing the AT Command Responses of AT Topic

MQTT AT Topic 2 This topic will be subscribed for receiving AT Commands (usefull for groups)

MQTT AT Resp Topic 2 This topic will be used for publishing the AT Command Responses of AT Topic 2

MQTT AT Topic 3 This topic will be subscribed for receiving AT Commands (usefull for all devices)

MQTT AT Resp Topic 3 This topic will be used for publishing the AT Command Responses of AT Topic 3

MQTT Script Topic 1 When data is received in this topic the 'Topic Script' will be executed.

MQTT Script Topic 2 When data is received in this topic the 'Topic Script' will be executed.

Ahora añadimos el siguiente script en la sección “Other - MQTT”:

Other > Titan Scripts v2 > MQTT Topic Function Script

This function script is executed when data is received in [Mqtt Script Topics](#) (File: 'script-mqtt.mtx')

```
function topicScript (topic, stringData)
{
    var objJson = JSON.parse(stringData);

    function replaceSpecialChars(data)
    {
        data=data.replace(/(\r\n|\n|\r)/gm, " ");
        return data;
    }

    mtx.println("topic:" + topic);
    mtx.println("stringData:" + stringData);

    //Si el JSON es de tipo MODB (Modbus) construiremos el nuevo JSON de resultado
    if (objJson.method=='command')
    {
        // ...
    }
}
```

A continuación se muestra el código del script completo y explicado.

```

//Creamos un objeto JSON con los datos recibidos de ThingsBoard
var objJson = JSON.parse(stringData);

//Función que usaremos para substituir caracteres especiales como ENTER, ... como espacios en blanco.
function replaceSpecialChars(data)
{
    data=data.replace(/(\r\n|\n|\r)/gm, " ");
    return data;
}

//Imprimimos por la salida de debug el topic y datos recibido por la plataforma
mtx.println("topic:" + topic);
mtx.println("stringData:" + stringData);

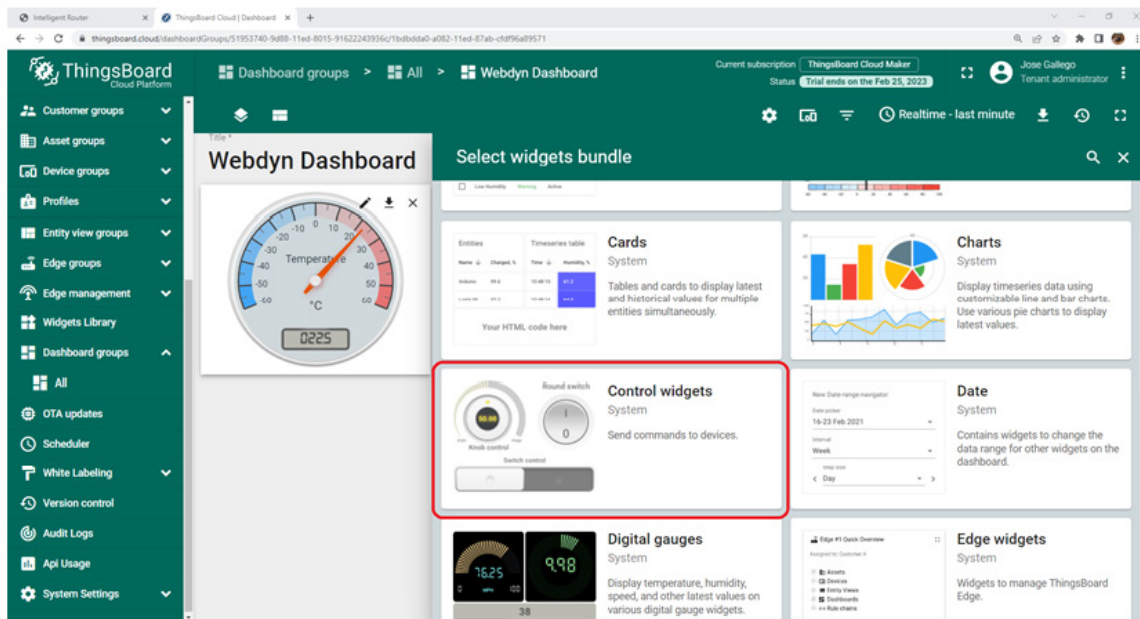
//Si en el campo "method" del json recibido de la plataforma tenemos la identificación "command"
if (objJson.method=='command')
{
    //Si hay un comando para ejecutar en el campo "params" del JSON recibido ...
    if (objJson.params!=null)
    {
        //Ejecutamos el comando AT
        var res=mtx.atSend(objJson.params,2000);

        //Creamos el topic de respuesta, substituyendo el texto "request" por "response"
        var topicResponse=topic.replace("request","response");

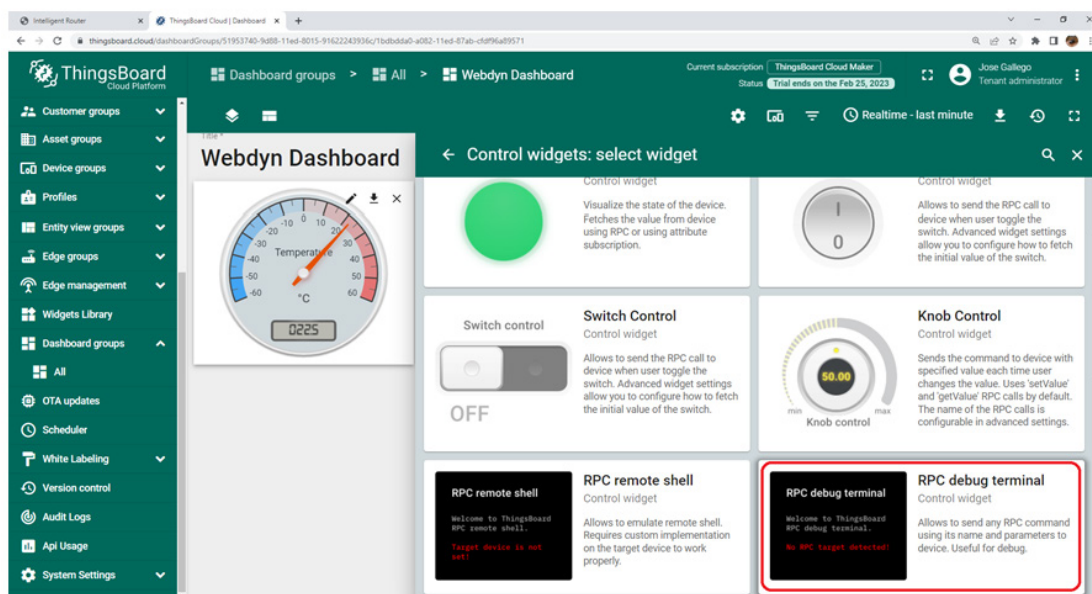
        //Enviamos un JSON con la respuesta al comando ejecutado
        var respuesta="{\"method\":\"command\", \"params\":\"\" + replaceSpecialChars(res)
        var r=mtx.mqttSend(respuesta,topicResponse,1);
    }
}

```

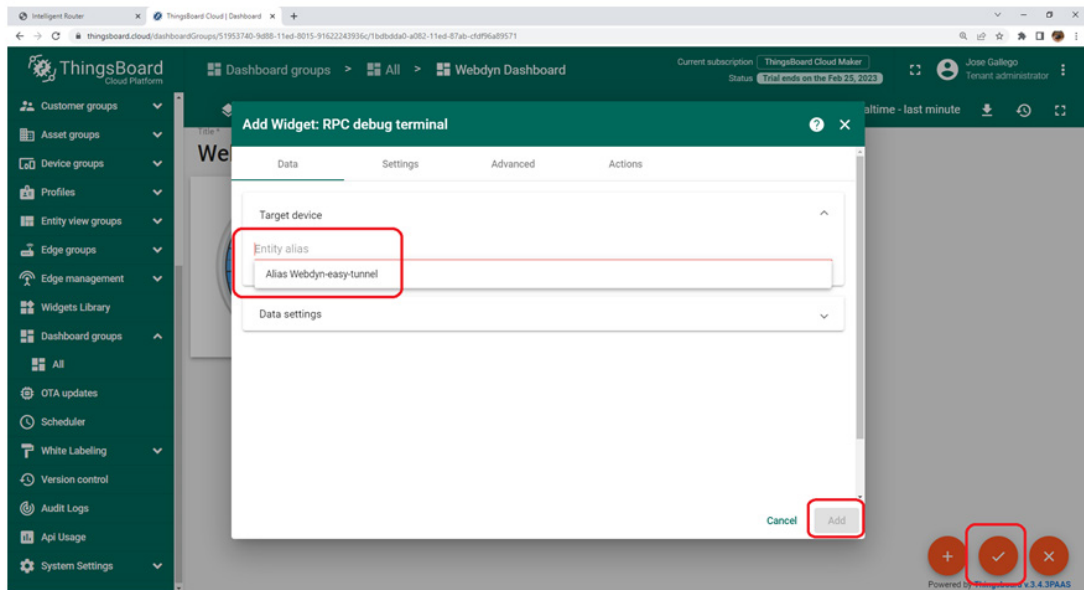
Por otro lado, en la plataforma ThingsBoard debe añadirse otro Widget, en este caso haciendo click en el tipo "Control widgets".



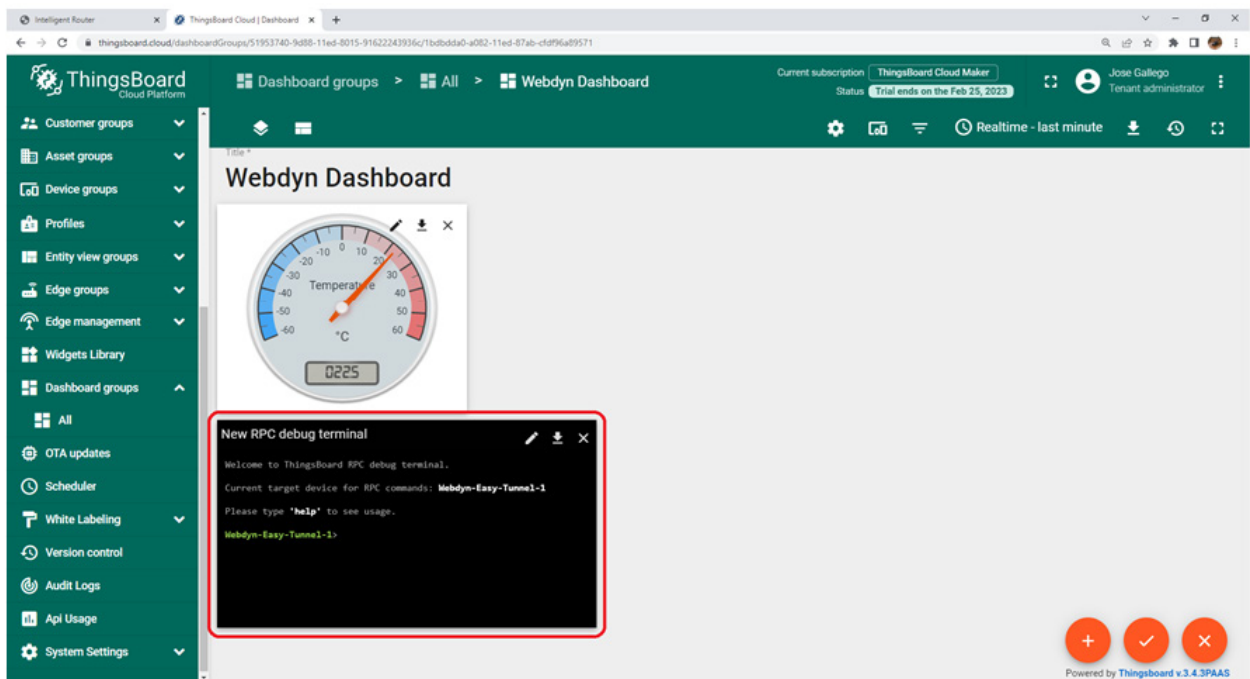
Y dentro de la sección “Control widgets” debe seleccionarse el tipo de widget “RPC debug Terminal”:



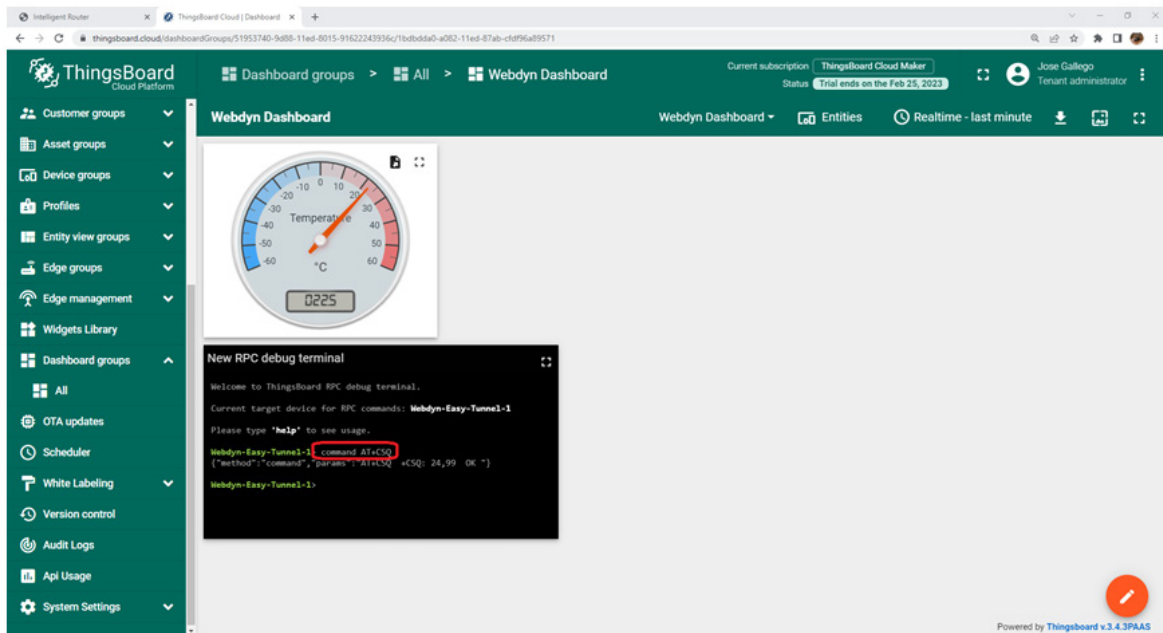
En el campo “Entity Alias” debe seleccionarse el alias creado anteriormente “Alias Webdyn-Easy-Tunnel” ,después debe pulsarse el botón “Add” y por último hacer click en el icono “V”.



En este punto aparecerá el widget que permite enviar comandos AT al dispositivo Titan de forma remota.



Por ejemplo, para enviar cualquier comando, basta con escribir el texto “command” seguido de 1 espacio en blanco y e introduciendo el comando a ejecutar. Por ejemplo, si se pretende ejecutar el comando AT+CSQ para conocer la cobertura de forma remota, bastaría con hacer lo que se muestra en la siguiente pantalla.



¿Más dudas?

Escríbanos sus consultas a iotsupport@mtxm2m.com