Ejemplo thingsBoard paso a paso

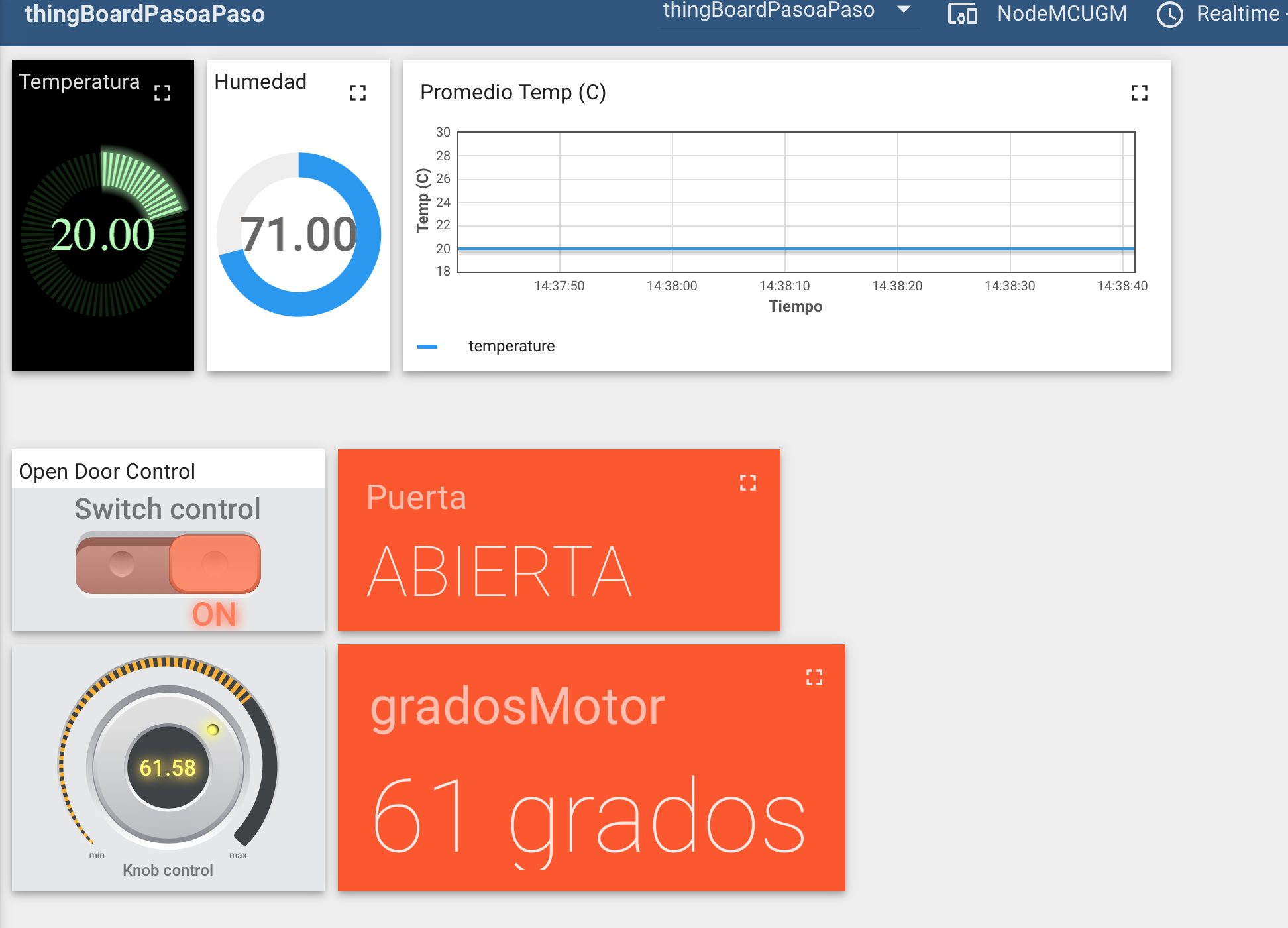
para profundizar ver : <https://blog.thingsboard.io/2017/01/esp8266-gpio-control-over-mqtt-using.html> y

<https://www.youtube.com/watch?v=JCW_hShAp7I&t=2s>

Este ejemplo muestra:

* cómo enviar telemetría de un sensor DHT11 (temperatura y humedad) hacia el servidor de thingsBoard (livedemo)
* como abrir y cerrar una puerta con un servo
* enviar los grados para posicionar el motor servo.

La información se despliega y manipula desde el siguiente dashboard



El nodeMCU tiene los siguientes sensores

* Sensor DGT11 conectado al pin D1, 3.3v y Gnd
* Servo Conectado al pin D4 y a Vin (5v) y Gnd

## Funcionamiento

Para enviar información de telemetría utilizando MQTT Thingsborad utiliza distintos tópicos que se definen en el sketch del nodeMCU.

### Para **Telemetría:** "v1/devices/me/telemetry";

### Para enviar comandos al dispositivo: "v1/devices/me/rpc/**request**/+"

### Para recibir respuestas a un request enviadas desde el dispositivo: "v1/devices/me/rpc/**response**/+"

### Reportar Temperatura y humedad

En el Dashboard se despliega la información del sensor de temperatura en los Widget Guage Digital y en un Chart. Estos datos los envía el nodeMCU como mensajes de telemetría.

El formato del mensaje JSON es { “temperature”: temperatura, “humidity”: humedad}

### Control de puerta

El SwitchControl (ON/OFF) controla una puerta (servo). “On” abre la puerta, Off la cierra. Cuando se activa el widget envía un mensaje al NodeMCU con el nombre de un método openDoor con un parámetro true o false.

El mensaje lo envía en formato JSON {“openDoor”: true} o {“openDoor”: false}

Luego de que se ejecuta el comando en el nodeMCU se responde con un mensaje que indica el estado de la puerta el cual se despliega en el Widget Card “Puerta”.

El mensaje lo envía en formato JSON {“openDoor”: true} o {“openDoor”: false}

{“doorState”: “ABIERTA”} o {“doorState”: “CERRADA”}

### Control del Servo

El widget KnobControl permite girar el servo determinada cantidad de grados. Cuando se gira el knob se envía un mensaje al servidor indicando la posición en grados a la cual se debe posicionar el servo.

El mensaje lo envía en formato JSON {“ rotateMotorValue”: grados}

Luego de que se ejecuta el comando en el nodeMCU se responde con un mensaje que indica la posición del servo en grados en el Widget Card “Grados Motor”.

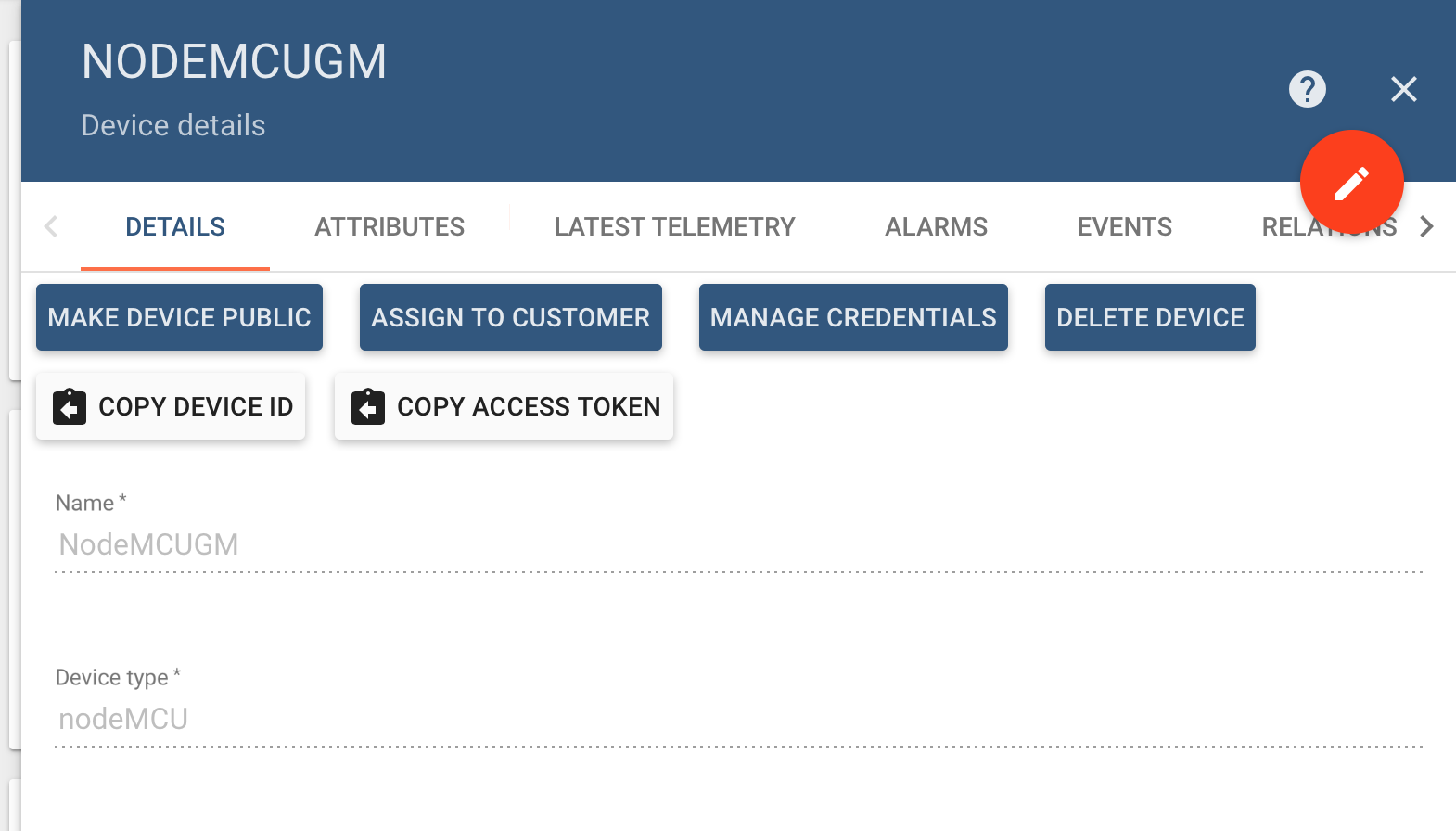
El mensaje lo envía en formato JSON {“gradosMotor”: grados}

# Instrucciones

## 1.- Crear un dispositivo

Crear en thingsboard un dispositivo (poner tipo “default)

Guardar el **Name** y el **Access token** de acceso



## 2.- Cargar el Scketch adjunto al NodeMCU

Cargar el sketch adjunto y cambiar la información de la RED y del DISPOSITIVO creado en thingsboard. **DESCOMENTAR LOS #define**

----- ver código sketch

// Configuración datos wifi

// decomentar el define y poner los valores de su red y de su dispositivo

**//#define WIFI\_AP "SSID RED"**

**//#define WIFI\_PASSWORD "PASSWORD RED"**

// configuración datos thingsboard

**//#define NODE\_NAME "NOMBRE DISPOSITIVO" //nombre que le pusieron al dispositivo cuando lo crearon**

**//#define NODE\_TOKEN "TOKEN DISPOSITIVO" //Token que genera Thingboard para dispositivo cuando lo crearon**

Donde **NODE\_NAME y NODE\_TOKEN** es copiado del dispositivo creado en thingsboard

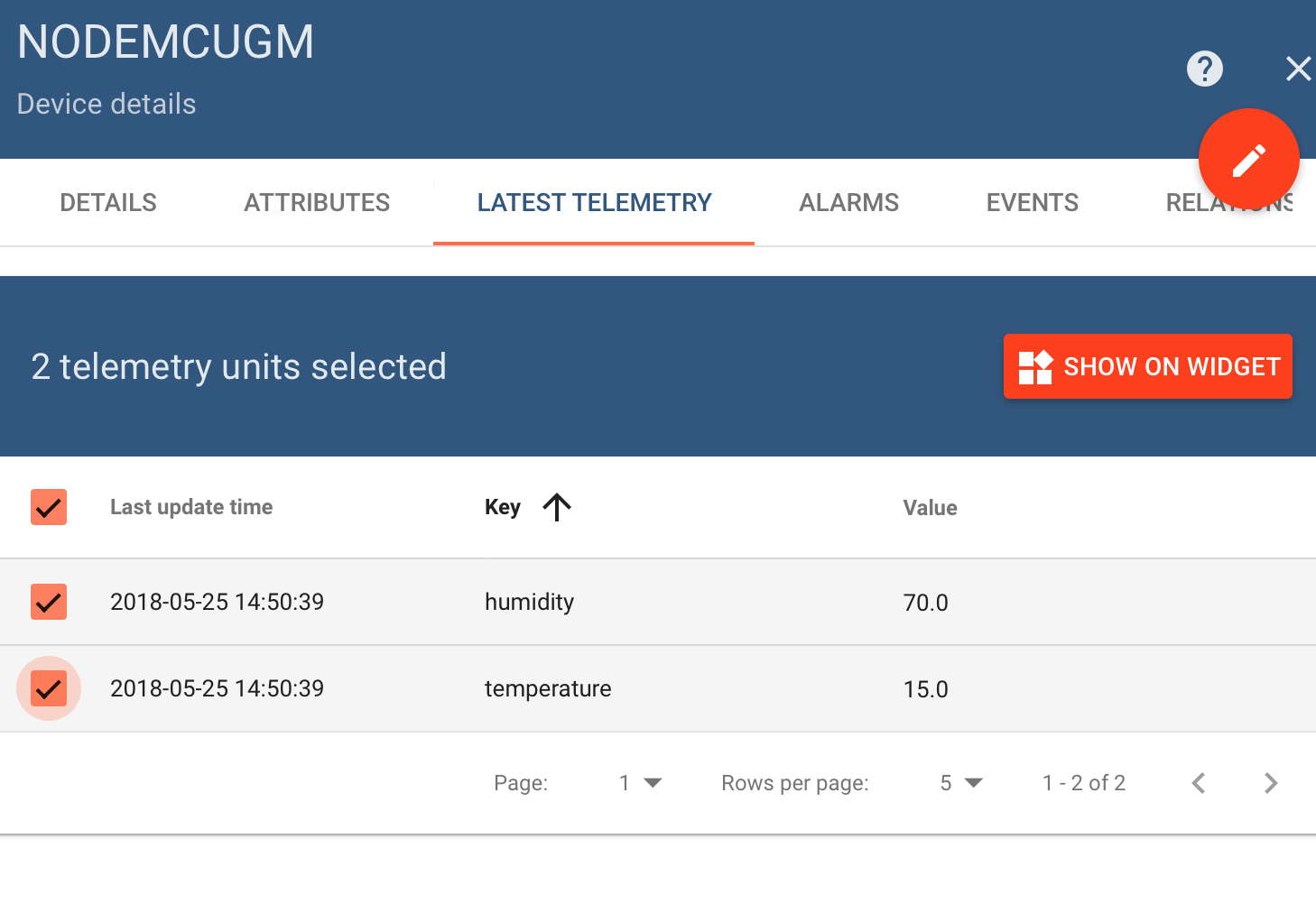
## 3.- Configurar sensores

El ejemplo esta conigurado para utilizar el DHT11 y un servo. Luego de que comprenda cómo funciona el código puede cambiarlo para que se ajuste a su proyecto. El código cuenta con comentarios que indican donde se debe \*\*\*\*\*\* MODIFICAR \*\*\*\*\*\*

## 4.- Asociar datos de telemetría a los widgets

Subir el Código al nodeMCU y ejecutarlo

Verificar que se reciban datos de los sensores en el tab **LATEST TELEMETRY** del panel del dispositivo en thingsboard (ver figura).

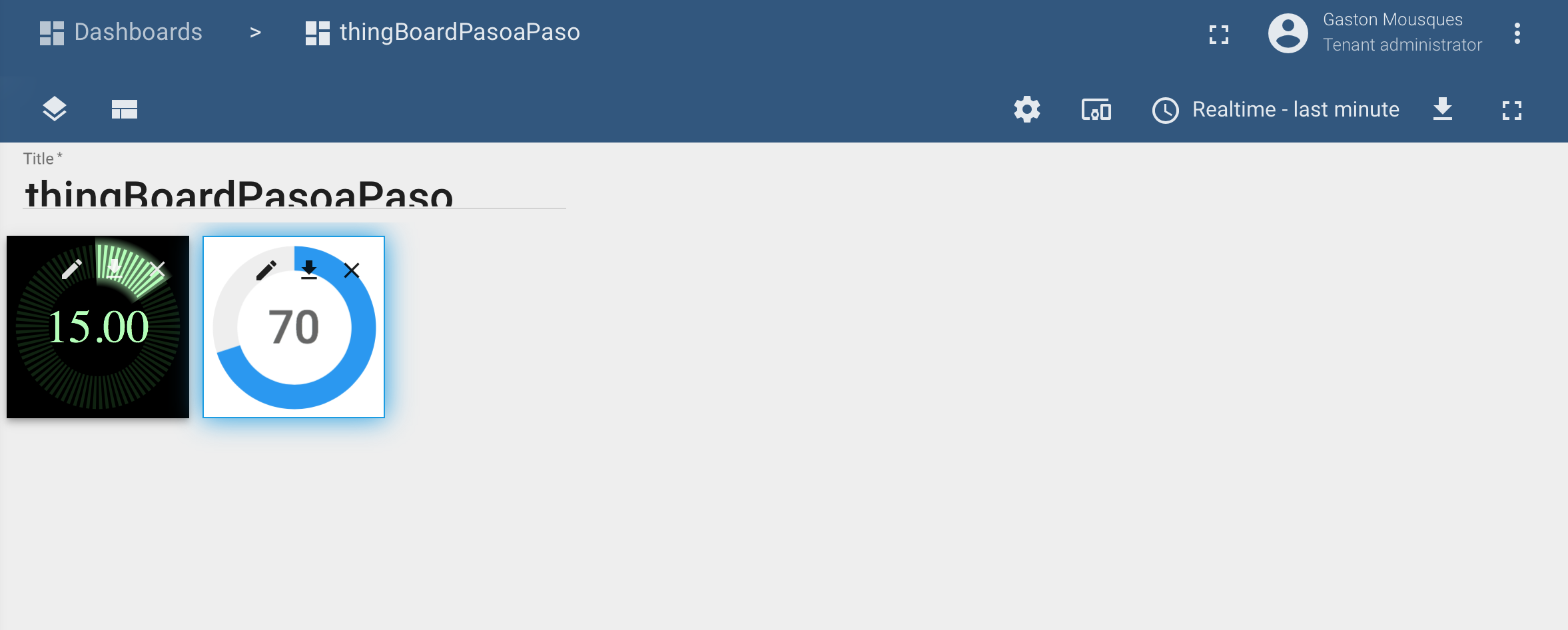


## Crear el dashboard y los widgets

Seleccionado los check boxes junto a las variables de telemetría agregar los widgets a un Dashboard (si no existe crearlo) utilizando el botón en rojo **SHOW ON WIDGET**

Ir al Dashboard y ajustar los widgets poniendo títulos y otras configuraciones

Se puede agregar nuevos widgets al dashboard



### Sensar en el nodeMCU

En este ejemplo en el loop() de nodeMCU se invoca un método **getAndSendData()** que es donde se envía la información mediante el tópico Telemetry

--- ver código en sketch para mayor claridad -----

void loop()

{

if ( !client.connected() ) {

reconnect();

}

if ( millis() - lastSend > elapsedTime ) { // Update and send only after 1 seconds

// FUNCION DE TELEMETRIA para enviar datos a thingsboard

**getAndSendData(); // FUNCION QUE ENVIA INFORMACIÓN DE TELEMETRIA**

lastSend = millis();

}

client.loop();

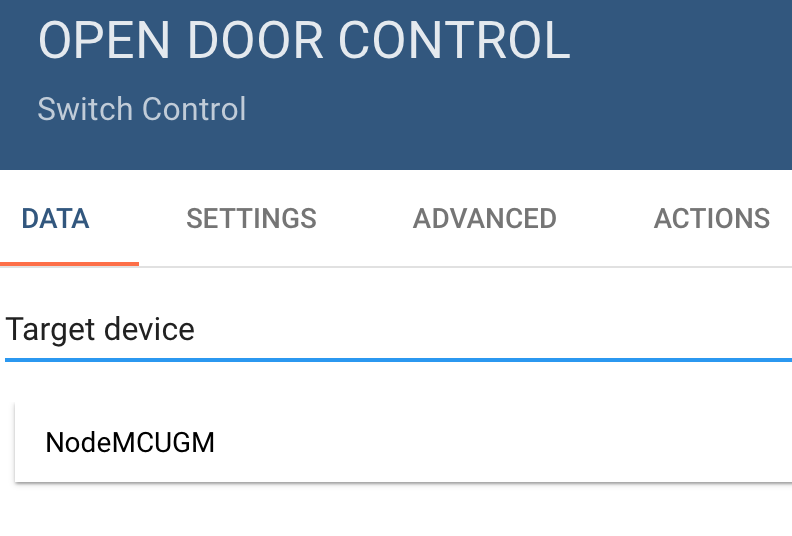
}

## 5.- Enviar comandos

Para controlar la puerta, agregar un widget de control al Dashboard

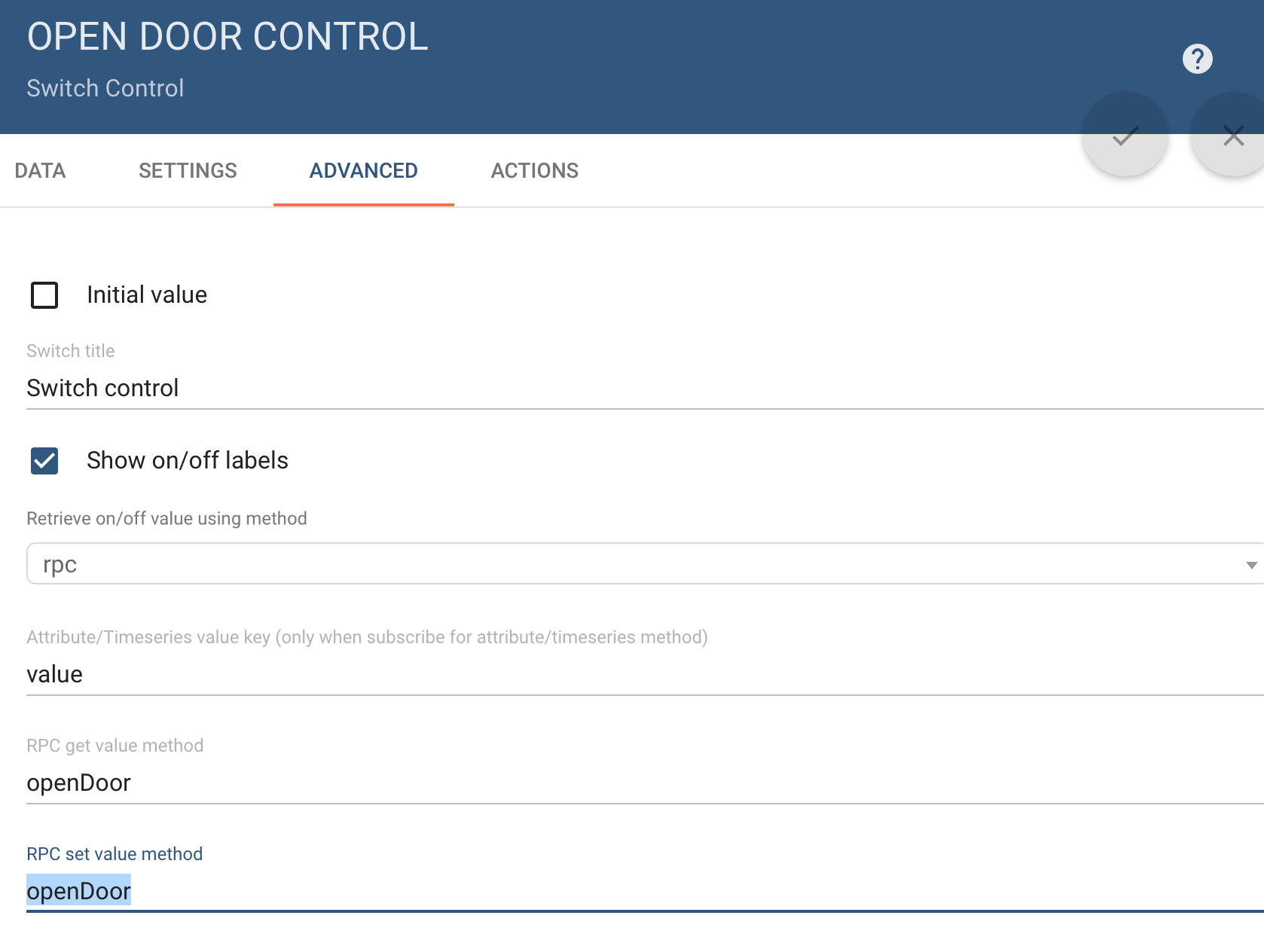


Editar el widget y en el target device poner el nombre del dispositivo que crearon



En la configuración del Switch en la hoja Advanced

Agregar el nombre del comando que se va a enviar cuando se active el Switch. En este caso **openDoor**.



## Intercambiar mensajes entre el servidor y el nodo

Cada vez que se active el switch se va a enviar el comando openDoor al nodeMCU mediante el tópico de request y con el formato JSON mencionado anteriormente.

Cuando el mansaje llega al nodeMCU se invoca la función **On\_Message()**

En esta función se parsea el mensaje para obtener el nombre del comando y los parámetros. Luego se invoca la función que realiza la operación.

En el sketch de Arduino en la función callback (on\_message) se agrega el método que se desea llamar para el comando que viene en el tópico.

Primero se parsea el nombre del método que vine en el mensaje.

En función del nombre del método se invoca el método correspondiente en el skectch, En este caso **openDoor** y **rotateMotorValue** (ver código debajo). Luego para se invoca el método que envía la respuesta utilizando el tópico

----- código -----

void on\_message(const char\* topic, byte\* payload, unsigned int length)

{

……..

…….

// Obtener el nombre del método invocado, esto lo envia el switch de la puerta y el knob del motor que están en el dashboard

String methodName = String((const char\*)data["method"]);

//responder según el método

**if (methodName.equals("openDoor")) {**

bool action = data["params"];

**String doorStatus = openOrCloseDoor (action);**

// responder al servidor con el estado de la puerta

**updateDoorStatus(doorStatus, topic);**

}

**else if (methodName.equals("rotateMotorValue")) {**

String gradosTemp = (data["params"]);

int grados = gradosTemp.toInt();

// se llama al motor para que gire los grados del parametro

**moveMotor(grados);**

**updateMotorStatus(grados, topic);**

}

## 6.- Recibir respuesta a comandos enviadas desde el dispositivo

Como se puede ver luego de llamar a las funciones que realizan las acciones se puede enviar una respuesta al servidor para que actualice algún widget, en este caso se actualizan los widgets del estado de la puerta y de los grados del servo

***El envió de mensajes de repuesta es opcional.***

Las funciones updateDoorStatus(doorStatus, topic); y updateMotorStatus(grados, topic);

envían mensajes de respuesta para actualizar los widgets Card

Para ello utilizan el tópico response y envían un JSON como se describe en la primera parte de este documento.

En este ejemplo luego de ejecutar el comendo OpenDoor() se envía una respuesta mediante la función **updateDoorStatus**. Como se puede ver en el código que sigue esta función realiza lo siguiente:

* Se cambia el nombre del tópico para utilizar el de respuesta
* Se arman el mensaje JSON con la respuesta
* Se publica el mensaje en el tópico.

----- código --------

void updateDoorStatus(String doorStatus, const char\* topic)

{

**// cambiar el topico de RPC a RESPONSE**

**String responseTopic = String(topic);**

**responseTopic.replace("request", "response"); //Notar que se cambio la palabra request por response en la cadena del topico**

// Prepare a JSON payload string dicendo el estado de la puerta, Notar que la tarjeta del dashboar tiene este atributo definido

**String payload = "{";**

**payload += "\"doorState\":"; payload += "\"" ; payload += doorStatus; payload += "\""; payload += "}";**

// Send payload

char attributes[100];

payload.toCharArray( attributes, 100 );

**// se envia la repsuesta la cual se despliegan en las tarjetas creadas para el atrubito**

**client.publish(attributesTopic, attributes);**

}

## Conectando los widgets para recibir mensajes

Para que el widget reciba los mensajes se debe agregar el mismo al Dashboard (en este caso un card Widget)

El intercambio de información se realiza utilizando atributos compartidos entre el servidor y el dispositivo.

Estos se agregan en la configuración del widget (o del dispositivo).

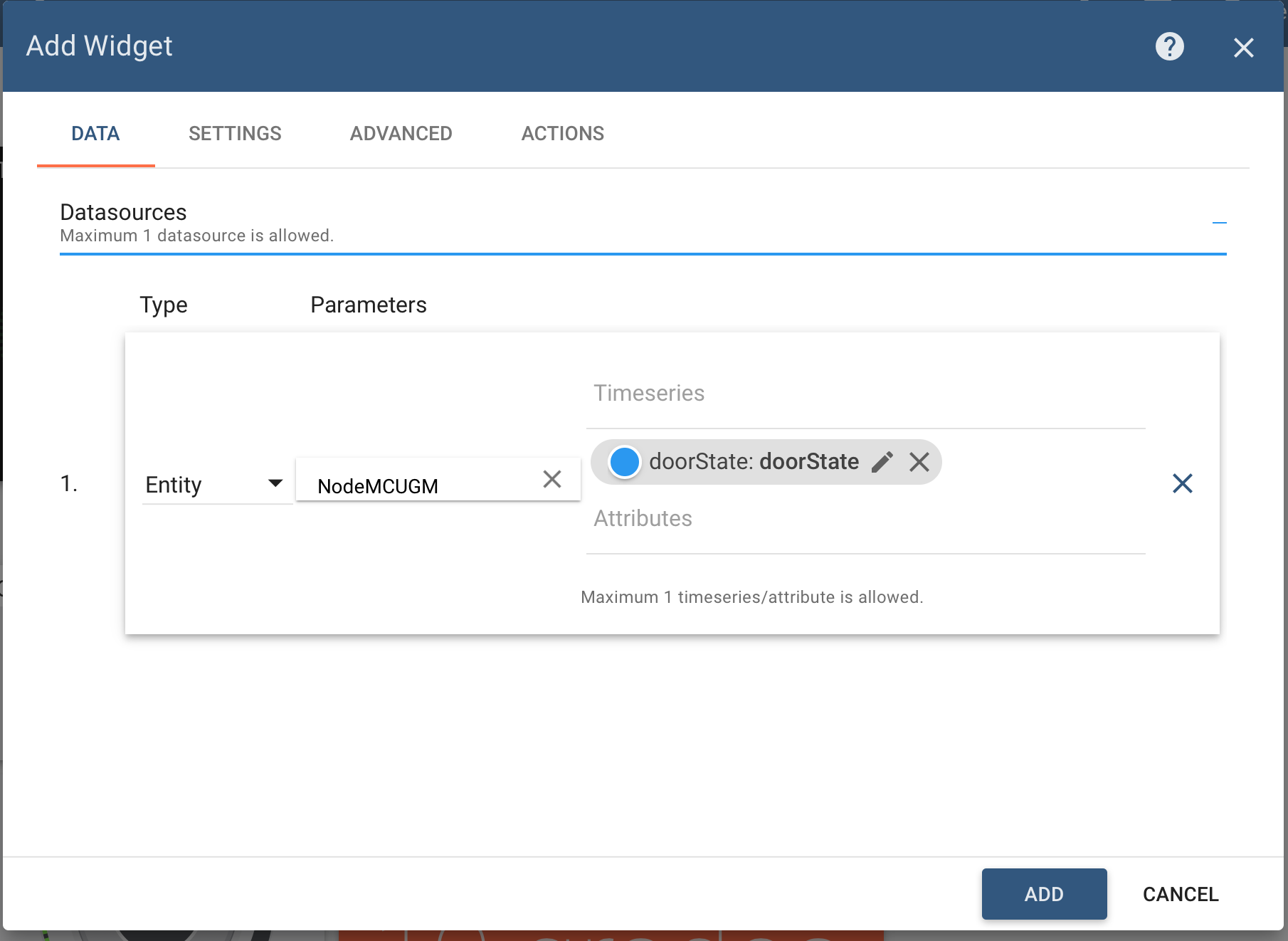
Al agregar el Widget se debe seleccionar el dispositivo y escribir en **attributes el nombre de la variable compartida y crearla.**

En el caso de la puerta sería **doorState** como se puede ver en el código de la función de respuesta de la sección anterior

String payload = "{";

payload += "\"**doorState**\":"; payload += "\"" ; payload += doorStatus; payload += "\""; payload += "}";

En la siguiente figura se muestra la configuración del widget.



Para Agregar otros widgets se debe seguir el mismo procediminto