

Sistemas Web

Práctica 1: Cliente IoT

Grado en Informática de Gestión
y Sistemas de Información

Dpto. de Ingeniería de Sistemas y Automática

Paso 10: ThinkSpeak

Oskar Casquero (oskar.casquero@ehu.eus)

María Luz Álvarez (mariluz.alvarez@ehu.eus)



Esta obra está bajo una [licencia de Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-sa/4.0/).



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

BILBOKO
INGENIARITZA
ESKOLA
ESCUELA
DE INGENIERÍA
DE BILBAO



Estructura del tema

- Presentación de la plataforma de IoT ThingSpeak.
- Almacenamiento de datos en *canales*.
- Gestión de un canal.
- API REST de ThingSpeak.
- Ejemplos para el envío de datos:
 - Enviar los datos en formato formulario y recibir una respuesta vacía.
 - Enviar los datos en formato formulario y recibir una respuesta en formato JSON.
 - Enviar los datos en formato JSON y recibir una respuesta en formato XML.

1

Plataforma de IoT ThingSpeak

- ThingSpeak ofrece un servicio de almacenamiento de datos y una interfaces (HTTP y MQTT) para su acceso.
- La licencia de uso gratuita, que se renueva cada año automáticamente, permite enviar 3.000.000 de mensajes.

My Account

License Type: Free

Free Messages	Messages Remaining	Expiration Date
3.000.000	2.940.461	04 Dec 2019

To purchase a license:

[How to Buy](#)

Recommended maximum daily usage based on your annual capacity: **8.219 messages**

For questions about exceeding suggested daily usage rate or for purchasing new units, see the [Licensing FAQ](#)





2

Almacenamiento de datos en canales

Los datos se almacenan en “canales” asociados a cuentas de usuario. Los canales pueden ser públicos o privados.

Canales
públicos

Canales
privados

My Channels			
New Channel		<input type="text" value="Search by tag"/> <input type="button" value="Q"/>	
Name	Created	Updated	
 Weather Station in Amurrio, Basque Country <small>temperature, relative humidity, atmospheric pressure, DS18B20, HH10D, BMP180, Arduino, xbee</small> Private Public Settings Sharing API Keys Data Import / Export	2016-11-28	2018-09-03 18:59	
 Mobile weather station <small>PiE</small> Private Public Settings Sharing API Keys Data Import / Export	2018-07-24	2018-12-14 14:42	
 GSM MQTT client Private Public Settings Sharing API Keys Data Import / Export	2018-07-26	2018-07-30 12:02	
 Prueba Private Public Settings Sharing API Keys Data Import / Export	2019-01-17	2019-01-17 20:44	

2

Almacenamiento de datos en canales

- En un canal se pueden guardar hasta 8 campos de datos.
- Las etiquetas que se asignan a los campos son descriptores. Los nombres de los campos son **field1, field2, field3...**

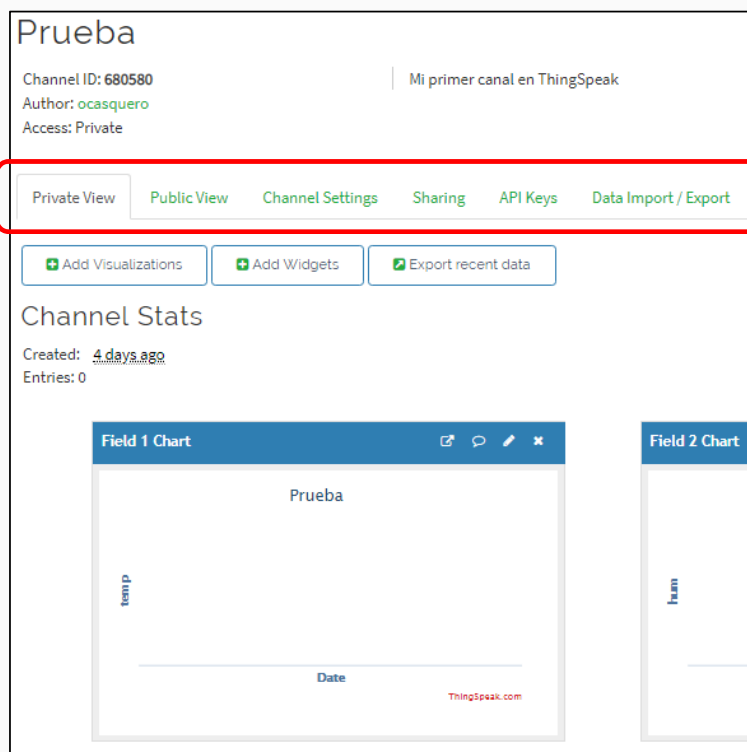
New Channel

Name

Description

Field 1	<input type="text" value="temp"/>	<input checked="" type="checkbox"/>
Field 2	<input type="text" value="hum"/>	<input checked="" type="checkbox"/>
Field 3	<input type="text"/>	<input type="checkbox"/>
Field 4	<input type="text"/>	<input type="checkbox"/>
Field 5	<input type="text"/>	<input type="checkbox"/>
Field 6	<input type="text"/>	<input type="checkbox"/>
Field 7	<input type="text"/>	<input type="checkbox"/>
Field 8	<input type="text"/>	<input type="checkbox"/>

Gestión de un canal



- Private view y Public view permiten visualizar los datos.
- Channel settings permite:
 - Editar los campos de datos
 - Vaciar el canal
 - Borrar el canal
- Sharing permite compartir el canal
- API keys:
 - Write API key para escribir datos en el canal.
 - Read API key para leer datos de un canal privado.
- Data import/export

API REST de ThingSpeak

- El API REST de ThingSpeak está constituido por el conjunto de operaciones/servicios que ThingSpeak ofrece para gestionar canales mediante el protocolo HTTP.
- El API REST de ThingSpeak está documentado en <https://es.mathworks.com/help/thingspeak/rest-api.html>

REST API

Use REST API calls to create and update ThingSpeak™ channels and charts

Representational state transfer (REST) is an architectural style designed as a request-response model that communicates using GET, POST, PUT, and DELETE to create and delete channels, read and write channel data, and clear the data in a channel. Web browsers use this interface to retrieve web pages or to send data to remote servers. You can also use REST API calls with [Act on Data](#) which let you interact with social media, web services, and devices.

RESTful API Reference

Read Data	
Read Data	Read data from all fields in a channel with HTTP GET
Read Field	Read data from a single field of a channel with HTTP GET
Read Status	Read status field of a channel with HTTP GET
Read Last Entry	Read the last entry in a channel with HTTP GET
Read Last Field Entry	Read the last entry in a field of a channel with HTTP GET
Read Last Status	Read the last status of a channel with HTTP GET

Tipos de servicios

Operaciones de lectura

Write Data

View Channels and Channel Settings

Create and Delete

API REST de ThingSpeak

La documentación indica cómo hay que construir la petición HTTP para el envío de datos.

Write Data
Update channel data with HTTP GET or POST

Request

HTTP Method
POST or GET

URL
`https://api.thingspeak.com/update.<format>`

URL Parameters

Name	Description
<format>	(Required) Format for the HTTP response, specified as blank, json, or xml.

Example: `https://api.thingspeak.com/update.json`

Body

Name	Description	Value-Type
api_key	(Required) Specify Write API Key for this specific channel. The Write API Key can optionally be sent via a THINGSPEAKAPIKEY HTTP header. The Write API Key is found on the API Keys tab of the channel view.	{Write - API - Key string}
field<X>	(Optional) Field X data, where X is the field ID.	<any>
lat	(Optional) Latitude in degrees.	<decimal>
long	(Optional) Longitude in degrees.	<decimal>
elevation	(Optional) Elevation in meters.	<integer>
status	(Optional) Status update message.	<string>
twitter	(Optional) Twitter™ username linked to ThingTweet.	<string>
tweet	(Optional) Twitter status update.	<string>
created_at	(Optional) Date when feed entry was created, in ISO 8601 format, for example: 2014-12-31 23:59:59. Must be unique within channel. Time zones can be specified via the timezone parameter.	<datetime>

Content-Type
Content-Type is required only for the POST method, not for the GET method.
application/x-www-form-urlencoded for most updates.
application/json for updates in JSON format.

PETICIONES DE EJEMPLO:

Enviar los datos en formato formulario y recibir una respuesta vacía (devuelve el ID del registro si OK y "0" si error)

POST /update HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

api_key=OQ1CVZKQX7HV9J8Y&field1=22&field2=47

Enviar los datos en formato formulario y recibir una respuesta en formato JSON (devuelve el registro si OK y "0" si error)

POST /update.json HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

api_key=OQ1CVZKQX7HV9J8Y&field1=22&field2=47

Enviar los datos en formato JSON y recibir una respuesta en formato XML (devuelve el registro si OK y "0" si error)

POST /update.xml HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/json
Content-Length: 58

{"api_key":"OQ1CVZKQX7HV9J8Y","field1":"22","field2":"47"}



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

BILBOKO
INGENIARITZA
ESKOLA
ESCUELA
DE INGENIERÍA
DE BILBAO

Envío de datos para su almacenamiento en un canal

**Ejemplo 1: Enviar los datos en formato formulario y recibir una respuesta vacía
(devuelve el ID del registro si OK y “0” si error)**

PETICIONES DE EJEMPLO:

Enviar los datos en formato formulario y recibir una respuesta vacía
(devuelve el ID del registro si OK y “0” si error)

POST /update HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

api_key=OQ1CVZKQX7HV9J8Y&field1=22&field2=47

Enviar los datos en formato formulario y recibir una respuesta en
formato JSON (devuelve el registro si OK y “0” si error)

POST /update.json HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

api_key=OQ1CVZKQX7HV9J8Y&field1=22&field2=47

Enviar los datos en formato JSON y recibir una respuesta en formato
XML (devuelve el registro si OK y “0” si error)

POST /update.xml HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/json
Content-Length: 58

{"api_key": "OQ1CVZKQX7HV9J8Y", "field1": "22", "field2": "47"}

Response

```
Raw Headers Hex
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 1
Connection: close
Status: 200 OK
X-Frame-Options: SAMEORIGIN
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE, PATCH
Access-Control-Allow-Headers: origin, content-type, X-Requested-With
Access-Control-Max-Age: 1800
ETag: "1679091c5a880faf6fb5e6087eb1b2dc"
Cache-Control: max-age=0, private, must-revalidate
Set-Cookie: request_method=POST; path=/
X-Request-Id: 17abd716-ca4f-4c30-920c-46ac4e3ba3f3
X-Runtime: 0.090177
X-Powered-By: Phusion Passenger 4.0.57
Date: Tue, 22 Jan 2019 11:05:22 GMT
Server: nginx/1.9.3 + Phusion Passenger 4.0.57
```

6

Envío de datos para su almacenamiento en un canal

Ejemplo 2: Enviar los datos en formato formulario y recibir una respuesta en formato JSON (devuelve el registro si OK y "0" si error)

PETICIONES DE EJEMPLO:

Enviar los datos en formato formulario y recibir una respuesta vacía
(devuelve el ID del registro si OK y "0" si error)

POST /update HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

api_key=OQ1CVZKQX7HV9J8Y&field1=22&field2=47

Enviar los datos en formato formulario y recibir una respuesta en
formato JSON (devuelve el registro si OK y "0" si error)

POST /update.json HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

api_key=OQ1CVZKQX7HV9J8Y&field1=22&field2=47

Enviar los datos en formato JSON y recibir una respuesta en formato
XML (devuelve el registro si OK y "0" si error)

POST /update.xml HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/json
Content-Length: 58

{"api_key":"OQ1CVZKQX7HV9J8Y","field1":"22","field2":"47"}

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Connection: close
Status: 200 OK
X-Frame-Options: SAMEORIGIN
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE, PATCH
Access-Control-Allow-Headers: origin, content-type, X-Requested-With
Access-Control-Max-Age: 1800
ETag: "c2841a2a7a7457a8b37f58a7b112e2d1"
Cache-Control: max-age=0, private, must-revalidate
Set-Cookie: request_method=POST; path=/
X-Request-Id: 0723bd68-73f8-493e-a09e-32f50a06c73f
X-Runtime: 0.033213
X-Powered-By: Phusion Passenger 4.0.57
Date: Tue, 22 Jan 2019 11:10:33 GMT
Server: nginx/1.9.3 + Phusion Passenger 4.0.57
Content-Length: 246

{"channel_id":680580,"created_at":"2019-01-22T11:10:33Z","entry_id":7,"field1":"22","field2":"47",
,"field3":null,"field4":null,"field5":null,"field6":null,"field7":null,"field8":null,"latitude":
:null,"longitude":null,"elevation":null,"status":null}
```



Envío de datos para su almacenamiento en un canal

Ejemplo 3: Enviar los datos en formato JSON y recibir una respuesta en formato XML (devuelve el registro si OK y "0" si error)

PETICIONES DE EJEMPLO:

Enviar los datos en formato formulario y recibir una respuesta vacía (devuelve el ID del registro si OK y "0" si error)

POST /update HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

api_key=OQ1CVZKQX7HV9J8Y&field1=22&field2=47

Enviar los datos en formato formulario y recibir una respuesta en formato JSON (devuelve el registro si OK y "0" si error)

POST /update.json HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

api_key=OQ1CVZKQX7HV9J8Y&field1=22&field2=47

Enviar los datos en formato JSON y recibir una respuesta en formato XML (devuelve el registro si OK y "0" si error)

POST /update.xml HTTP/1.1
Host: api.thingspeak.com
Content-Type: application/json
Content-Length: 58

{"api_key":"OQ1CVZKQX7HV9J8Y","field1":"22","field2":"47"}

Response

Raw Headers Hex XML

HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8
Connection: close
Status: 200 OK
X-Frame-Options: SAMEORIGIN
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE, PATCH
Access-Control-Allow-Headers: origin, content-type, X-Requested-With
Access-Control-Max-Age: 1800
ETag: "331bb5526b094f19368308ddb73af7ce"
Cache-Control: max-age=0, private, must-revalidate
Set-Cookie: request_method=POST; path=/
X-Request-Id: cc7c47e8-8d8c-450e-8d8e-a0cbe9b1afc8
X-Runtime: 0.042861
X-Powered-By: Phusion Passenger 4.0.57
Date: Tue, 22 Jan 2019 11:12:39 GMT
Server: nginx/1.9.3 + Phusion Passenger 4.0.57
Content-Length: 489

```
<?xml version="1.0" encoding="UTF-8"?>
<feed>
  <channel-id type="integer">680580</channel-id>
  <created-at type="dateTime">2019-01-22T11:12:39Z</created-at>
  <entry-id type="integer">8</entry-id>
  <field1>22</field1>
  <field2>47</field2>
  <field3 nil="true"/>
  <field4 nil="true"/>
  <field5 nil="true"/>
  <field6 nil="true"/>
  <field7 nil="true"/>
  <field8 nil="true"/>
  <latitude nil="true"/>
  <longitude nil="true"/>
  <elevation nil="true"/>
  <status nil="true"/>
</feed>
```

