



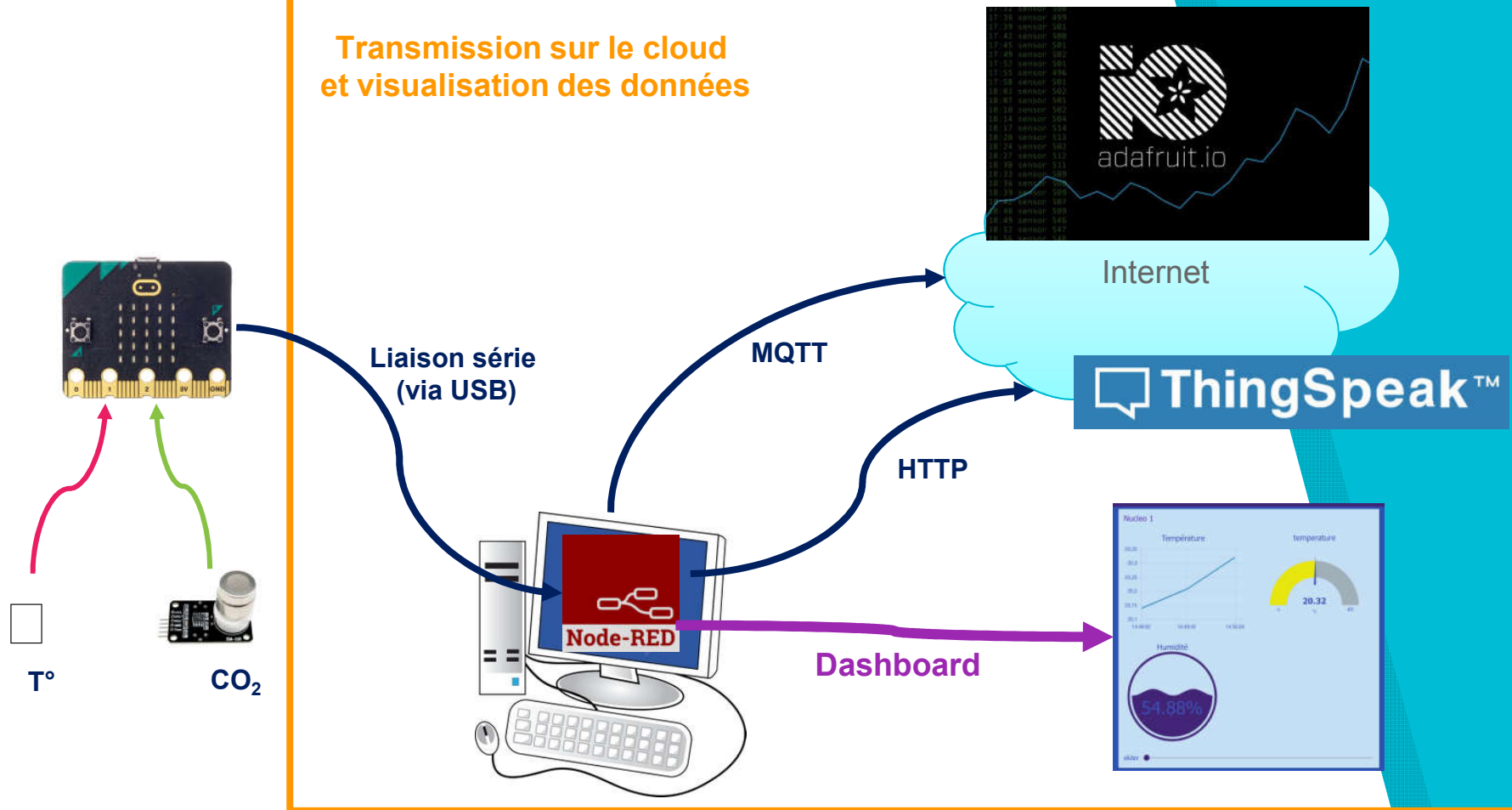
Architecture IoT (alternative au composant UART)



SOMMAIRE

- ▶ Architecture cible
- ▶ Installer Node-Red (+ node serial)
- ▶ Configurer le cloud (io.adafruit.com ou thingspeak)
- ▶ Configurer un node sous Node-Red
 - ▷ mqtt in
 - ▷ Ou HTTP Request
- ▶ Configurer le flow Node-Red

Transmission sur le cloud et visualisation des données





INSTALLATION DE NODE-RED

- ▶ Raspberry Pi / VM Debian : utiliser le script curl

<https://nodered.org/docs/getting-started/raspberrypi>

```
sudo apt install build-essential git curl
```

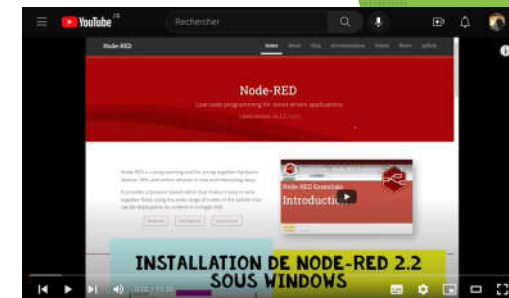
```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

- ▶ Commande « `node-red` » pour démarrer le serveur

- ▶ Installation sous Windows

- ▷ Tuto video :

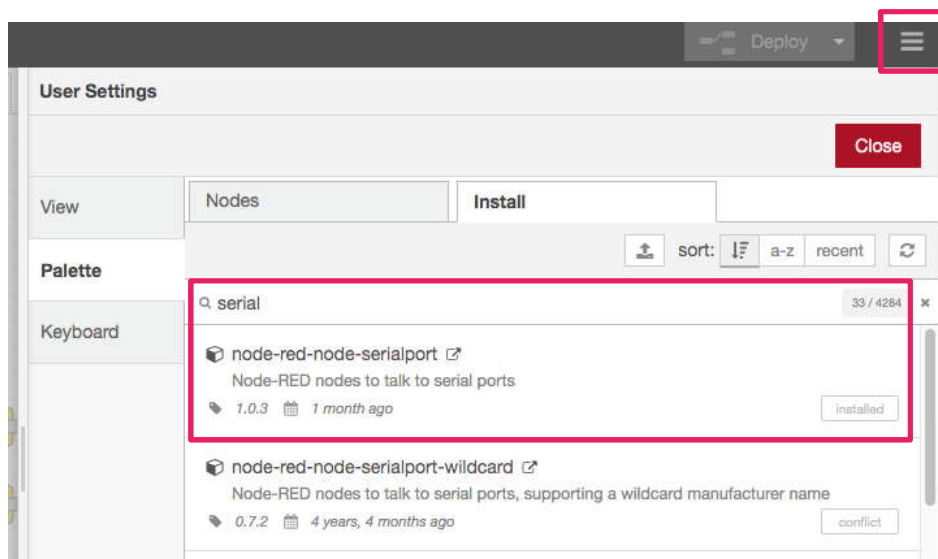
- ▷ <https://youtu.be/Y6uKy5OCDgA>





INSTALLATION DU NODE SERIAL

- ▶ Menu principal / manage palette / Install



Tuto video :

<https://youtu.be/Y6uKy5OCDgA>





RÉCUPÉRATION DES DONNÉES : NODE-RED

- ▶ Node-Red remplace le composant UART
- ▶ Attention au programme Python
 - ▷ Initialisation de la liaison série
 - ▷ `uart.init(9600)`
 - ▷ (sans pin RX/TX)
- ▶ <https://microbit-micropython.readthedocs.io/en/latest/uart.html>

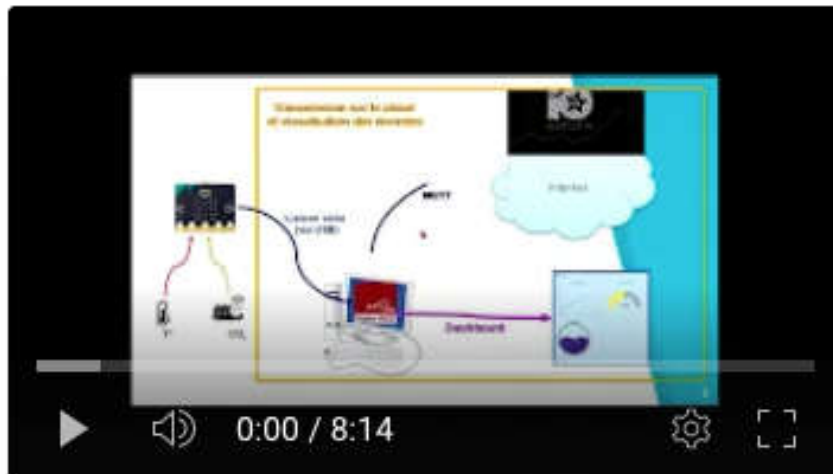
ⓘ Warning

Initializing the UART on external pins will cause the Python console on USB to become unaccessible, as it uses the same hardware. To bring the console back you must reinitialize the UART without passing anything for `tx` or `rx` (or passing `None` to these arguments).



TUTO YOUTUBE : RÉCUPÉRER LES DONNÉES DE LA MICRO:BIT

Architecture IoT (1) : récupérer sous Node-Red les données envoyées par Micro:bit sur liaison série



<https://youtu.be/J2-RCy4Tgw>



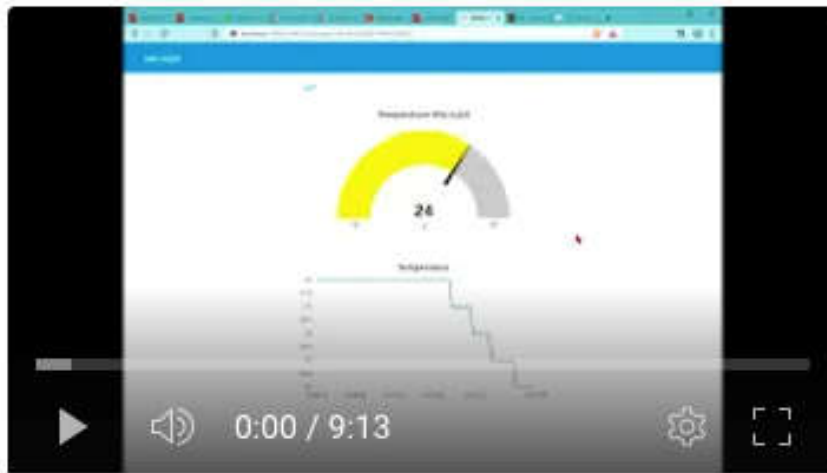
CODE : TEMPÉRATURE INTERNE DE LA MICRO:BIT

```
1 """ Programme de la micro::bit """
2
3 import os
4 import sys
5
6 from microbit import *
7
8 # liaison série
9 # ne pas mettre les broches tx et rx pour que le serial soit disponible pour les données
10 uart.init(9600)
11
12 # boucle infinie
13 while 1:
14     # temporisation 1s
15     sleep(2000)
16     # lecture de la temperature interne du microbit
17     dataTemp = temperature()
18     print(dataTemp)
19     # creation du buffer format string
20     buffer = str(dataTemp)+ "\n"
21     # ecriture sur la liaison série
22     uart.write(buffer)
23
24 sys.exit()
25
```




TUTO YOUTUBE : DASHBOARD ET ENVOI DES DONNÉES AVEC MQTT

Architecture IoT (2) : Créer un tableau de bord sous Node-red (dashboard)



https://youtu.be/MiYBG7_QW4



VISUALISATION DES DONNÉES : DASHBOARD ADAFRUIT

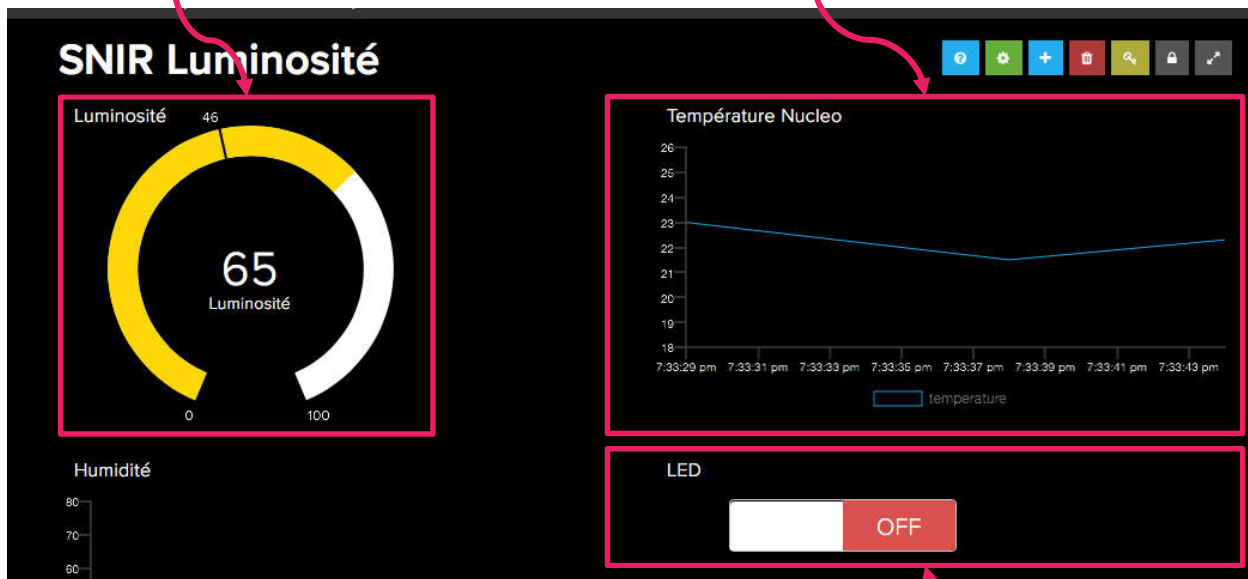
- ▶ Cloud Adafruit (io.adafruit.com)
- ▶ Compte gratuit permet d'avoir
 - ▷ 10 fils de données (feeds)
 - ▷ 30 données par minute
 - ▷ 5 tableaux de bord (dashboards)
 - ▷ Plusieurs widgets par tableau de bord
- ▶ Broker MQTT : io.adafruit.com



CLOUD ADAFRUIT : EXEMPLE DE DASHBOARD

Widget « Gauge »

Widget « Line Chart »



Widget « Toggle » (interrupteur)



NODE MQTT POUR IO.ADAFRUIT.COM LE BROKER (SERVEUR)

Configuration du node MQTT IN (Node-Red)

mqtt in

Name: Adafruit Prof

Connection | Security | Messages

Server: io.adafruit.com Port: 1883

Enable secure (SSL/TLS) connection

Client ID: Leave blank for auto generated

Port par défaut
du protocole MQTT

Hostname du serveur
(broker MQTT)



NODE MQTT POUR IO.ADAFRUIT.COM IDENTIFICATION

Configuration du node MQTT IN (Node-Red)

Edit mqtt out node > Edit mqtt-broker node

Delete Cancel Update

Properties

Name Adafuit Prof

Connection Security Messages

Username ProfTonnerre

Password

- C'est « l'active key » qui joue le rôle du mot de passe
- L'affichage présente toujours 8 étoiles

Information à récupérer sur le compte Adafuit (API Key)

Shop Learn Blog Forums LIVE! AdaBox IO Account 0

adafuit Devices Feeds Dashboards Actions Power-Ups New Device

YOUR ADAFRUIT IO KEY

Your Adafuit IO Key should be kept in a safe place and treated with the same care as your Adafuit username and password. People who have access to your Adafuit IO Key can view all of your data, create new feeds for your account, and manipulate your active feeds.

If you need to regenerate a new Adafuit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

Username ProfTonnerre

Active Key aio_H...6N1 REGENERATE KEY

Hide Code Samples

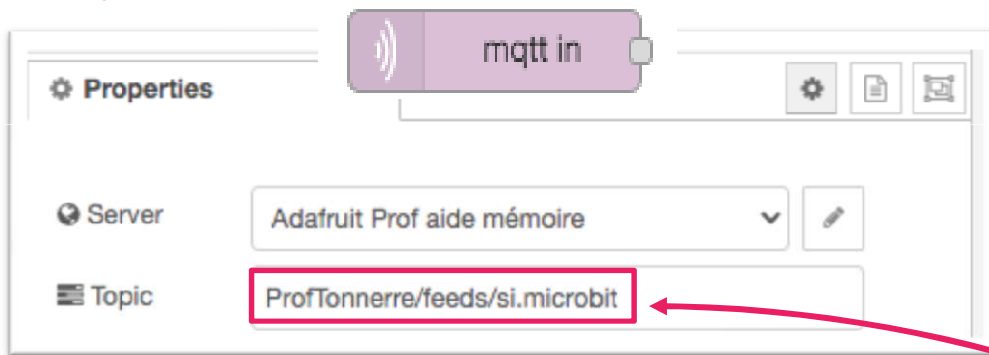
Arduino

```
#define IO_USERNAME "ProfTonnerre"
#define IO_KEY "aio_H...6N1"
```



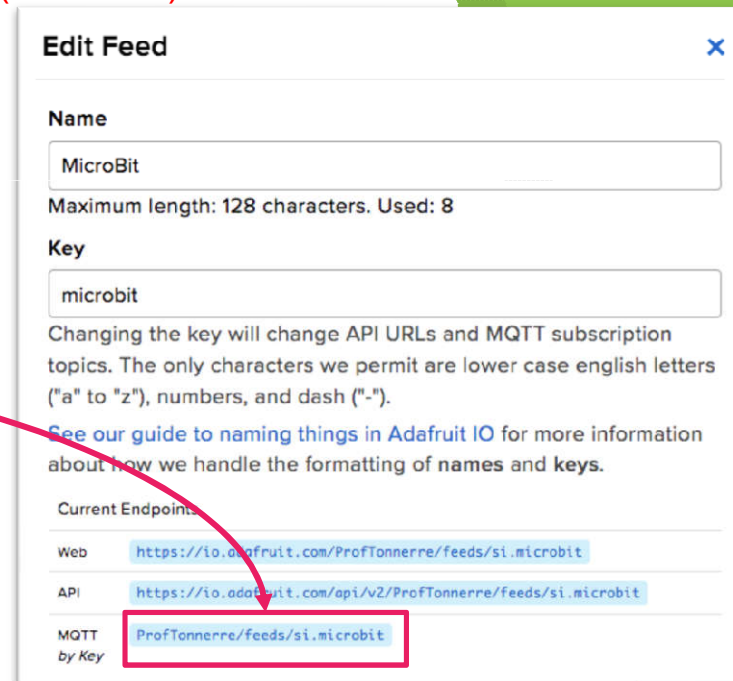
NODE MQTT POUR IO.ADAFRUIT.COM LE FIL DE DONNÉES (TOPIC)

Configuration du node MQTT IN (Node-Red)



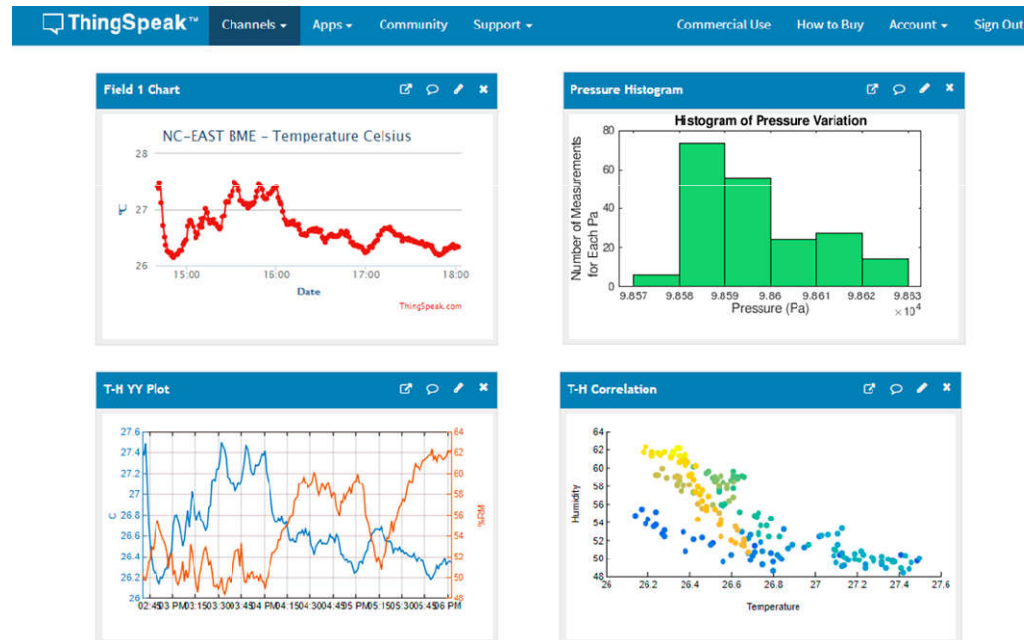
topic = <username>/<feeds>/<feed_group>.<feedname>

Information à récupérer sur le compte Adafruit (Feed Info)





UTILISATION DU CLOUD THINGSPEAK EN MQTT



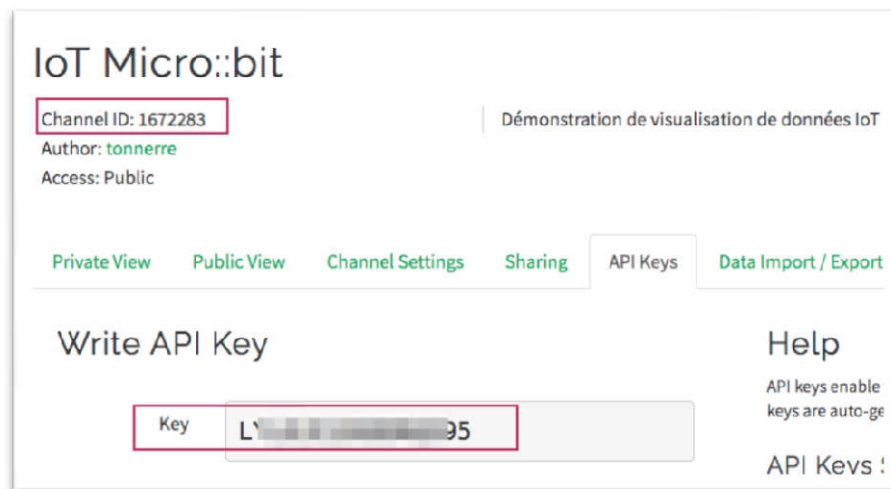


COMPTE GRATUIT THINGSPEAK

- ▶ Cloud Adafruit (io.adafruit.com)
- ▶ Compte gratuit permet d'avoir
 - ▷ 4 channels
 - ▷ 8 fields par channel
 - ▷ 1 msg toutes les 15s
- ▶ Broker MQTT : mqtt3.thingspeak.com
- ▶ API REST (HTTP)

UTILISATION DE THINGSPEAK EN MQTT (1)

- ▶ Créer un channel
 - ▷ Nommer un field (ex: temperature1)
 - ▷ Générer une clef API d'écriture (Write API key)



The screenshot shows the 'Write API Key' page in the Thingspeak interface. At the top, the channel name 'IoT Micro::bit' is displayed. Below it, the 'Channel ID: 1672283' is highlighted with a red box. Other details include 'Author: tonnerre' and 'Access: Public'. A navigation bar contains tabs for 'Private View', 'Public View', 'Channel Settings', 'Sharing', 'API Keys' (which is selected), and 'Data Import / Export'. The main section is titled 'Write API Key' and features a 'Key' label next to a text input field containing a partially obscured key ending in '95', which is also highlighted with a red box. To the right, there is a 'Help' section with the text 'API keys enable keys are auto-g...' and 'API Keys :'. The page title 'Démonstration de visualisation de données IoT' is visible in the top right corner.



UTILISATION DE THINGSPEAK EN MQTT (2)

- ▶ Créer un device
 - ▷ Autoriser le device sur le ou les channels créés
 - ▷ On obtient un clientID / Username / mot de passe (à sauvegarder)

Add a new device

Device Information

Name* device1

Description Micro:bit Device1

Authorize channels to access ⓘ

-- Select a Channel --

...

Add Channel

Authorized Channel ⓘ	Allow Publish	Allow Subscribe
IoT Micro:bit (1572283)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Cancel Add Device

MQTT Credentials

Use these MQTT credentials to publish and subscribe to ThingSpeak channels. [Learn More](#)

Client ID MzE: [masked] [copy]

Username MzE: [masked] [copy]

Password [masked] [toggle] [copy]

⚠ ThingSpeak does not store a copy of your device's MQTT password. Download or copy it to keep it safe.

Download Credentials -



NODE MQTT POUR THINGSPEAK LE BROKER (SERVEUR)

Configuration du node MQTT IN (Node-Red)

Attention ! Il y a eu une modification
mqtt3 au lieu de mqtt. Se référer à la
[documentation Matlab](#)

The screenshot shows the configuration interface for the 'mqtt in' node in Node-Red. The 'Name' field is set to 'ThingSpeak'. The 'Connection' tab is selected, showing the 'Server' field with the value 'mqtt3.thingspeak.com' and 'Port' set to '1883'. The 'Client ID' field contains a masked value 'MzE...3Q'. Other options include 'Keep alive time (s)' set to 60, 'Use clean session' checked, and 'Use legacy MQTT 3.1 support' unchecked. A red box highlights the 'Client ID' field, and a red arrow points from the 'MQTT Credentials' dialog to it.

Information à récupérer des « MQTT
credentials » (après avoir créer un device)

The screenshot shows the 'MQTT Credentials' dialog box. It contains fields for 'Client ID', 'Username', and 'Password', all with masked values. A red box highlights the 'Client ID' field, and a red arrow points from the 'MQTT Credentials' dialog to the 'Client ID' field in the Node-Red configuration. Below the fields is a warning message: 'ThingSpeak does not store a copy of your device's MQTT password. Download or copy it to keep it safe.' and a 'Download Credentials -' button.

<https://fr.mathworks.com/help/thingspeak/use-desktop-mqtt-client-to-publish-to-a-channel.html>



NODE MQTT POUR THINGSPEAK : IDENTIFICATION AU SERVEUR

Configuration du node MQTT IN (Node-Red)

The image shows the configuration interface for the 'mqtt in' node in Node-Red. The 'Name' field is set to 'ThingSpeak'. Under the 'Security' tab, the 'Username' field contains 'MzE...' and the 'Password' field contains '.....'. Both fields are highlighted with red boxes. Red arrows point from these fields to the 'MQTT Credentials' dialog box and the 'Write API Key' form on the right.

The 'MQTT Credentials' dialog box provides instructions to use MQTT credentials for publishing and subscribing to ThingSpeak channels. It includes fields for 'Client ID', 'Username', and 'Password'. The 'Username' field is highlighted with a red box. A warning message states: 'ThingSpeak does not store a copy of your device's MQTT password. Download or copy it to keep it safe.' A 'Download Credentials' button is visible at the bottom.

IoT Micro::bit

Channel ID: 1672283
Author: tonnerre
Access: Public

Démonstration de visualisation de données IoT

Password = Write API Key

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Write API Key

Key L...

Help

API keys enable y...
keys are auto-gen...

API Keys S...

NODE MQTT POUR THINGSPEAK LE TOPIC



The screenshot shows the Thingspeak IoT Micro:bit interface. On the left, the 'Properties' panel is visible with the following settings:

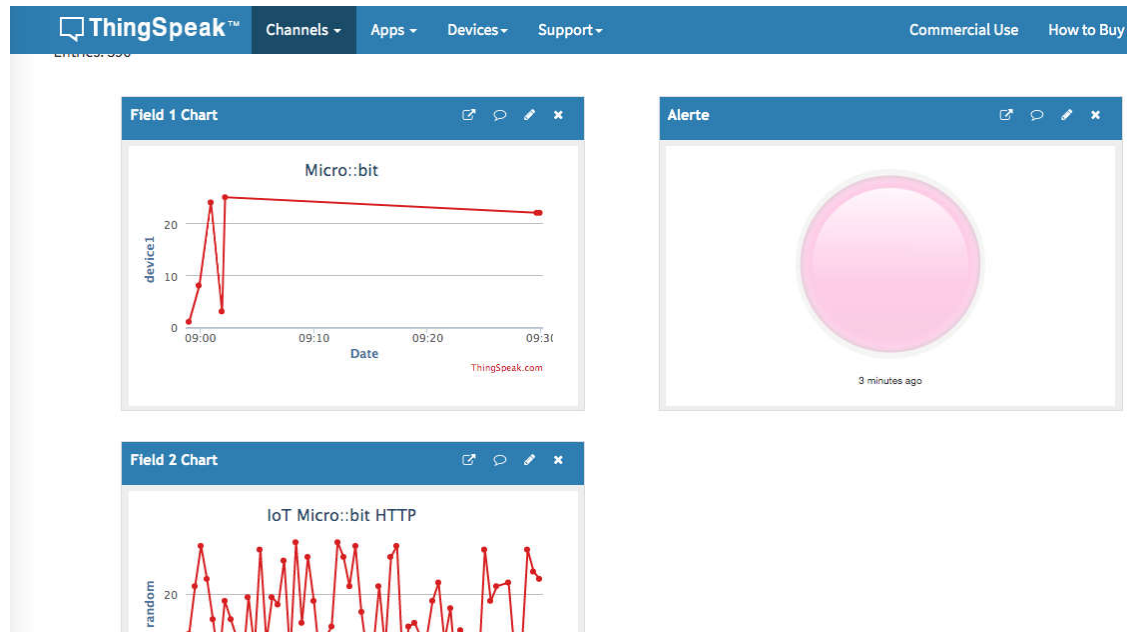
- Server: ThingSpeak
- Topic: channels/1672283/publish/fields/field1/L'
- QoS: [Dropdown]
- Retain: [Dropdown]
- Name: MQTT Thingspeak

On the right, the 'IoT Micro:bit' channel page is shown. The 'Channel ID' is 1672283, the author is 'tonnerre', and the access is 'Public'. The 'Write API Key' section is active, showing a 'Key' field with a value starting with 'L' and ending with '95'. A red arrow labeled 'Write API Key' points from the 'Key' field to the 'Topic' field in the Properties panel. A blue arrow labeled '<channelID>' points from the 'Channel ID' field to the '<channelID>' placeholder in the 'Topic' field.

topic = channels/<channelID>/publish/fields/field<fieldnumber>/<apikey>



UTILISATION DU CLOUD THINGSPEAK EN HTTP





TUTO YOUTUBE : ENVOI DES DONNÉES AVEC HTTP

Architecture IoT (4) : Envoyer et visualiser les données sur le cloud ThingSpeak par requêtes HTTP.



<https://youtu.be/rwDsE9auKRl>



NODE HTTP POUR THINGSPEAK (GET)

Edit http request node

Delete Cancel Done

Properties

Method: GET

URL: `https://api.thingspeak.com/update?api_key=L[redacted]&field2={{payload}}`

Append msg.payload as query string parameters

Enable secure (SSL/TLS) connection

Use authentication

Enable connection keep-alive

Use proxy

Return: a UTF-8 string

Name: send HTTP Thingspeak

Format « moustache » : insère dynamiquement le payload du message qui arrive dans le node HTTP

Le field associé à la donnée (un des 8 fields du channel)

Write API Key

Key: L[redacted]35

IoT Micro::bit

Channel ID: 1672283
Author: tonnerre
Access: Public

Démonstration de visualisation de données IoT

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Help

API keys enable y
keys are auto-gen

API Keys S



NODE HTTP POUR ADAFRUIT (POST)

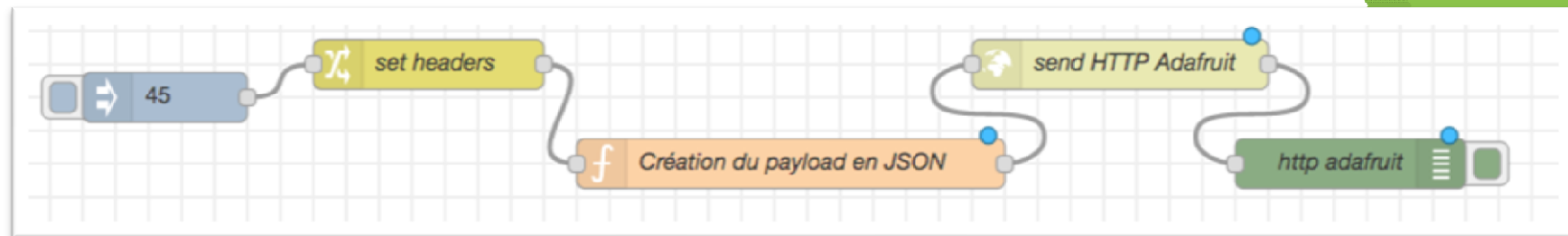
- ▶ Reprendre les informations de l'API pour curl
<https://io.adafruit.com/api/docs/?shell#create-data>
- ▶ Requête POST
 - ▷ Le **Header** contient l'**API KEY**
 - ▷ Une donnée : body contient du texte
 - ▷ Une donnée + localisation : body contient du JSON

```
curl -H "Content-Type: application/json" -d '{"value": 42, "lat": 23.1, "lon": "-73.3"}' -H "X-AIO-Key: {io_key}"  
https://io.adafruit.com/api/v2/{username}/feeds/{feed_key}/data
```





NODE HTTP POUR ADAFRUIT : FLOW EXEMPLE





NODE HTTP POUR ADAFRUIT : MISE EN ŒUVRE (1)

- ▶ Node « Change » pour créer le header et le body de la requête

Adafruit Active Key

Content-Type

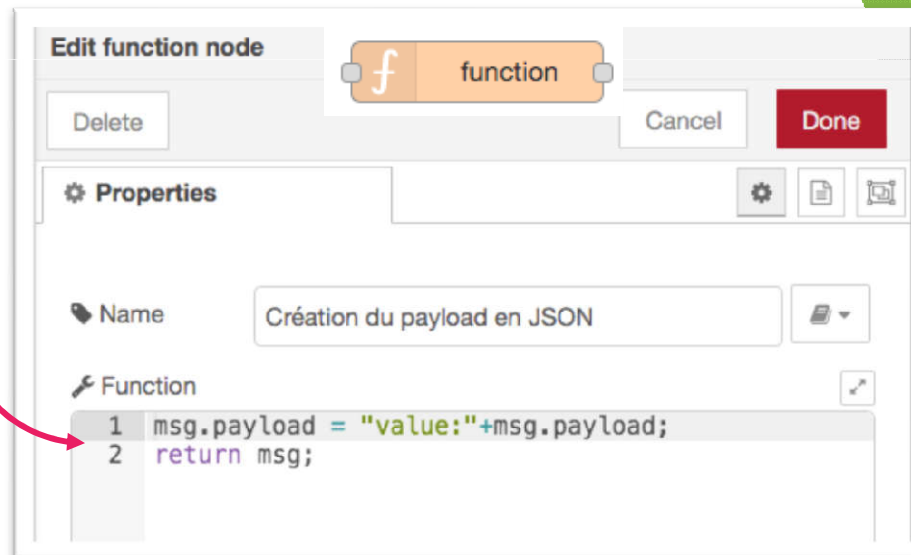
The screenshot shows the configuration of a 'change' node in Node-RED. The node is titled 'change: 2 rules'. It has a 'Name' field set to 'Name'. Under the 'Rules' section, there are two rules defined:

- Rule 1: Action 'Set', Target 'msg.headers.X-AIO-Key', Value 'aio_g...'. A red box highlights the value, and a red arrow points from the 'Adafruit Active Key' label to it.
- Rule 2: Action 'Set', Target 'msg.headers.Content-Type', Value 'application/json'. A red box highlights the value, and a red arrow points from the 'Content-Type' label to it.



NODE HTTP POUR ADAFRUIT : MISE EN ŒUVRE (2)

- ▶ Node « fonction » pour créer le body de la requête au format JSON à partir du payload (nombre)



Concaténation Javascript