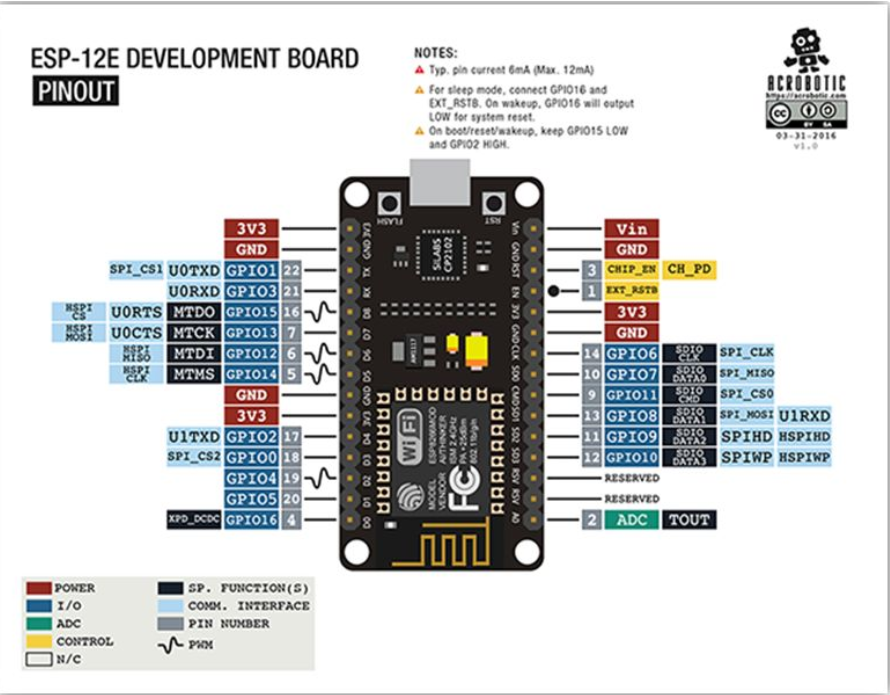
# ESP8266

L’ESP 8266 est un circuit integré à Microcontrôleur (comme le FRKM25Z) intégrant un service WIFI.  
On peut coder dessus en plusieurs langages mais ce qui nous interesse dans ce projet c’est d’utiliser l’IDE alternatif à Arduino Plateforme IO compatible avec celui-ci.

Pour en savoir plus consultez :

La doc : <https://arduino-esp8266.readthedocs.io/>

<https://ncd.io/weather-report-thingspeak-mqtt-ifttt-applets/>

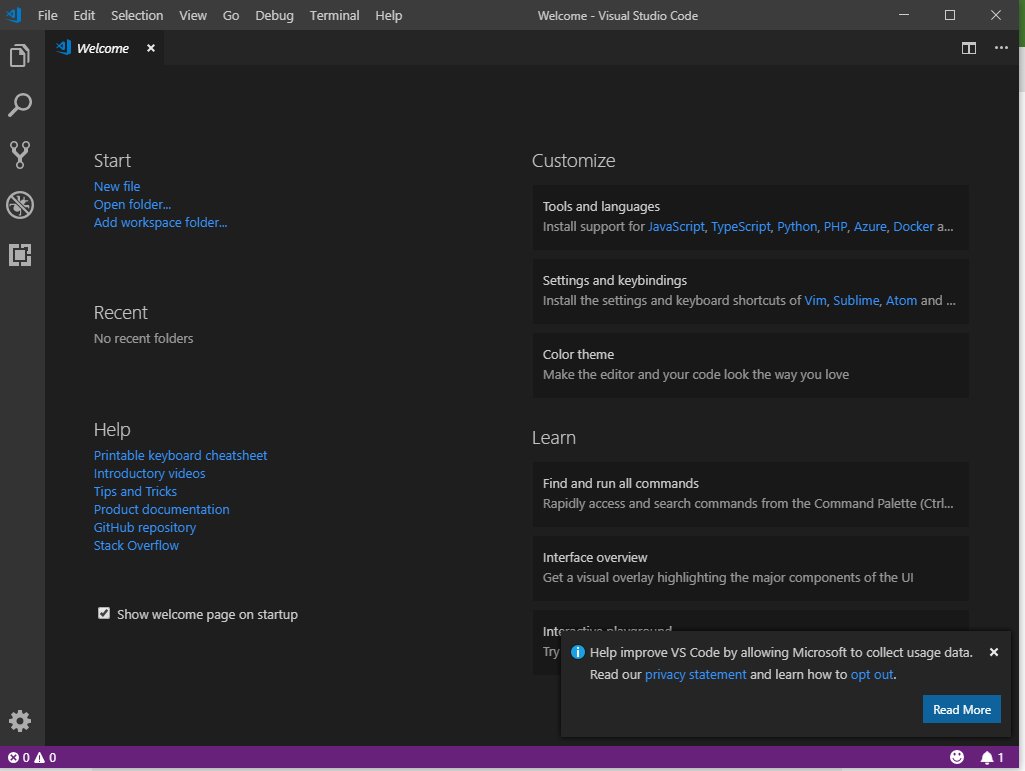


Avant de commencer, deux outils sont necessaires pour poursuivre ce projet :

* VS Code
* Plateform IO
* Thingspeak

**Installation de VS code : https://code.visualstudio.com/**

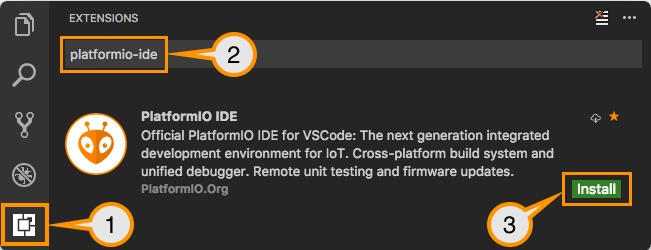
[Visual Studio Code](https://code.visualstudio.com) (VSC par la suite) est un éditeur de code open-source, gratuit et multi-plateforme (Windows, Mac et Linux), développé par Microsoft, à ne pas confondre avec Visual Studio, l'IDE propriétaire de Microsoft. Principalement conçu pour le développement d'application avec [JavaScript](https://edutechwiki.unige.ch/fr/JavaScript), [TypeScript](https://www.typescriptlang.org/) et [Node.js](https://edutechwiki.unige.ch/fr/Node.js), l'éditeur peut s'adapter à d'autres types de langages grâce à un système d'extension bien fourni.



**Installation de Plateform IO :**

**PlatformIO IDE est un environnement de développement qui peut se substituer à l’IDE Arduino traditionnel. PlatformIO prend en charge plus de 400 cartes de développement des principaux micro-contrôleurs (Atmel AVR, Espressif ESP8266 et ESP32, STMicro STM32, Teensy…).** Il s’agit d’une Exstension, utilisable sur des logiciels de developpement tel que VS Code, installé precedemment.

source : https://platformio.org/install/ide?install=vscode



**1 – Appuyer sur le bouton extensions**

**2 – Checher flarformio-ide dans la barre de recherche**

**3 – Installer** 

**Nouveau Projet et firmway ESP8266 :**

**Source :** [**https://projetsdiy.fr/bien-demarrer-platformio-ide-arduino-esp8266-esp32-stm32/**](https://projetsdiy.fr/bien-demarrer-platformio-ide-arduino-esp8266-esp32-stm32/)

Une fois l’extension installée, selectionner plateformIO et appuyer sur New Project :

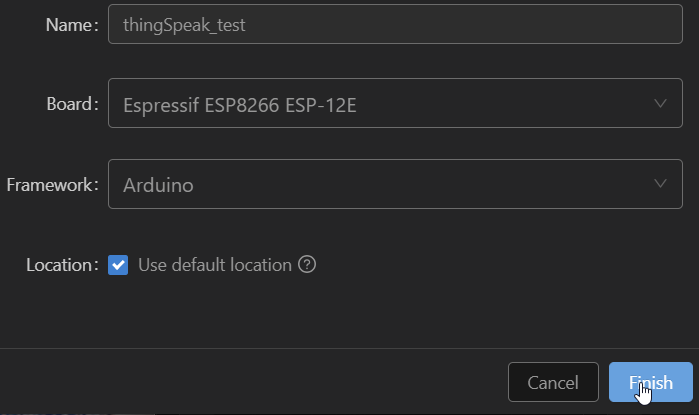


Remplir les informations demandées :

Nom : thingSpeak\_test

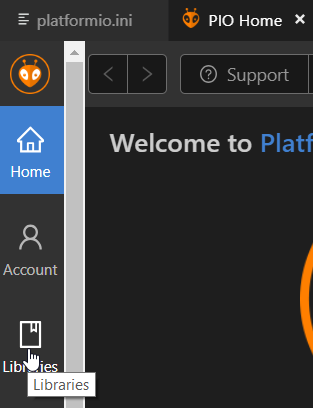
Plateform (Board) : ESP8266

Framework : Arduino

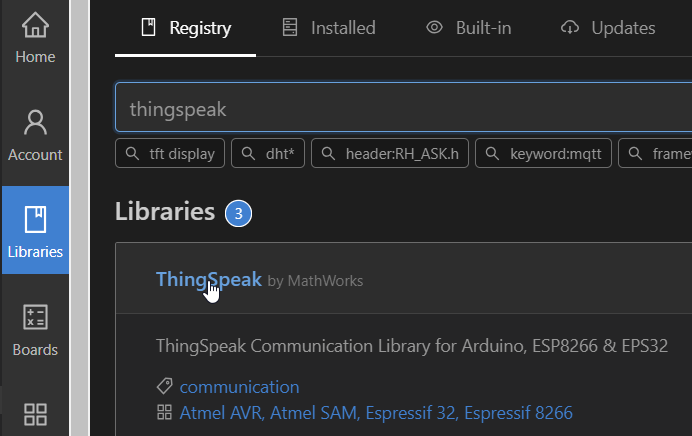


Une fois validé, l’extension crée ensuite les arborescences. La première Initialisation se fait, et l’extension se relance, il se peut qu’ils vous demandent de faire des mises à jour, faites-les.

Il faut ensuite telecharger les libraires pour l’utiliser sur thingspeak.  
Selectionner librairies.



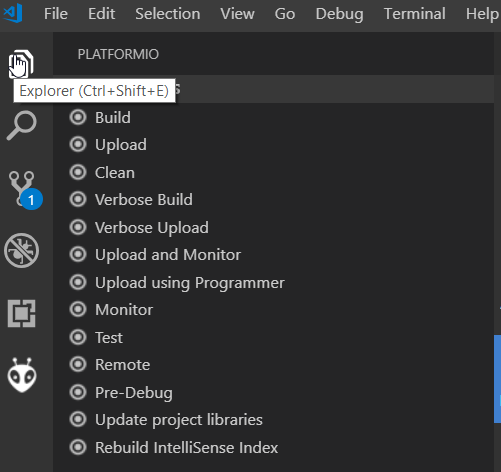
Cherchez la libraire thingspeak dans la barre de recherche, selectionnez et installez.





**Premier programme et Visualisation sur thingspeak**

Dans la barre de recherche de VS CODE, appuyer sur explorer.



Dans platformio Explorer, on ajoute lib\_deps = thingspeak ce qui permet de fournir le projet et d’avoir l’installation automatique de la bibliothèque et on build.

[env:esp12e]

platform = espressif8266

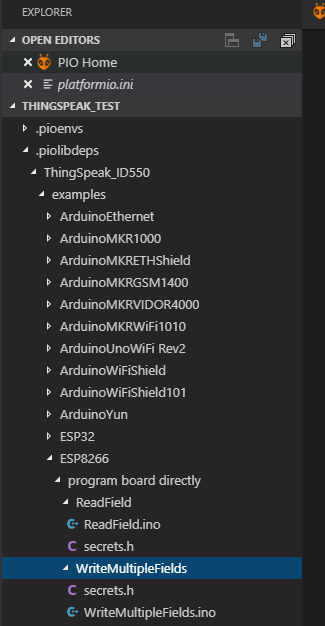
board = esp12e

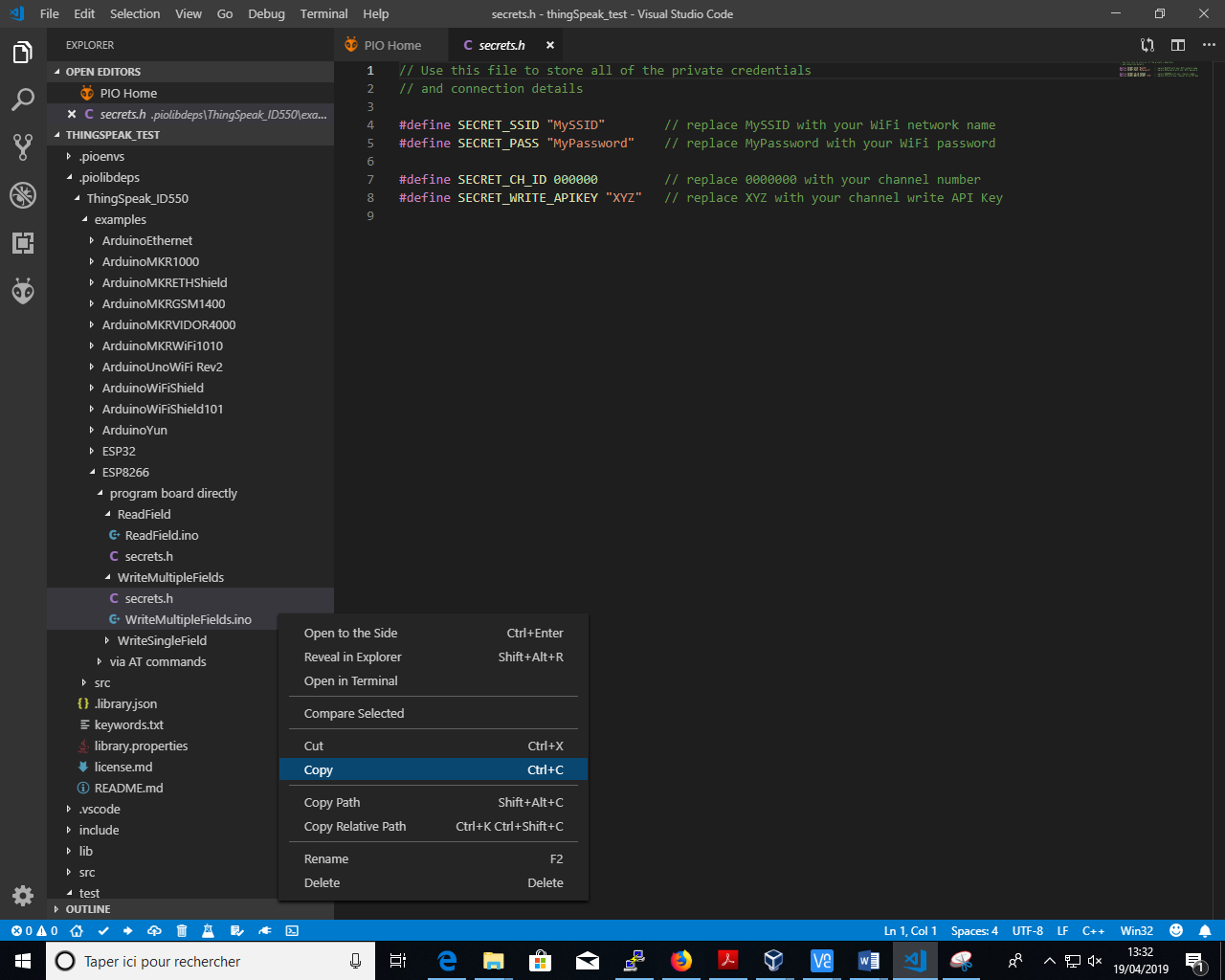
framework = arduino

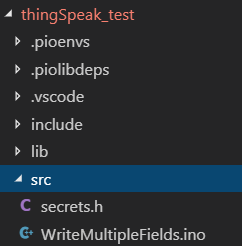
lib\_deps =

thingspeak

Dans pilbdeps/thingspeak/examples/esp8266/programboarddirectly/readfield/writemultiplefields



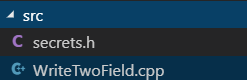
Copier les 2 fichiers du projet dans votre répertoire src : 



On renomme le fichier writemultiplefields.ino end writetwofields.cpp

Il y’aura des erreurs vous disant que le fichier n’est pas valide, les ignorer.

On peut suppimer le main.cpp



Avant de modifier le fichier secrets.h, assurez vous de connaitre le SSID et le mot de passe de votre wifi,

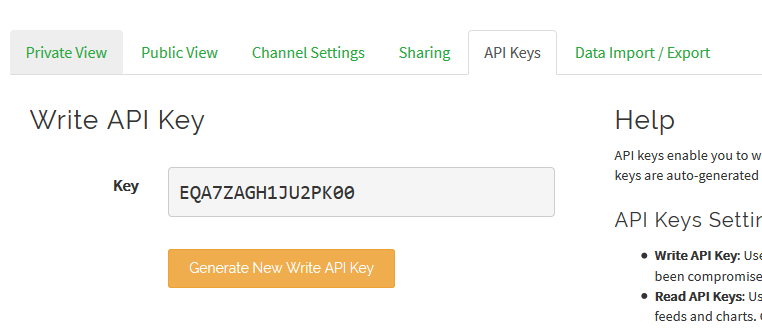
Sinon créer un hotspot wifi à l’aide de votre télephone.

SSID : stagelinux

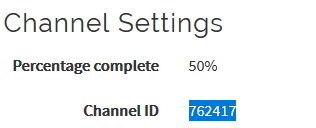
Passwd : 123456789

Créer un compte thingspeak

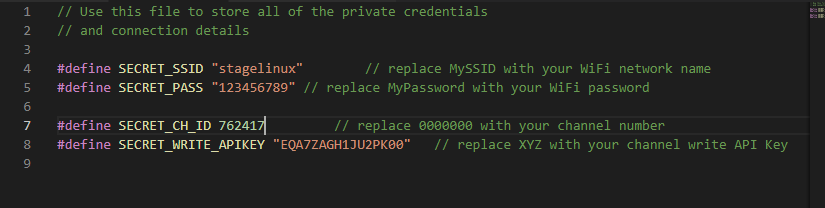
Créer un channel et recuperer la clé api :



Recuperer aussi l’id du channel que vous avez crée



Dans secrets.h, remplacez toutes les informations necessaires :



Le programme, genere 2 nombres, le premier commence à 0 et compte jusqu’à 100,

Le deuxième est un nombre aleatoire generé par une fonction.

On supprime le 3eme et le 4 eme.

On compare ensuite les deux nombres et on affiche qui sont les plus grands :

#include "ThingSpeak.h"

#include "secrets.h"

#include <ESP8266WiFi.h>

char ssid[] = SECRET\_SSID; // your network SSID (name) on recupere de secrets qu'on a modifié

char pass[] = SECRET\_PASS; // your network password pareil qu'en haut

int keyIndex = 0; // your network key Index number (needed only for WEP)

WiFiClient client;

unsigned long myChannelNumber = SECRET\_CH\_ID; // toujours recuperer de secrets.h

const char \* myWriteAPIKey = SECRET\_WRITE\_APIKEY;

// Initialize our values(initialisation des variables)

int number1 = 0;

int number2 = random(0,100); // nombre au hasard

String myStatus = "";

void setup() {

Serial.begin(115200); // Initialize serial

WiFi.mode(WIFI\_STA);

ThingSpeak.begin(client); // Initialize ThingSpeak

}

void loop() {

// On se connecte au WIFI

if(WiFi.status() != WL\_CONNECTED){

Serial.print("Attempting to connect to SSID: ");

Serial.println(SECRET\_SSID);

while(WiFi.status() != WL\_CONNECTED){

WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open or WEP network

Serial.print(".");

delay(5000);

}

Serial.println("\nConnected.");

}

// on associe les champs thingspeak au nombre generés

ThingSpeak.setField(1, number1);

ThingSpeak.setField(2, number2);

// figure out the status message

if(number1 > number2){

myStatus = String("field1 is greater than field2");

}

else if(number1 < number2){

myStatus = String("field1 is less than field2");

}

else{

myStatus = String("field1 equals field2");

}

// set the status

ThingSpeak.setStatus(myStatus);

// write to the ThingSpeak channel

int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

if(x == 200){

Serial.println("Channel update successful.");

}

else{

Serial.println("Problem updating channel. HTTP error code " + String(x));

}

// change the values

number1++; // compteur

if(number1 > 99){

number1 = 0;

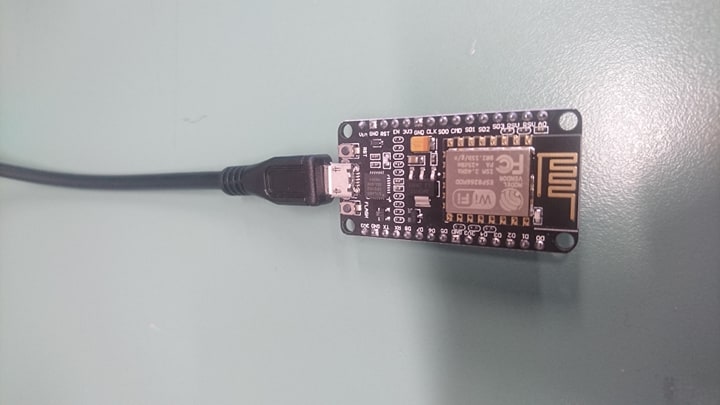
}

number2 = random(0,100); // nombre au hasard

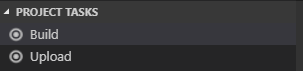
delay(20000); // Wait 20 seconds to update the channel again

}

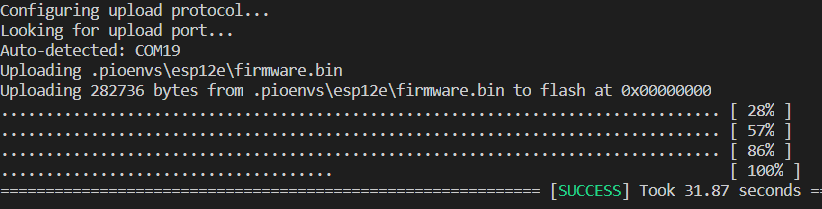
Connecté l’ESP8266 et upload trouvera le port com lié au microcontroleur automatiquement.



Dans project tasks :

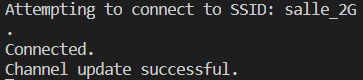


Build et upload



Ensuite 

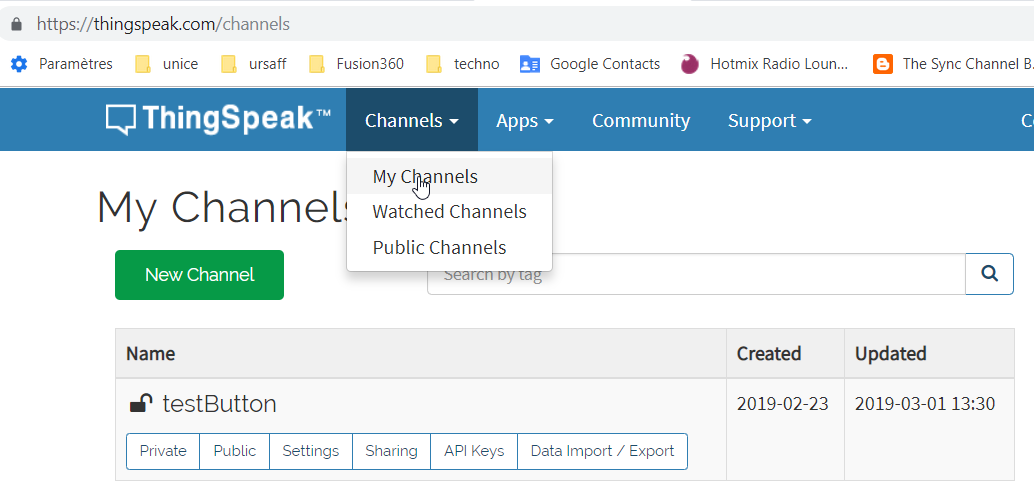
Vous pouvez aussi utiliser putty et terraterm. (Conseillé car vitesse 115200 est plus appropriée) CF en bas pour voir comment modifier la vitesse sur plateform IO.



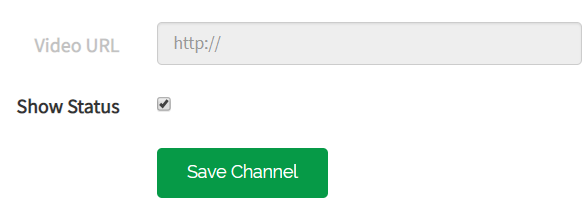
On envoie aussi le status…

**Visualisation sur thingspeak**

Maintenant sur thingspeak, on va parametrer pour visualier notre programme.

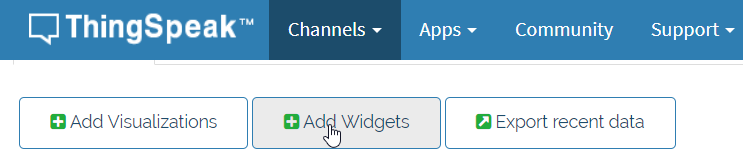


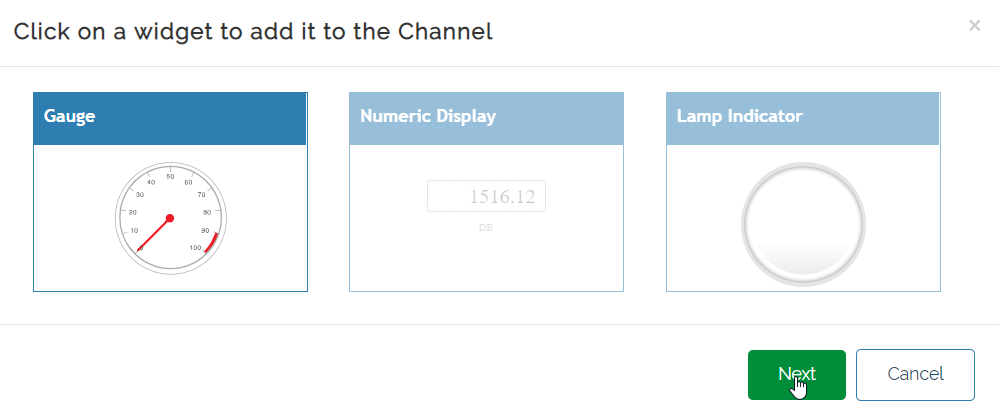


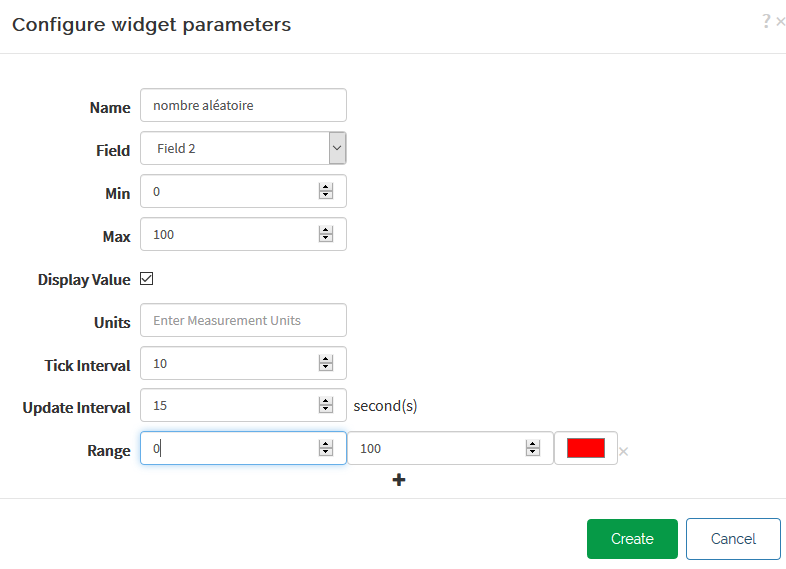


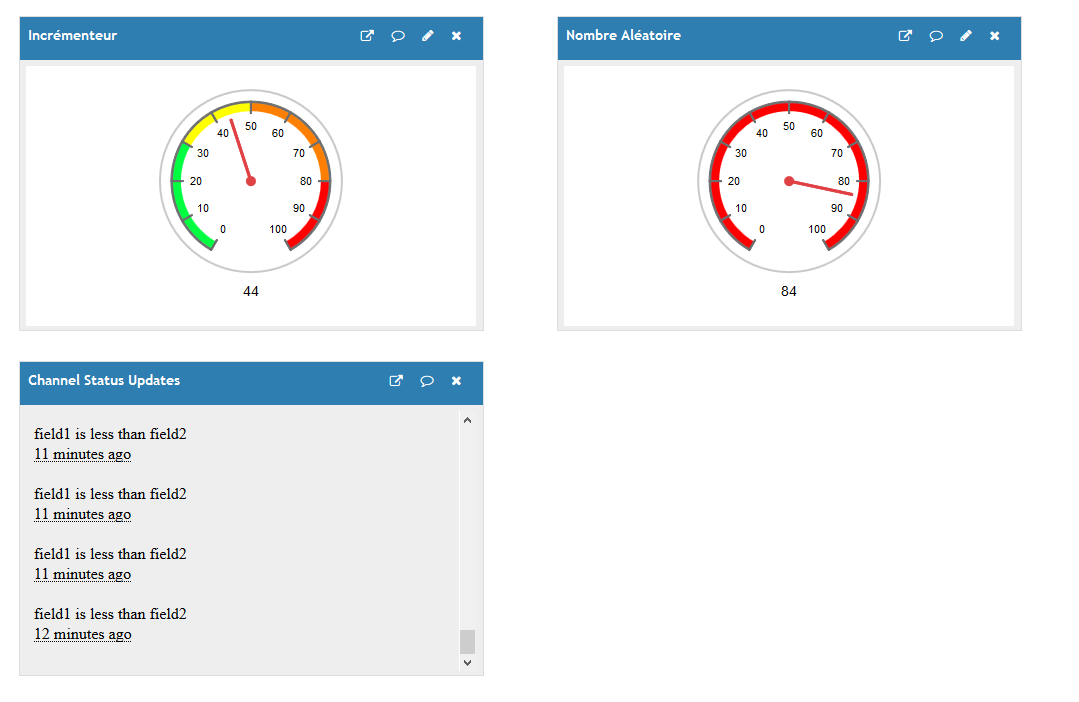


On ajoute 2 jauges, pour le nombre 1 et 2 : toutes les 20 secondes le nombre 1 s’incrémente de 1, et toutes les 20 secondes un nombre aleatoire est generé.

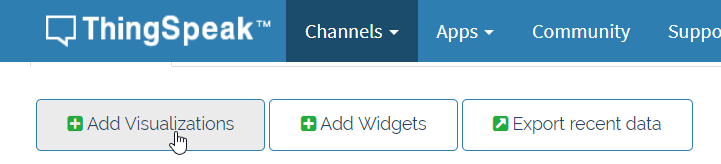


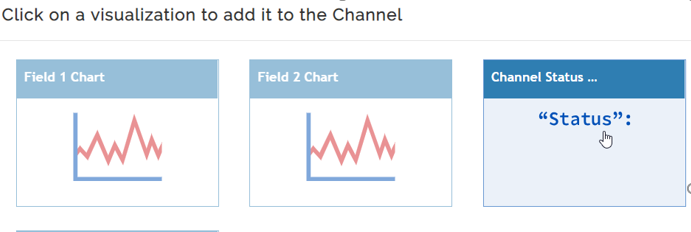




On peut sélectionner la couleur pour chaque intervalle : Ce qui donne avec les deux jauges : 

Si vous n’avez pas le status qui a été rajouté automatiquement :





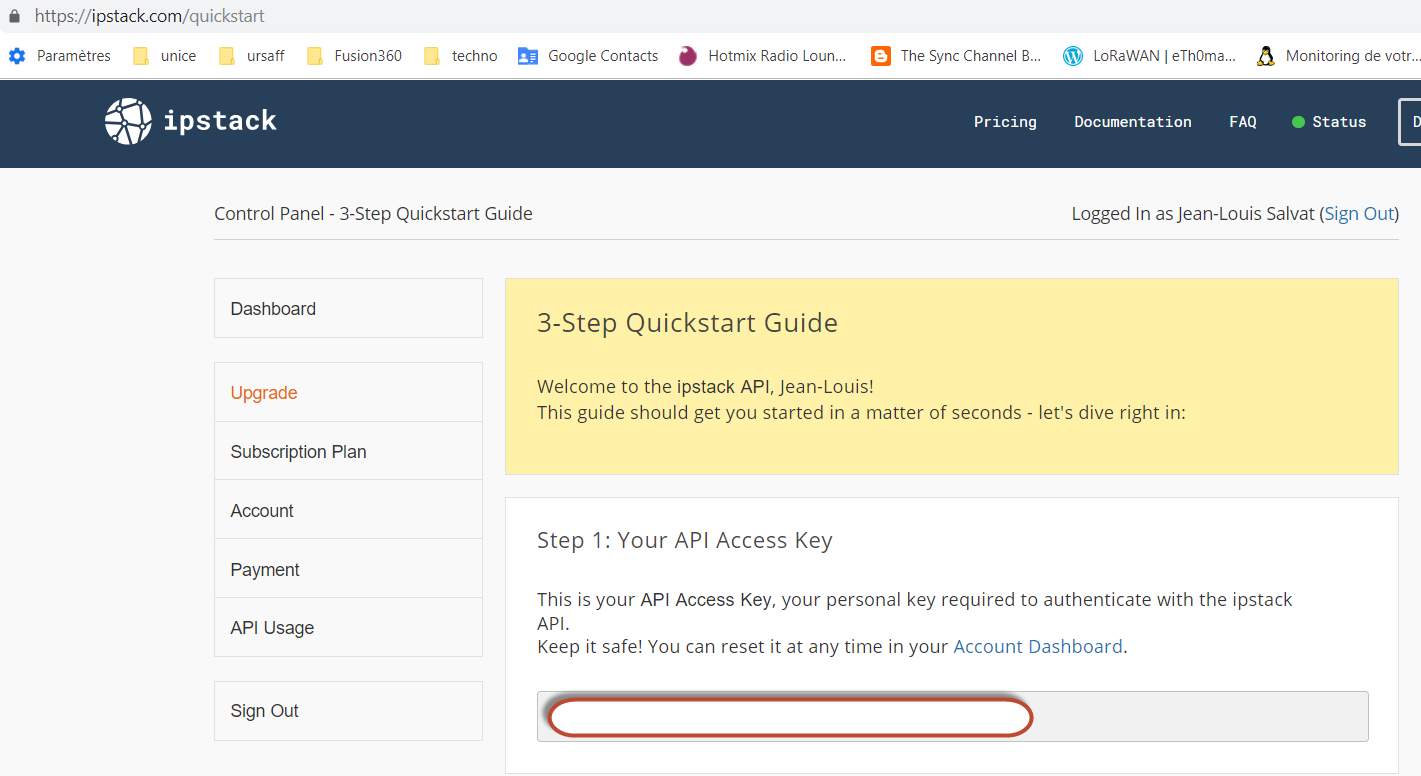
Nous allons maintenant essayer d’utiliser un service de géolocalisation :

Mais problème, google demande une carte bleue…

On va donc utiliser un service opensource :

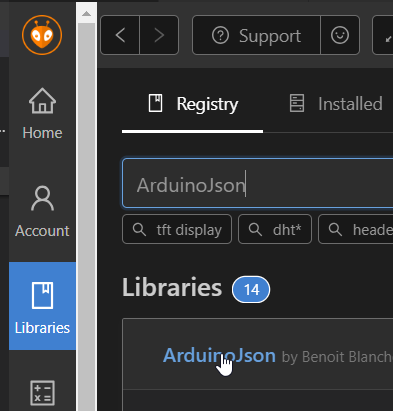
Créer un compte

<https://ipstack.com/> qui propose des clés API gratuites :



Gardez votre API précieusemment

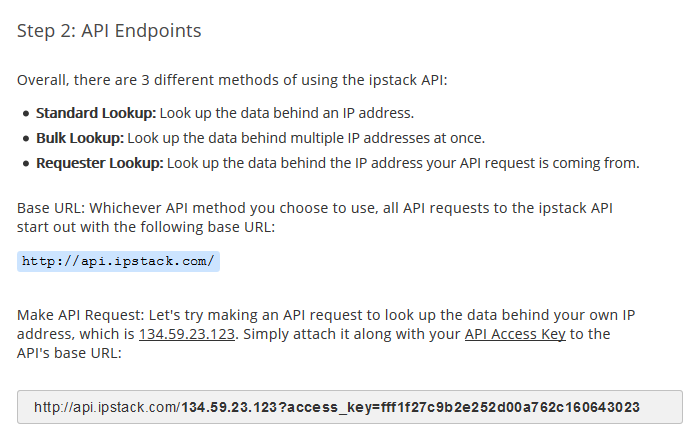
Une fois ceci fait, on installe la libraire AruinoJson, comme fait precedemment :

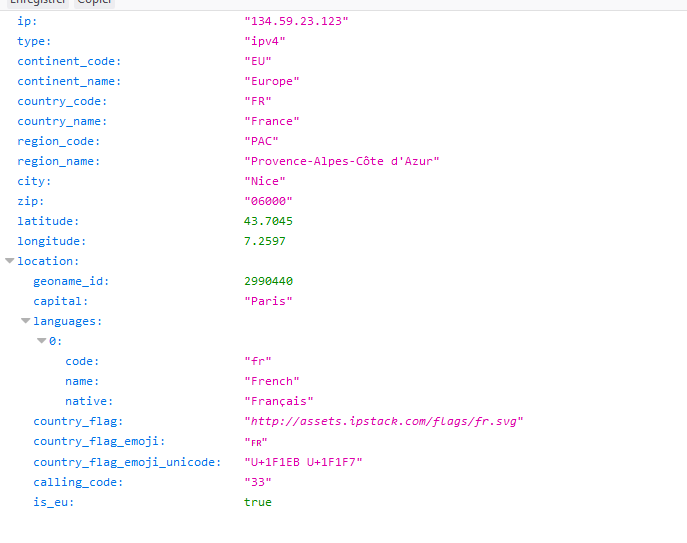




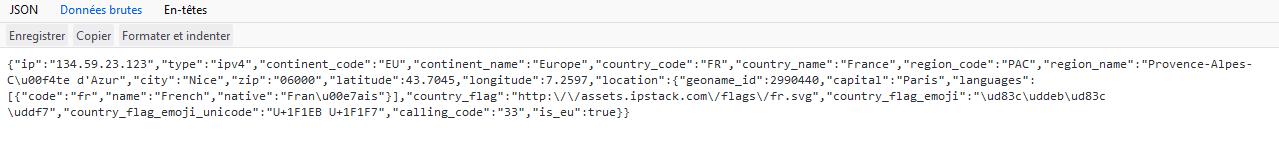
Ne pas oublier de build.

Pour tester l’api ipstack possede un service, il suffit d’aller dans STEP 2 API Endpoints, et de recopier l’adresse qu’il vous donne afin de voir si votre API marche bel et bien :





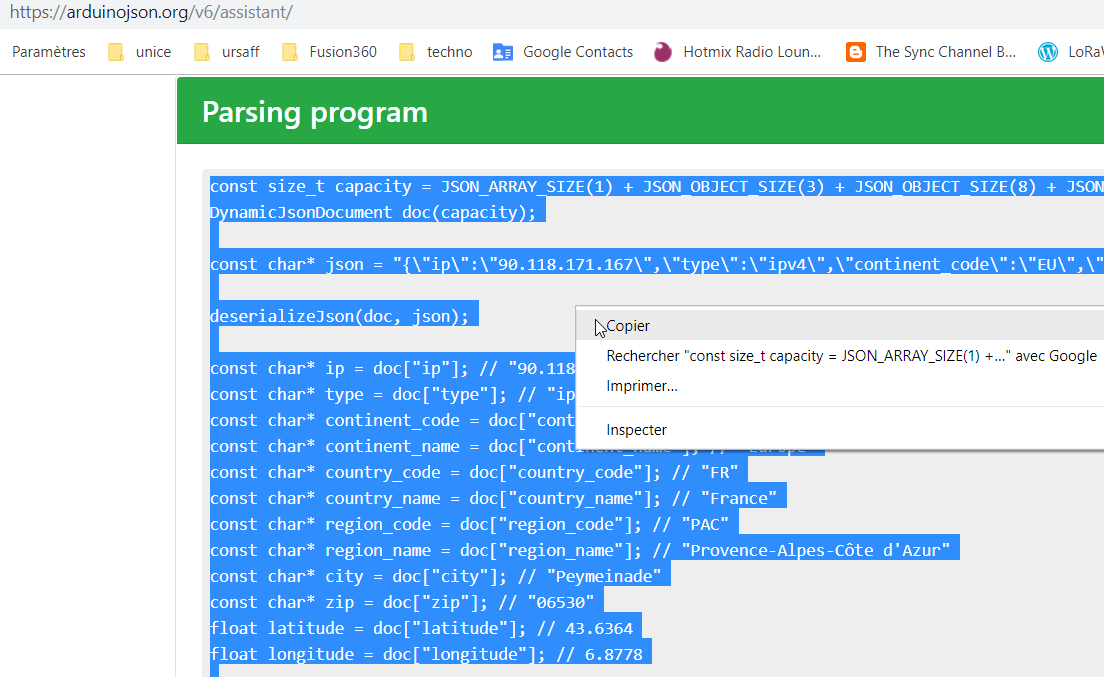
Appuyer sur données brutes :



Copier et allons tester sur le site : https://arduinojson.org/v6/assistant/

Coller les données dans l’input ici :





Quelques explications

<https://projetsdiy.fr/esp8266-client-web-exemples-communication-tcp-ip-esp8266wifi-esp8266httpclient/>

https://www.instructables.com/id/ESP8266-Parsing-JSON/

Bien on a maintenant le programme de Parsing, Il faut modifier le programme pour lancer une requete GET. ( cf lien 2 ).

Il existe plusieurs types de requête http, les plus utilisées étant la requête get et post. Le plus souvent ces requêtes sont déjà codées en fonction de ce que vous voulez faire dans des fonctions rangées dans un fichier .h, que vous pouvez aller recuperer et utiliser, cela sera fait notammant dans la suite avec une requête post pour twitter.

Nous allons cependant dans un premier temps coder la requête manuellement.

Après le loop de connexion wifi, On rajoute celui ci, mais avant cela il faut rajouter les laibrairies qu’on a importé dans notre programme :

#include <ESP8266HTTPClient.h>

#include <ArduinoJson.h>

Une fois ceci, fait et juste après la connexion wifi : (preter attention aux commentaires) !

if(WiFi.status() != WL\_CONNECTED){

Serial.print("Attempting to connect to SSID: ");

Serial.println(SECRET\_SSID);

while(WiFi.status() != WL\_CONNECTED){

WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open or WEP network

Serial.print(".");

delay(5000);

}

}

Serial.println("\nConnected.");

HTTPClient http; //Object of class HTTPClient création d’un objet de de class client http

http.begin("http://api.ipstack.com/134.59.23.123?access\_key=fff1f27c9b2e252d00a762c160643023"); //on indique sur quel url nous allons employer nos requêtes

int httpCode = http.GET(); // on récupere le code dans une variable http code, à l’aide de la requête get

//Check the returning code

if (httpCode > 0)  // verification

//Code Parsing à mettre ici

}

http.end(); //Close connection

// Requete GET

On colle ensuite notre programme de parsing dans le champ :

// On se connecte au WIFI

if(WiFi.status() != WL\_CONNECTED){

Serial.print("Attempting to connect to SSID: ");

Serial.println(SECRET\_SSID);

while(WiFi.status() != WL\_CONNECTED){

WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open or WEP network

Serial.print(".");

delay(5000);

}

Serial.println("\nConnected.");

HTTPClient http; //Object of class HTTPClient

http.begin("http://api.ipstack.com/134.59.23.123?access\_key=fff1f27c9b2e252d00a762c160643023");

int httpCode = http.GET();

//Check the returning code

if (httpCode > 0) {

//Code Parsing à mettre ici

const size\_t capacity = JSON\_ARRAY\_SIZE(1) + JSON\_OBJECT\_SIZE(3) + JSON\_OBJECT\_SIZE(8) + JSON\_OBJECT\_SIZE(13) + 470;

DynamicJsonDocument doc(capacity);

const char\* json = "{\"ip\":\"134.59.23.123\",\"type\":\"ipv4\",\"continent\_code\":\"EU\",\"continent\_name\":\"Europe\",\"country\_code\":\"FR\",\"country\_name\":\"France\",\"region\_code\":\"PAC\",\"region\_name\":\"Provence-Alpes-Côte d'Azur\",\"city\":\"Nice\",\"zip\":\"06000\",\"latitude\":43.7045,\"longitude\":7.2597,\"location\":{\"geoname\_id\":2990440,\"capital\":\"Paris\",\"languages\":[{\"code\":\"fr\",\"name\":\"French\",\"native\":\"Français\"}],\"country\_flag\":\"http://assets.ipstack.com/flags/fr.svg\",\"country\_flag\_emoji\":\"🇫🇷\",\"country\_flag\_emoji\_unicode\":\"U+1F1EB U+1F1F7\",\"calling\_code\":\"33\",\"is\_eu\":true}}";

deserializeJson(doc, json);

const char\* ip = doc["ip"]; // "134.59.23.123" //on range dans des variables

const char\* type = doc["type"]; // "ipv4"

const char\* continent\_code = doc["continent\_code"]; // "EU"

const char\* continent\_name = doc["continent\_name"]; // "Europe"

const char\* country\_code = doc["country\_code"]; // "FR"

const char\* country\_name = doc["country\_name"]; // "France"

const char\* region\_code = doc["region\_code"]; // "PAC"

const char\* region\_name = doc["region\_name"]; // "Provence-Alpes-Côte d'Azur"

const char\* city = doc["city"]; // "Nice"

const char\* zip = doc["zip"]; // "06000"

float latitude = doc["latitude"]; // 43.7045

float longitude = doc["longitude"]; // 7.2597

JsonObject location = doc["location"];

long location\_geoname\_id = location["geoname\_id"]; // 2990440

const char\* location\_capital = location["capital"]; // "Paris"

JsonObject location\_languages\_0 = location["languages"][0];

const char\* location\_languages\_0\_code = location\_languages\_0["code"]; // "fr"

const char\* location\_languages\_0\_name = location\_languages\_0["name"]; // "French"

const char\* location\_languages\_0\_native = location\_languages\_0["native"]; // "Français"

const char\* location\_country\_flag = location["country\_flag"]; // "http://assets.ipstack.com/flags/fr.svg"

const char\* location\_country\_flag\_emoji = location["country\_flag\_emoji"]; // "🇫🇷"

const char\* location\_country\_flag\_emoji\_unicode = location["country\_flag\_emoji\_unicode"]; // "U+1F1EB U+1F1F7"

const char\* location\_calling\_code = location["calling\_code"]; // "33"

bool location\_is\_eu = location["is\_eu"]; // true

}

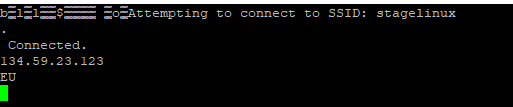
http.end(); //Close connection

}

Une fois qu’on a recuperé toutes nos données, on essaye d’afficher quelque unes pour voir si c’est ok :

Serial.println(ip);

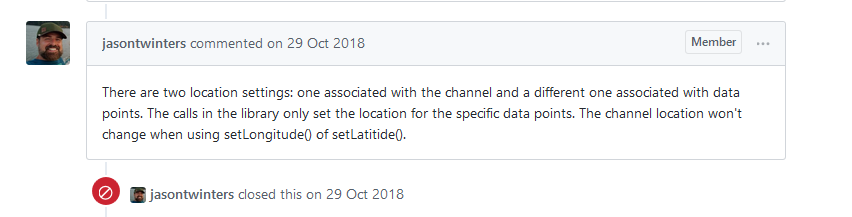
Serial.println(continent\_code);



Super ! maintenant il nous reste plus qu’à modifier le code pour l’envoyer à thingspeak :

Une fois nos informations stockées, nous allons maintenant chercher à l’envoyer à thingspeak pour afficher la carte avec notre position !

Cependant attention !

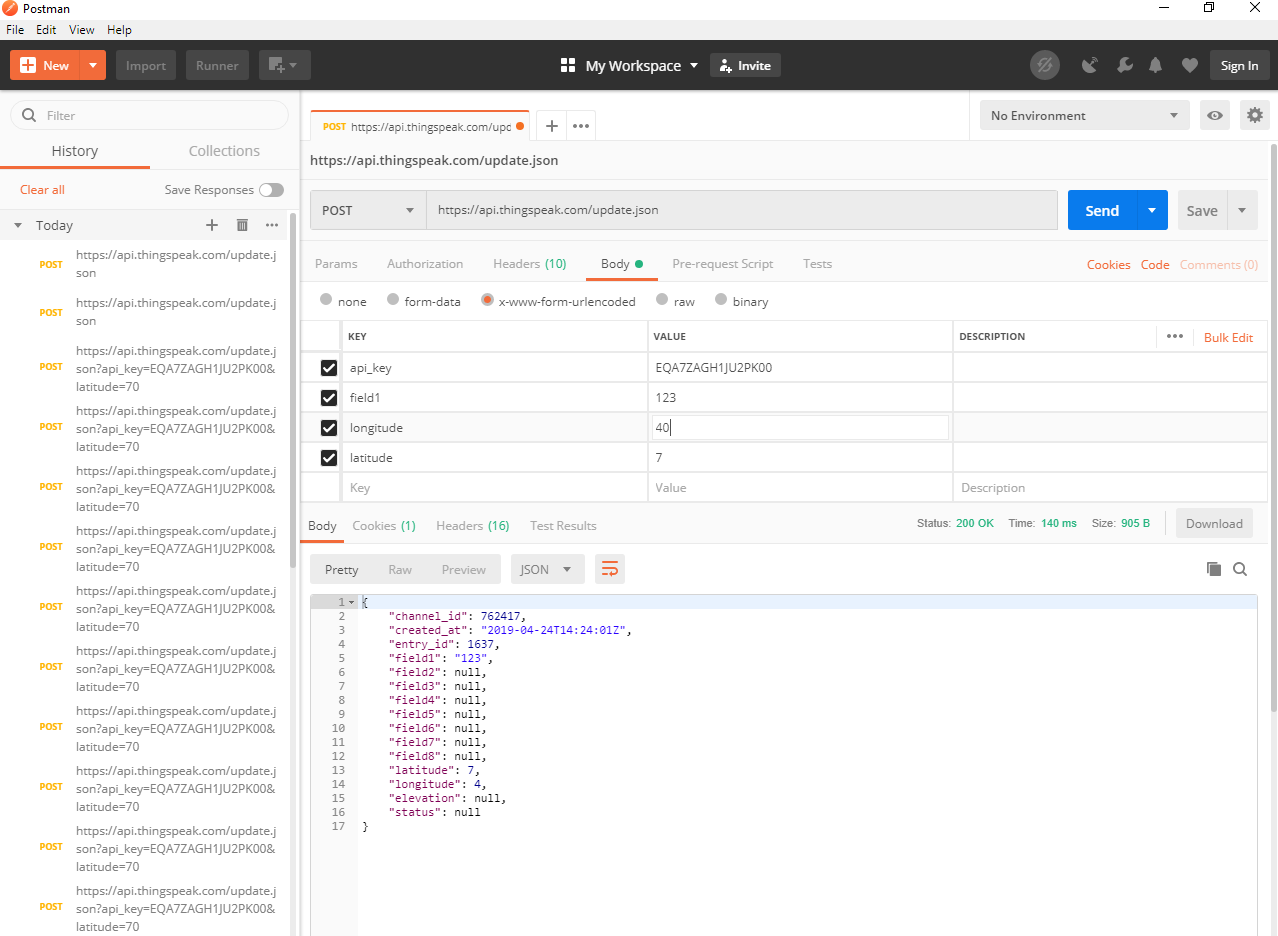




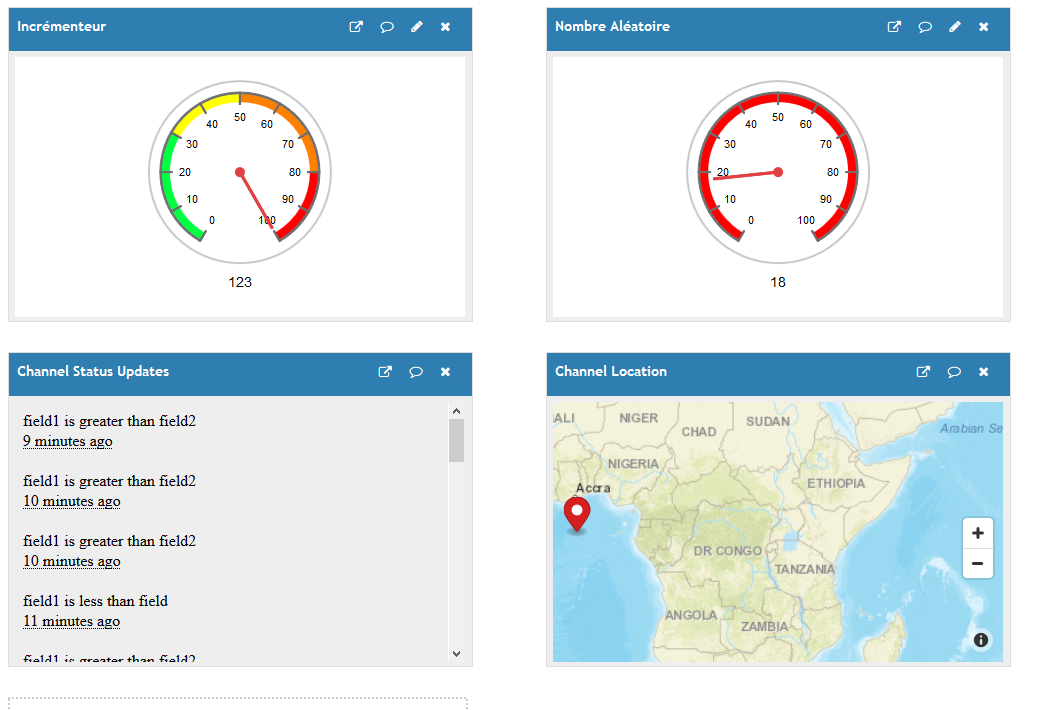
Ce qui veut dire qu’il ne faut pas utiliser le menu ci-dessus, il faudra le créer nous même :

https://blogs.mathworks.com/iot/2017/01/31/create-a-gps-tracker-with-matlab-mobile-and-thingspeak/

Afin de comprendre comment on écrit dans les champs, on test de faire une requête post à l’aide d’un logiciel : « postman » :



Le Field 1, se met bien à jour, mais le marqueur sur la carte n’a pas bougé restant toujours aux coordonnées : 0,0.



On a vu precedemment que l’on pouvait récuperer des données d’un site WEB à l’aide d’une requête GET.

Nous allons utiliser la même méthode ici afin de programmer un site web qui va récuperer nos données de localisations et les afficher sur une carte.

Pour ainsi faire nous allons avoir besoin d’utiliser du javascript. Nous allons donc telecharger et installer wordpress.

Mais avant, nous avons besoin d’un serveur http.

Pour avoir un serveur http, on va utiliser notre carte banana pi il faudra installer les librairies suivantes :

Suivre ce guide :

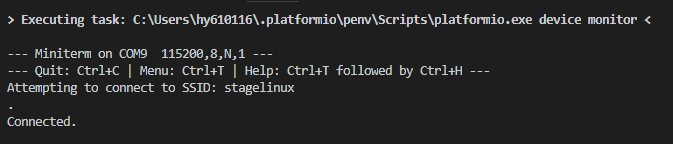
<https://raspbian-france.fr/installer-serveur-web-raspberry-lamp/>

Après que cela soit validé, On va tenter de generer un tweet sur un seuil.

Avant, nous allons changer la vitesse du terminal inclut dans le logiciel pour ne pas avoir à utiliser putty et terraterm, pour cela il existe une commande à taper sur plateform ini :

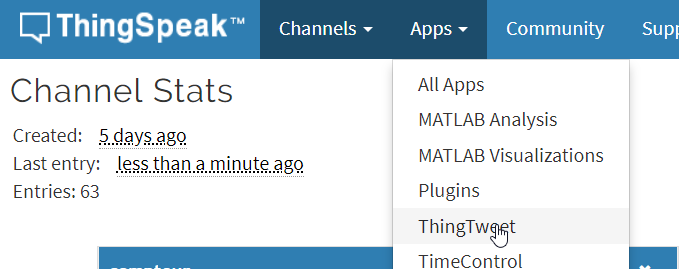
https://docs.platformio.org/en/latest/projectconf/section\_env\_monitor.html

monitor\_speed = 115200

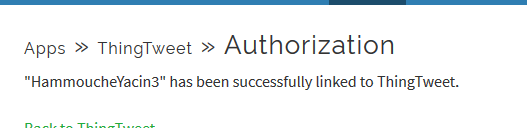


(Ctrl T et H sur le terminal pour plus d’informations).

Si vous n’avez pas de compte tweeter, en créer un.

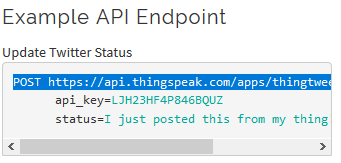


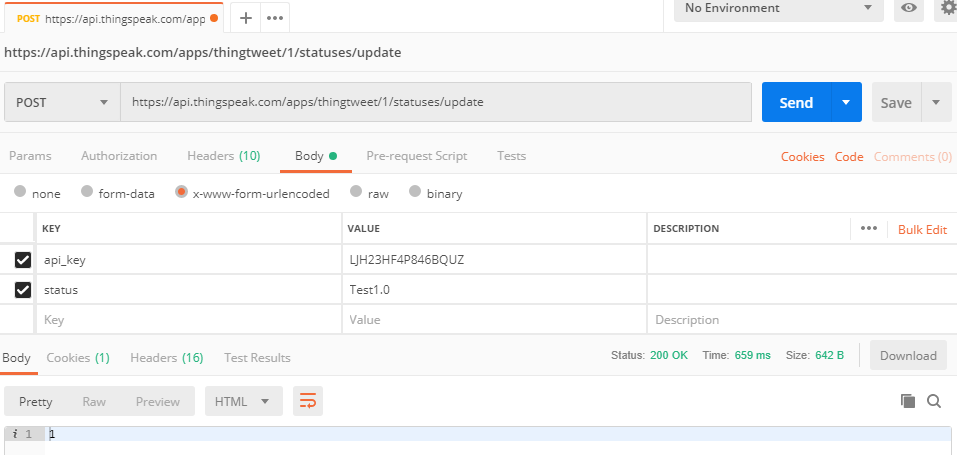
Liez votre compte, à thingspeak :



Dans un premier temps, nous testons l’exemple que propose le site.

<https://techtutorialsx.com/2016/07/21/esp8266-post-requests/>

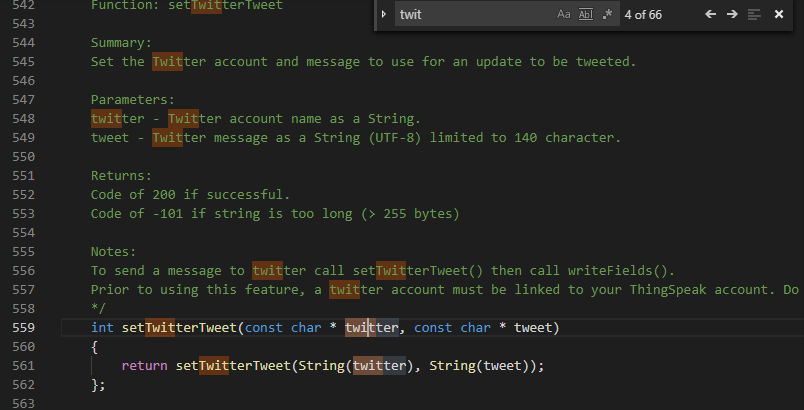






On recherche ensuite dans les librairies de fonctions, celles qui sont liées à twitter : (ctrl +f)

Et ctrl + clic sur n’importe quelle fonction pour aller dans le .h, sinon le chercher manuellement.



On crée donc deux constantes, une qui va porter le nom de compte et l’autre le tweet, et on les initialise :

String compte ="";

String tweet="";

On les remplit ensuite avec le nom de compte, ainsi que le tweet que l’on veut afficher :

Il suffit ensuite d’appeler la fonction setTwitterTweet, qui s’occupe de faire le post.

compte = String("HammoucheYacin3");

tweet = String("Test1.2");

ThingSpeak.setStatus(myStatus);

ThingSpeak.setTwitterTweet(compte,tweet);



On peut alors faire en sorte de lancer un tweet dès que le nombre 1 = 5, on va consideré que c’est notre seuil.

ThingSpeak.setStatus(myStatus);

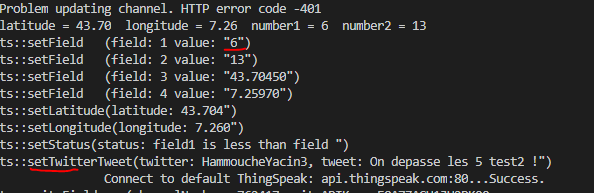
if(number1>5){

tweet = String("On depasse les 5 test2 !");

ThingSpeak.setTwitterTweet(compte,tweet);

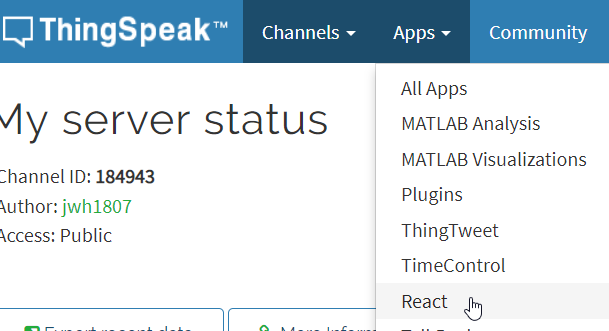
}//on met la fonction dans le if pour ne pas envoyez de tweets dans le vide

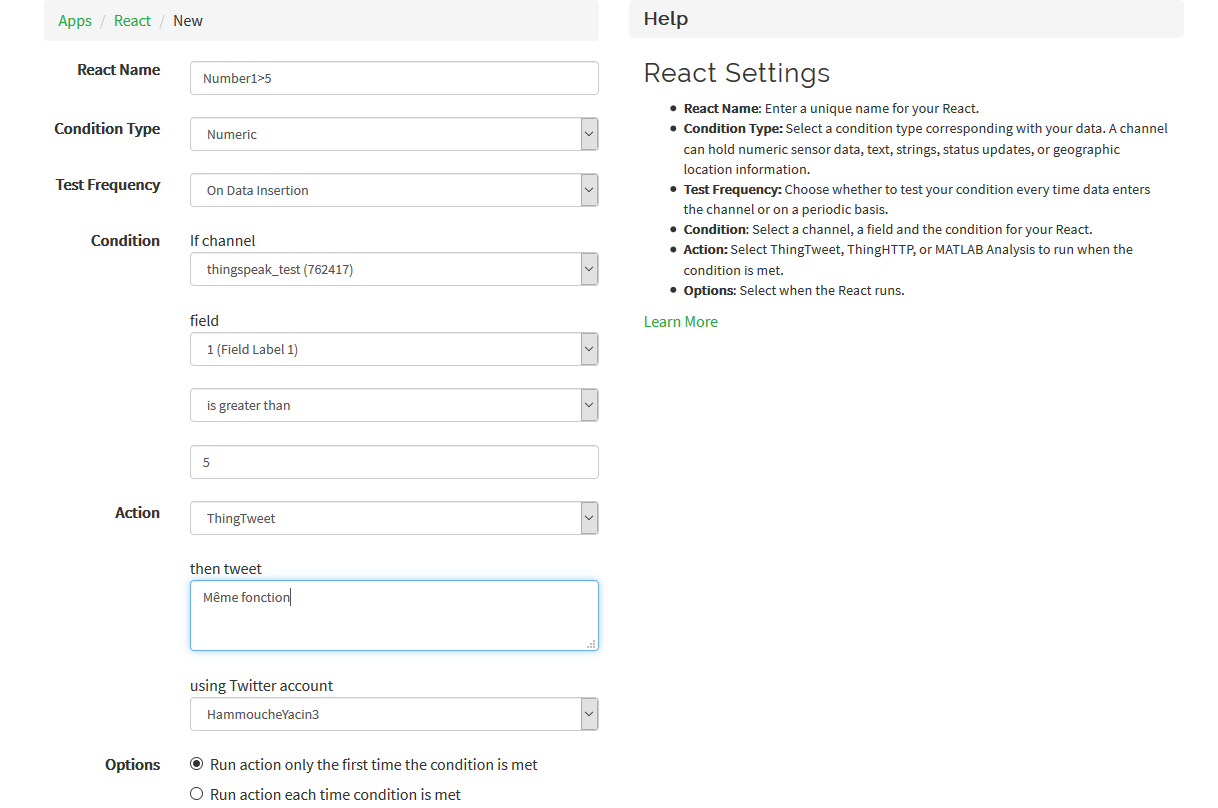
Et on test :

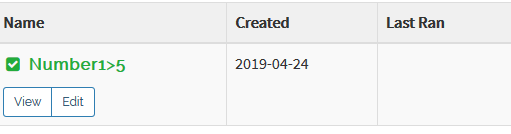




Pour ce genre de seuils, on peut aussi utiliser React de ThingSpeak :







(J’ai remplacé par 65 pour eviter de rebooter).

