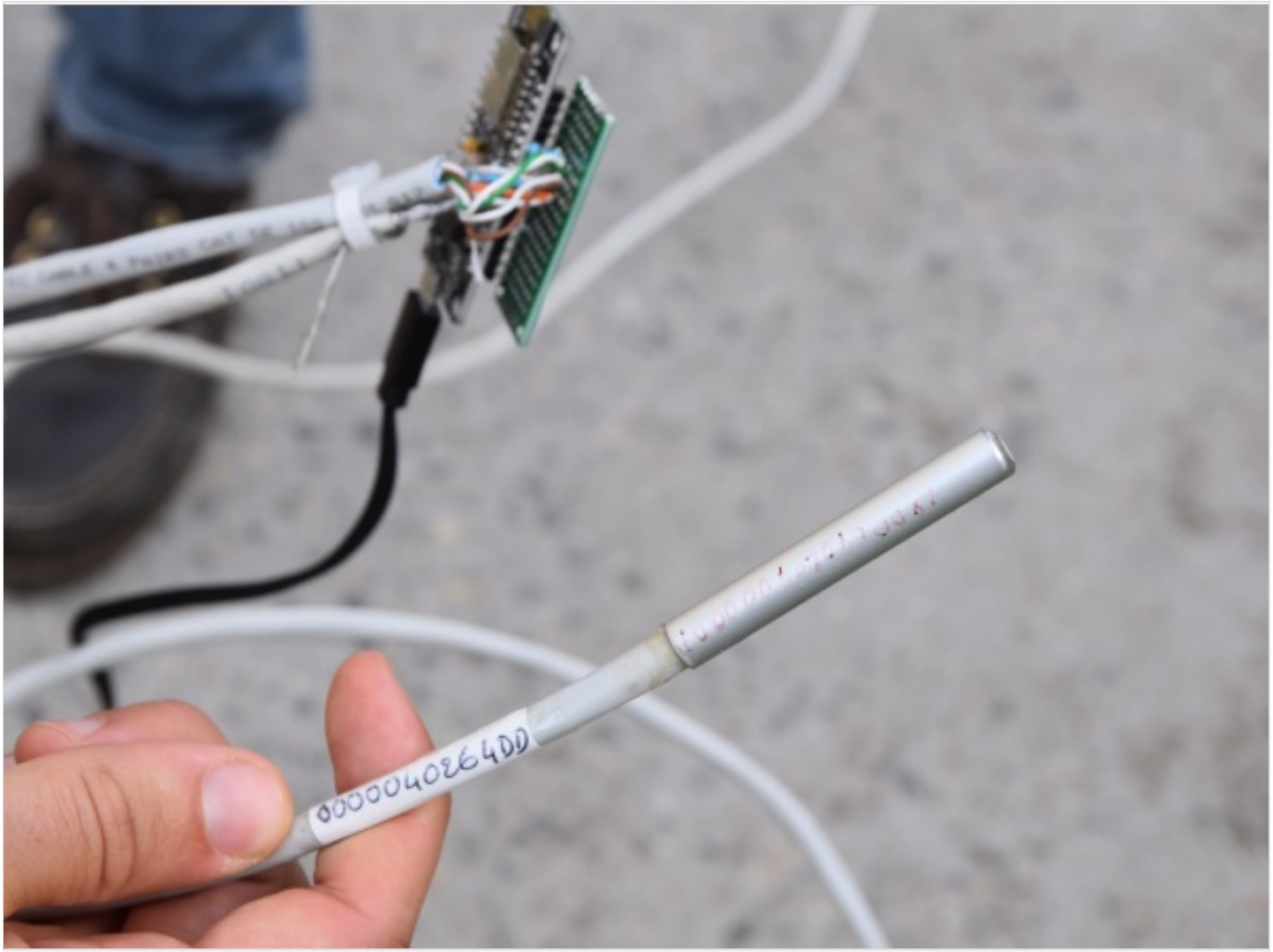
Capteur - Enregistreur de température

Low-tech Lab



https://wiki.lowtechlab.org/wiki/Capteur_-_Enregistreur_de_temp%C3%A9rature

Dernière modification le 21/04/2024

Difficulté **Moyen**

Durée **5 heure(s)**

Coût **25 EUR (€)**

Description

Ce tutoriel présente comment réaliser un système de mesure, d'enregistrement et de visualisation de données de température.

Sommaire

Sommaire

Introduction

Avec la hausse du prix de l'énergie, vous recherchez probablement un moyen économique de mieux isoler votre habitation pour plus de confort pendant les périodes hivernales. Ce qui est proposé ici n'est pas une solution miracle pour réparer une mauvaise isolation, mais juste une petite installation permettant de réaliser des mesures de température, que ce soit dans une pièce fermée, à l'extérieur ou bien même dans un liquide.

A priori, ce système n'est pas vraiment une low-tech. On pourrait même le comparer à un simple thermomètre. Mais la différence avec un thermomètre est que ce montage est capable d'enregistrer les températures qu'il capte au fur et à mesure.
En effet, les données de température cheminent du/des capteur(s) vers une carte électronique, qui les envoie elle-même vers un serveur appelé broker MQTT où s'échangent de nombreuses données en temps réel. Enfin, celles qui nous intéressent seront alors récupérables grâce à un programme.

Grace à ça, vous pourrez suivre en direct l'évolution de la température dans votre chambre, dans votre salle de bain, et plus encore.

Matériaux

- Un ou plusieurs capteurs de température du type DS18B20
- Une carte électronique ESP8266
- Une résistance électrique de 4,7 kΩ
- Une mini Breadboard
- Des fils électriques (une dizaine sera largement suffisant)

Outils

- Un câble USB micro B pour brancher l'ESP8266
- Un ordinateur avec :
  - Un logiciel capable de lire et d'exécuter des programmes en Python (comme par exemple VSCode ou Thonny)
  - Le logiciel Tasmotizer téléchargeable via le lien disponible dans la rubrique "Fichiers" de ce tuto
  - Une prise USB

https://github.com/tasmota/tasmotizer/releases

https://mqtt-explorer.com/

Étape 1 - Montage de la carte ESP

Pour commencer, nous devons tout d'abord réaliser le montage qui nous permettra de brancher nos capteurs de température à l'ESP8266. Pour cela, nous aurons besoin de brancher les capteurs DS18B20, les fils électriques et la résistance à la mini breadboard comme sur le 1er schéma ci-dessus.
Notez que le 1er et le 2e schéma sont identiques, seul le type de breadboard utilisée est différent. Le 3e schéma montre comment brancher plusieurs capteurs de température sur une breadboard classique. A vous d'adapter le montage sur la mini breadboard pour pouvoir brancher plusieurs capteurs si besoin.

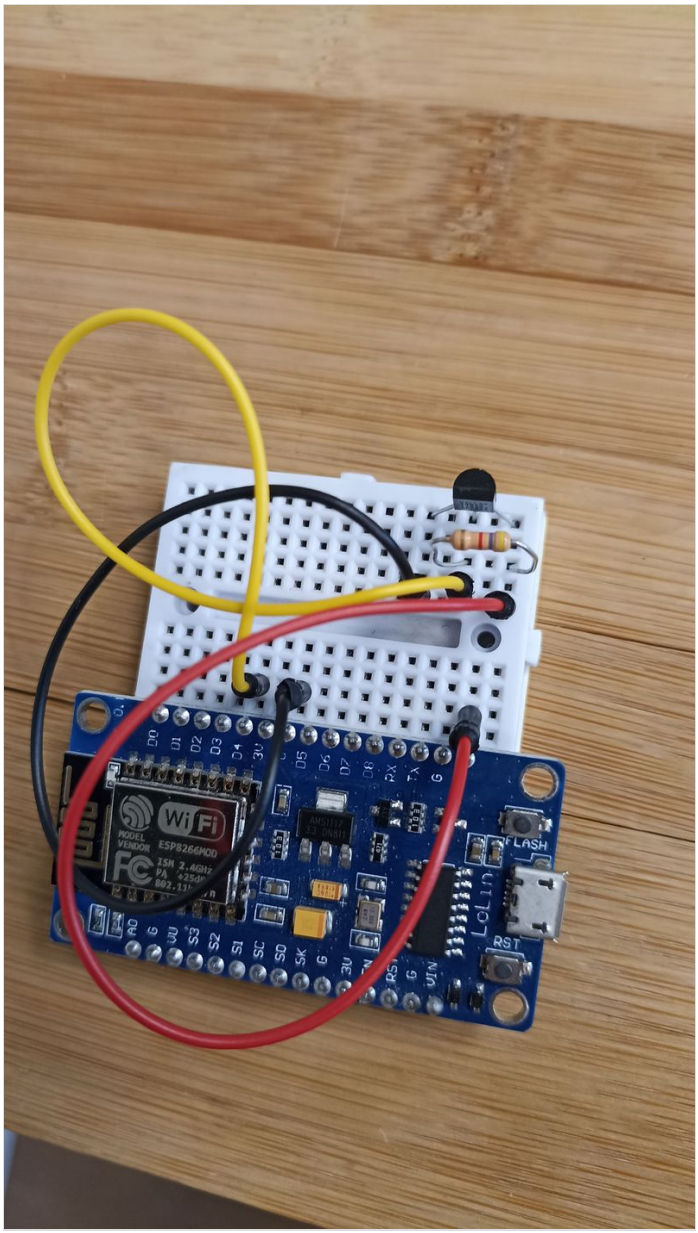
Si, pour mieux comprendre les schémas, vous ne savez pas comment fonctionne une breadboard classique, vous pouvez cliquer ici pour avoir des explications.

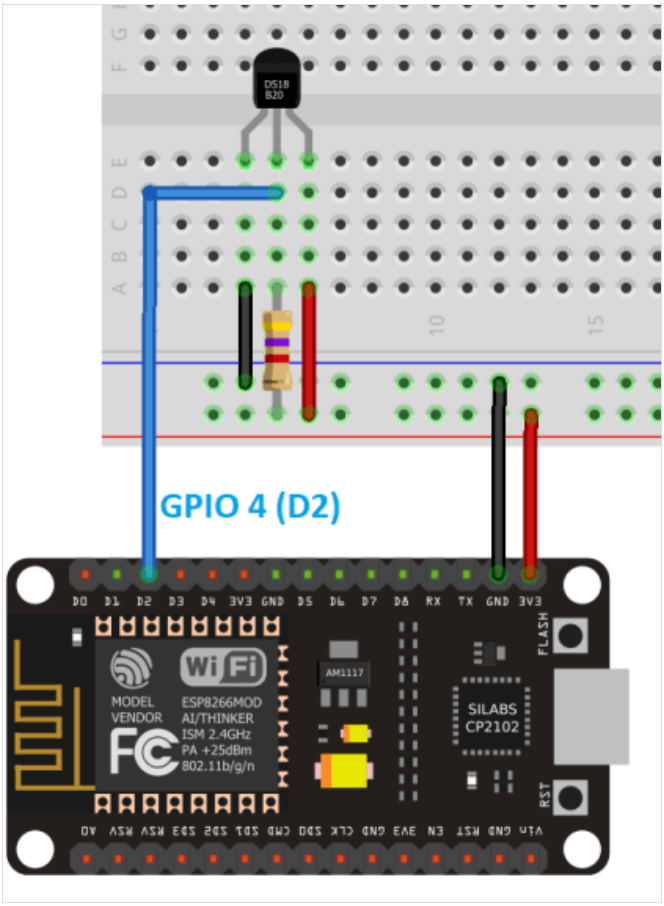
A noter que sur l'ESP, il y a plusieurs prises d'entrée (3V) et de sortie (G). Il a également plusieurs prises de capteurs (D0, D1, D2, etc.). Comme la carte est directement branchée sur la mini Breadboard, le plus simple serait d'utiliser la prise capteur D4 avec les prises entrée et sortie d'à côté.

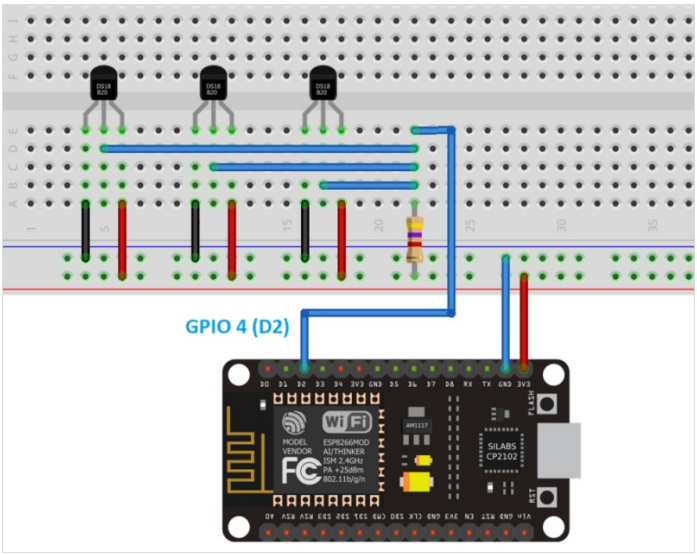
Notez également que les capteurs du modèles DS18B20 ont des broches qui doivent être branchés d'une manière bien spécifiques :

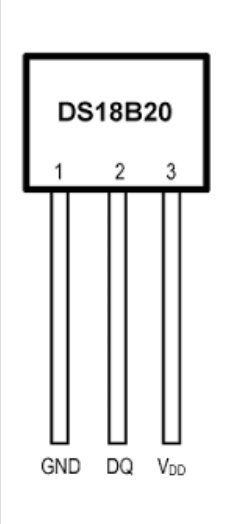
- la broche d'entrée VDD (rouge)
- la broche de collecte de donnée DQ (jaune)
- la broche de sortie GND (noir)

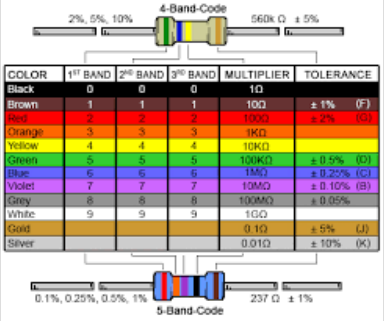
Si le capteur est branché à l'envers, il aura tendance à chauffer fort. C'est pour cela qu'il vaut mieux garder un œil au capteur après avoir branché le système. Cependant, cela ne l'abime pas forcément.




DS18
B20
GPIO 4 (D2)
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
WiFi
MODEL ESP8266MOD
VENDOR AI/THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
AM1117
SILABS
CP2102
FLASH
RST


GPIO 4 (D2)





COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8	100MΩ	± 0.05%
White	9	9	9	1GΩ	
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)

Étape 2 - Utilisation de Tasmotizer

Maintenant que votre système de capteurs est prêt, branchez votre carte ESP à votre ordinateur, puis lancez Tasmotizer.
Vous allez devoir réaliser les configurations suivantes :

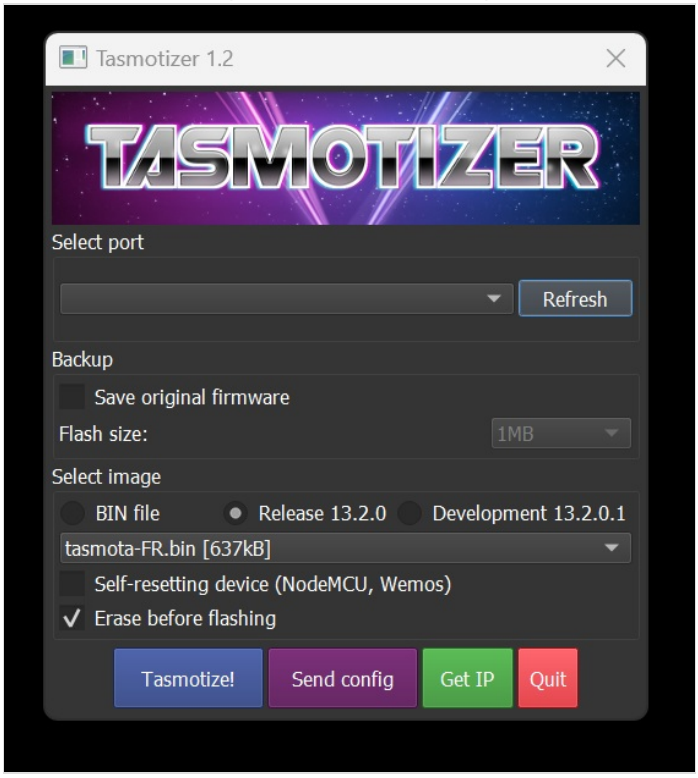
- Déjà, sélectionnez le port USB de votre ordinateur sur lequel est branché votre ESP (ce sera sûrement déjà fait).
- Il va falloir sélectionner un fichier .bin pour configurer le mode de Tasmota plus tard. Tasmotizer téléchargera celui que vous choisirez dans "Release" (comme par exemple le mode Tasmota en français indiqué sur l'image) si vous n'en avez pas déjà un.
- Ensuite il faudra faire en sorte que, avant d'utiliser l'ESP, que tout programme qui pourrait déjà s'y trouver soit effacé. Pour cela, cochez la case "Erase before flashing"
- Enfin, vous pouvez cliquer sur "Tasmotize!", et un téléchargement devrait s'être lancé.

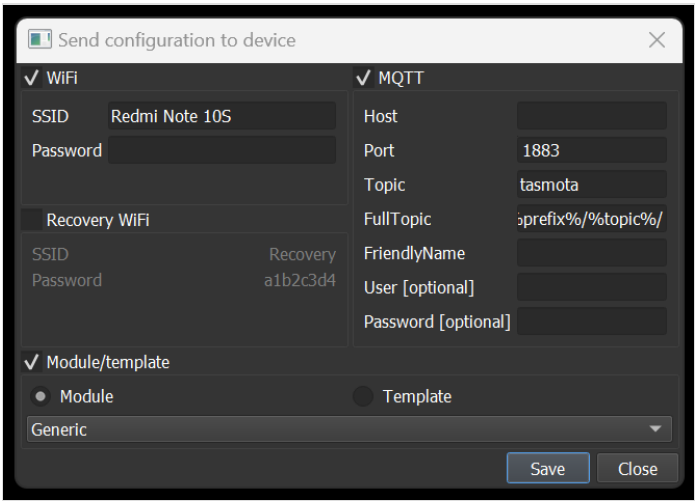
La phase d'après consistera à cliquer sur "Send config" pour connecter votre carte :

- à un réseau Wifi : c'est obligatoire pour que votre carte puisse se connecter à Tasmota par Wifi
- à un broker MQTT : pour cela vous aurez besoin :
  - "Host" : de l'adresse du broker
  - "Port" : du numéro de port du broker
  - "Full Topic" : le nom complet (sous forme de cheminement) du topic sur lequel vous voulez placer vos données
  - "User" : l'identifiant pour se connecter au broker (si nécessaire)
  - "Password" : le mot de passe pour se connecter au broker (si nécessaire)
- Dans "Module/template", il faudra choisir le module "Generic"

> Maintenant que votre carte est connectée à Internet, vous pouvez la débrancher de votre PC et la brancher à une prise murale par exemple. Lorsqu'elle sera à nouveau alimentée, elle se connectera automatiquement au réseau Wifi et au broker MQTT que vous avez choisi.

A présent que votre ESP est configurée, l'étape suivante consiste à accéder au menu de Tasmota qui est liée à votre carte. Pour cela, demandez à Tasmotizer l'adresse IP de votre ESP ("Get IP"), copiez la et, dans votre navigateur de recherche, coller cette adresse IP dans la barre de recherche. A partir de là, vous devriez pouvoir accéder à Tasmota.





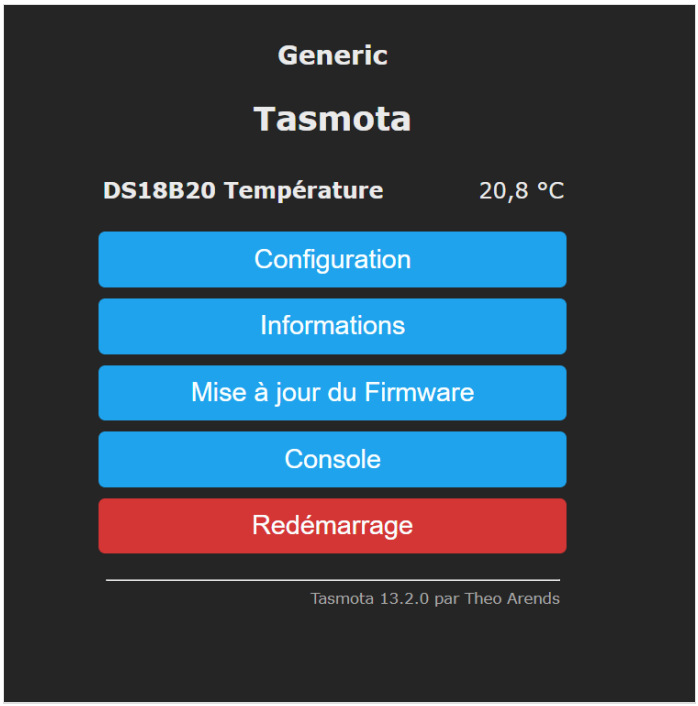
Étape 3 - Utilisation de Tasmota

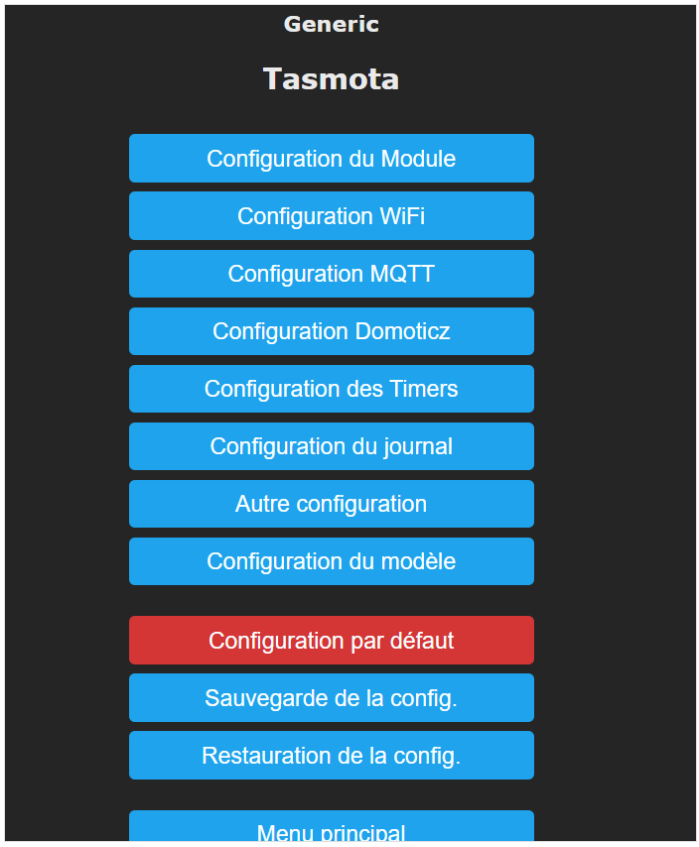
Une fois arrivé dans le menu principal de Tasmota, il va falloir que ce dernier détecte que la carte ESP utilise des capteurs de température de type DS18B20. Pour cela, aller dans "Configuration", puis dans "Configuration du Module".
Puis, sélectionnez sur la prise adéquate le bon modèle d'appareil (ici, dans notre cas, on sélectionne "DS18x20" sur la borne D4). Puis cliquez sur "Enregistrer" : le module de la carte va recharger pour afficher sur le menu principal les températures en temps réel de chaque capteurs que vous avez branché.
Vous pouvez également vérifiez les infos liées au broker MQTT sur lequel vous envoyez toutes vos données de température en cliquant sur "Configuration MQTT"

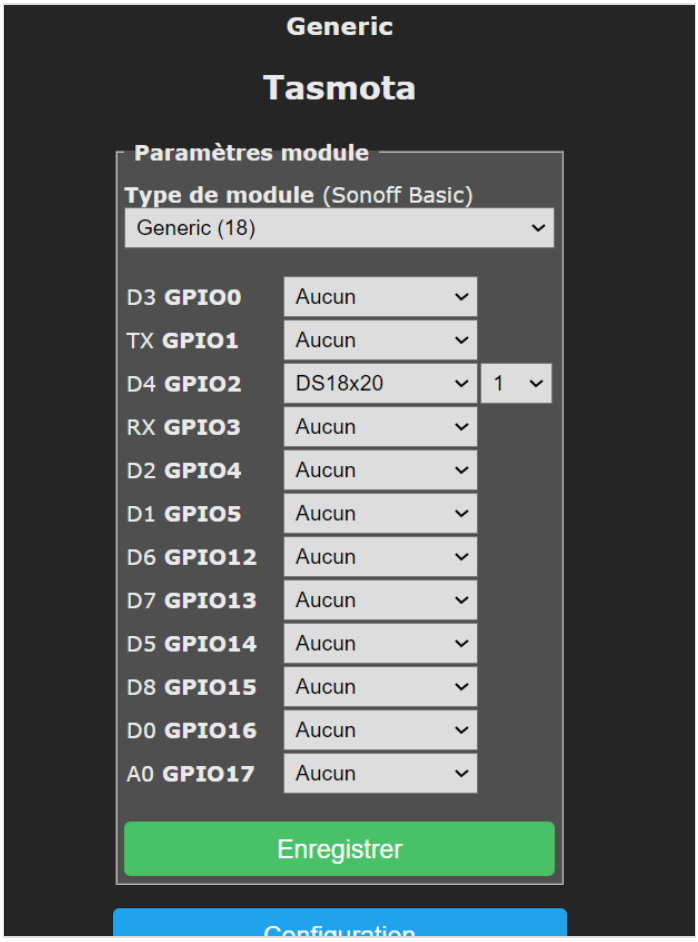
> Notez qu'il existe des broker MQTT publics tel que "test.mosquitto.org" qui ne nécessite donc pas de mot de passe et de non d'utilisateur pour y accéder.

Si vous retournez dans le menu principal de Tasmota et que vous cliquez sur "Console", le logiciel va vous montrer ce que la carte envoie au broker en temps réel. Ainsi, vous pouvez donc constater que la carte renvoie une valeur de prise de mesure du temps (date et heure) et une valeur de température pour chaque capteur branché à la carte.

> Si vous voulez modifier le temps d'intervalle entre deux mesures, allez dans "Configuration du journal" et modifiez la période de télémétrie. Elle est indiquée en secondes et est réglée par défaut en 300 secondes.

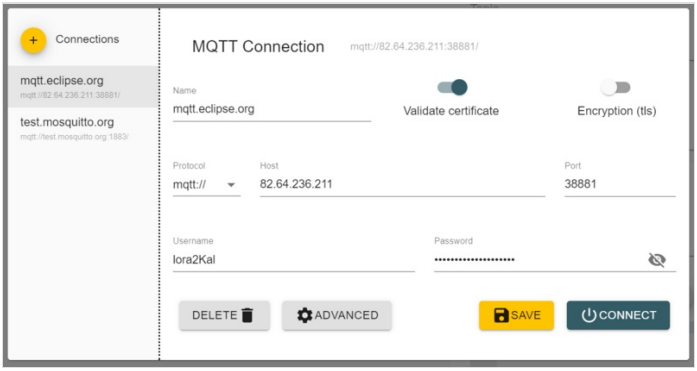


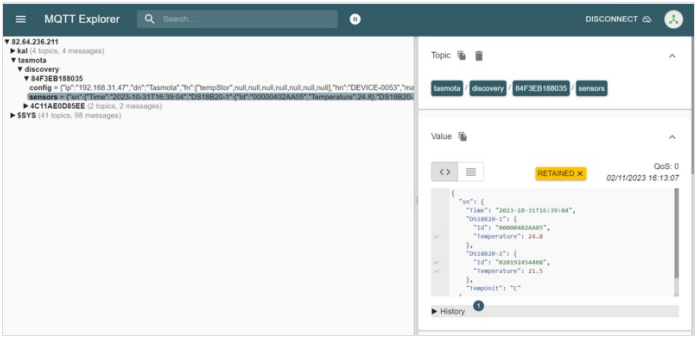



Generic
Tasmota
Paramètres module
Type de module (Sonoff Basic)
Generic (18)
D3 GPIO0 Aucun
TX GPIO1 Aucun
D4 GPIO2 DS18x20 1
RX GPIO3 Aucun
D2 GPIO4 Aucun
D1 GPIO5 Aucun
D6 GPIO12 Aucun
D7 GPIO13 Aucun
D5 GPIO14 Aucun
D8 GPIO15 Aucun
D0 GPIO16 Aucun
A0 GPIO17 Aucun
Enregistrer

Étape 4 - (Optionnel) Vérification des données que contient le broker MQTT

Il existe un outil qui permet de vérifier le contenu par topic sur un broker MQTT : MQTT Explorer.
Vous pouvez donc, en le téléchargeant en suivant le lien correspondant dans la section outil du wiki et en renseignant les informations de votre broker (host, port, username, password), afin de pouvoir observer le contenu de chaque topic compris dans ce broker, et donc regarder le contenu de celui qui vous intéresse. Vous pouvez dès lors constater que les données sont stockées sous format *json*, ce qui est un bon format pour envoyer plusieurs valeurs différentes en une seule fois.





Étape 5 - Récupération des données du serveur

A présent que les données qui nous intéressent sont stockées dans un serveur, il nous faut alors de quoi les récupérer. Un programme codé en langage Python est suffisant.
Pour cela, il faut lancer votre logiciel de programmation en Python, puis recopier le programme ci-contre. Mais avant de lancer le programme en question, il faut installer ce qu'on appelle en programmation des bibliothèques : la bibliothèque Paho MQTT pour pouvoir utiliser certaines commandes qui nous permettrons d'interagir avec les données du serveur MQTT, et la bibliothèque Matplotlib qui va nous permettre de tracer un graphique à partir de données. Pour cela, aller dans votre invite de commande (celui de votre ordinateur ou celui de VSCode si vous avez installé VSCode) et entrez les lignes suivantes l'une après l'autre pour installer respectivement les deux bibliothèques énoncées ci-dessus : "pip install paho-mqtt" et "pip install matplotlib".

> En programmation, un IDE (pour Integrated Development Environment) est un interpréteur Python intégré à un logiciel de programmation en langage Python. Par exemple, VSCode possède un IDE, en plus de pouvoir programmer en Python. Une alternative à cela consiste à développer son code dans un éditeur de texte (par exemple "Notepad++" sous Windows ou "GNU nano" sous Linux), puis de l'exécuter dans une invite de commande. Elle s'ouvre en tapant "cmd" dans la barre de recherche de Windows ou en ouvrant directement le terminal pour les systèmes macOS ou Linux.

Dorénavant, si vous recopiez le programme Python ci-contre, votre logiciel devrait pouvoir vous détecter normalement toutes les commandes que vous avez entré.

Pour plus de précision, on peut constater que le programme est composé en plusieurs parties :

- On commence par importer toutes les bibliothèques dont on a besoin.
- On créer plusieurs variables qui vont par la suite stocker les données de température et de temps (*max_data_points* peut être modifiable, servant à déterminer le nombre de mesures maximal que prendra en compte votre graphique).
- La fonction *on_message* sert à recevoir les données du topic qui nous intéresse et à les "ajouter" à nos variables précédemment créées. S'il y a une erreur avec vos données, alors la fonction vous enverra un message sur votre console d'exécution.
- La fonction *update_plot* sert à mettre le graphique à jour avec les nouvelles données à chaque fois qu'elle est appelée.
- La fonction *show_plot* va afficher le graphique mis à jour.
- Le reste du programme sert majoritairement à configurer l'abonnement au topic MQTT.

Le programme va donc former une boucle et modifier le graphique au fur et à mesure du temps.

> Evidemment, pour que votre programme fonctionne, il faut que votre ordinateur soit connecté à Internet. Sinon, un message d'erreur apparaîtra dans votre console.

```
import paho.mqtt.client as mqtt
import json
import matplotlib.pyplot as plt
from collections import deque
import threading
import time

# Variables pour stocker les données du graphique
max_data_points = 10  # Pour définir le nombre maximal de point que va prendre le graphique
data_queue1 = deque(maxlen=max_data_points)
data_queue2 = deque(maxlen=max_data_points)
data_queue3 = deque(maxlen=max_data_points)

```

```python
# Fonction de gestion des messages MQTT
def on_message(client, userdata, message):
    print("Message reçu!")
    payload = message.payload.decode("utf-8")
    print(f"Contenu du message : {payload}")

    try:
        data = json.loads(payload)  # "data" contient toutes les données du topic
        valeur1 = data.get("Time")  # On récupère la valeur associée à la mesure du temps
        valeur2 = (data.get("DS18B20-1")).get("Temperature")  # On récupère la donnée de température qui nous intéresse
        valeur3 = (data.get("DS18B20-2")).get("Temperature")

        # Mettre à jour les données du graphique
        data_queue1.append(valeur1[:10]+'\n'+valeur1[11:])  # On ajoute l'affichage du temps à une liste
        data_queue2.append(valeur2)  # On ajoute la valeur de la température à une autre liste
        data_queue3.append(valeur3)
        #print(list(data_queue1))
        #print(list(data_queue2))
        #print(list(data_queue3))

        # Mettre à jour le graphique
        update_plot()

    except json.JSONDecodeError as e:
        print(f"Erreur d'analyse JSON : {e}")
        # Ce message s'affichera s'il y a un problème avec les données de votre topic
```

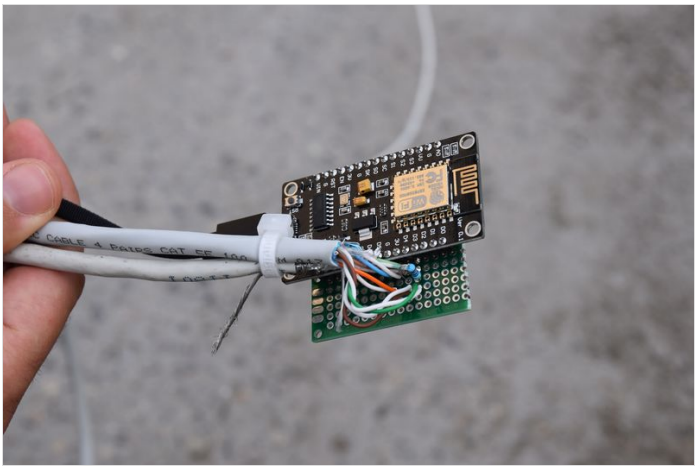
```python
# Fonction pour mettre à jour le graphique
def update_plot():
    print("Mise à jour du graphique!")
    plt.clf()  # Effacer le graphique existant

    # Tracer les données du graphique
    plt.plot(list(data_queue1),list(data_queue2), marker='o', linestyle='-')
    plt.plot(list(data_queue1),list(data_queue3), marker='o', linestyle='-')
        #On trace le graphique de la température par rapport au temps

    # Ajouter des étiquettes et un titre
    plt.xlabel("Date et heure")
    plt.xticks(rotation = 'vertical')
    plt.ylabel("Température en °C")
    plt.tick_params(axis = 'both', labelsize = 7)
    plt.title("Graphique en temps réel")
    plt.grid()

    # Afficher le graphique
    plt.pause(0.5)  # Pause pour permettre la mise à jour du graphique

def show_plot():
    plt.ion()  # Mode interactif pour permettre la mise à jour du graphique en temps réel
    plt.show()

```

```python
if __name__ == "__main__":
    # Configuration du client MQTT
    client = mqtt.Client()
    client.on_message = on_message

    # Informations d'authentification (optionnel si le serveur est public)
    username = "identifiant"
    password = "mot_de_passe"

    # Connexion au broker MQTT
    broker_address = "adresse_IP_du_serveur_MQTT"
    port = int("numéro_du_broker_MQTT")
    client.username_pw_set(username, password)
    client.connect(broker_address, port=port)
    print("Connecté au broker MQTT")

    # S'abonner au sujet MQTT "topic/example"
    topic = "nom_complet_du_topic"
    client.subscribe(topic)
    print("Abonné au topic")

    # Démarrer la boucle de l'application pour recevoir les messages
    client.loop_start()

    # Créer un thread pour afficher le graphique dans le thread principal
    plot_thread = threading.Thread(target=show_plot)
    plot_thread.start()

    # Garder le programme en cours d'exécution
    while True:
        time.sleep(0.1) # Pause courte pour laisser le temps à Matplotlib de mettre à jour le graphique

```

Étape 6 - Exemple d'utilisation

Comme exemple d'utilisation du Capteur/Enregistreur de température, voici un montage de ce système dans une cuve en inox d'une brasserie, contenant de l'eau chaude permettant au brassage de la bière. L'objectif étant de faire des mesures de température à l'intérieur et à l'extérieur, puis de recommencer après avoir mis en place un système d'isolation sur la cuve.

Le montage comprend donc 4 capteur DS18B20, trois à l'intérieur (fond, milieu et surface de la cuve pour avoir une mesure plus complète) et un à l'extérieur de la cuve. Ces 4 capteurs sont accrochés à un câble d'une longueur assez conséquente pour atteindre le fond de la cuve par des colliers de serrage.
De plus, au lieu d'utiliser une breadboard classique, on utilise comme alternative un autre modèle de board qui nous oblige à souder les fils pur les connecter entre eux.
Pour le reste, la manipulation reste pratiquement la même. Résultat: les données apparaissent sur un site Internet sous la forme d'un graphique représentant la température mesurée par rapport au temps, et ce pour les 4 capteurs.



Notes et références

- Définition MQTT : https://www.paessler.com/fr/it-explained/mqtt
- Broker MQTT publics : https://mntolia.com/10-free-public-private-mqtt-brokers-for-testing-prototyping/
- Documentation Tasmota : https://tasmota.github.io/docs/
- Complément : publier et s'abonner à un sujet MQTT : https://objets.ccdmd.qc.ca/manuel/5-3-echange-de-donnees-par-le-protocole-mqtt/