Projet ilot de chaleur

- Capteur Température & Humidité - Dragino LHT65-E1 LoraWAN
- Home Assistant et ESP32
- Home Assistant et traitement de données

Capteur Température & Humidité - Dragino LHT65-E1 LoraWAN

https://www.dragino.com/products/temperature-humidity-sensor/item/151-lht65.html

Connexion au Serveur LoRaWAN ChirpStack

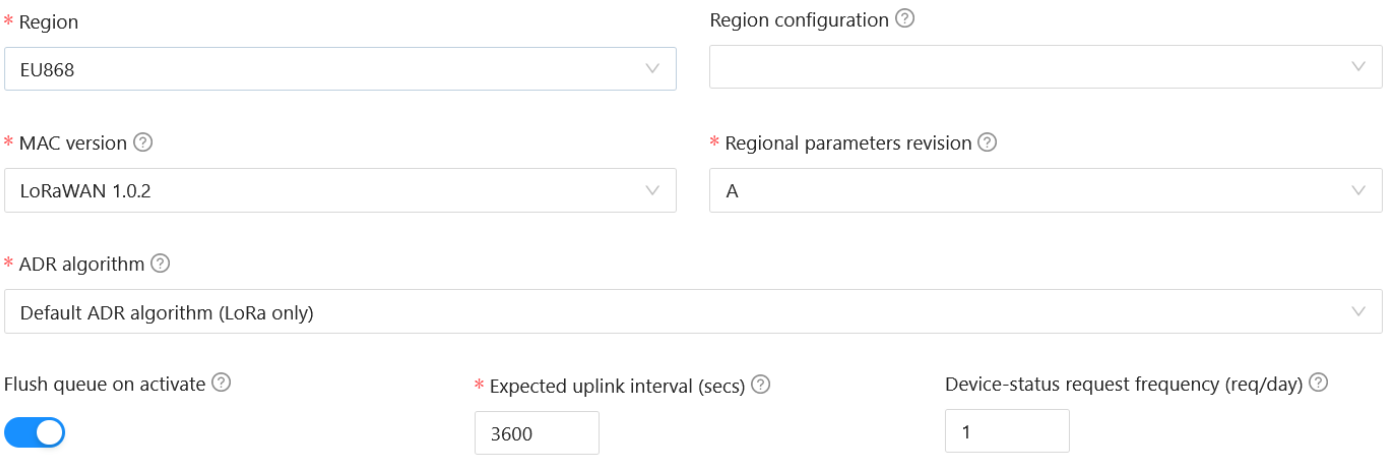
Un serveur est maintenu par l'Unistra pour le compte du réseau LoRa de l'Eurométropole de Strasbourg. Le serveur est une instance du serveur de réseau Open Source Chirpstack

Chirpstack fournit une interface Web pour l'administration des Portails (Gateway), des appareils. L'Interface Web permet également l'interfaçage des données du serveur vers des fournisseurs de Cloud, des bases de données et autres services communément utilisés pour le traitement des données émises par les appareils. Cette interface Web est accessible à l'URL suivante :

https://inetlab-lorawan.icube.unistra.fr/

Créer un nouveau profil pour le capteur

- Sélectionner la rubrique **Devices Profiles**.
- Sélectionner ensuite l'onglet **Général**.
- Donner un nom au profile et configurer le profile LoRa du pour le capteur Dragino LHT65 avec les paramètres suivants :



Ajouter le Codec du capteur Dragino LH65

- Télécharger le décodeur Dragino TH65 pour ChirpStack :

  https://github.com/dragino/dragino-end-node-decoder/tree/main/LHT65N
- Sélectionner l'onglet **Codec** puis l'option **Java-Script** à partir du menu déroulant et déposé le code du décodeur du capteur :

```
function decodeUplink(input) {
        return {
            data: Decode(input.fPort, input.bytes, input.variables)
        };
}

function Str1(str2){
  var str3 ="";
  for (var i=0;i<str2.length;i++){
    if (str2[i]<=0x0f){
    str2[i]="0"+str2[i].toString(16)+"";
    }
  str3+= str2[i].toString(16)+"";}
  return str3;
}
function str_pad(byte){
    var zero = '00';
    var hex= byte.toString(16);
    var tmp  = 2-hex.length;
    return zero.substr(0,tmp) + hex + " ";
}
```

```
function datalog(i,bytes){
  var Ext= bytes[6]&0x0F;
  var bb;
  if((Ext=='1')||(Ext=='9'))
{
bb=parseFloat(((bytes[0+i]<<24>>16 | bytes[1+i])/100).toFixed(2));
}
  else if(Ext=='2')
{
bb=parseFloat(((bytes[0+i]<<24>>16 | bytes[1+i])/100).toFixed(2));
}
else if(Ext=='4')
{
var Exti_pin_level=bytes[0+i] ? "High":"Low";
    var Exti_status=bytes[1+i] ? "True":"False";
    bb=Exti_pin_level+Exti_status;
}
else if(Ext=='5')
{
bb=bytes[0+i]<<8 | bytes[1+i];
}
else if(Ext=='6')
{
bb=(bytes[0+i]<<8 | bytes[1+i])/1000;
}
else if(Ext=='7')
{
bb=bytes[0+i]<<8 | bytes[1+i];
}
else if((Ext=='8')||(Ext=='14'))
{
bb=bytes[0+i]<<8 | bytes[1+i];
}
else if(Ext=='11')
{
  bb=parseFloat(((bytes[0+i]<<24>>16 | bytes[1+i])/100).toFixed(2));
}
  var cc= parseFloat(((bytes[2+i]<<24>>16 | bytes[3+i])/100).toFixed(2));
  var dd= parseFloat((((bytes[4+i]<<8 | bytes[5+i])&0xFFF)/10).toFixed(1));
```

```
  var ee= getMyDate((bytes[7+i]<<24 | bytes[8+i]<<16 | bytes[9+i]<<8 |
bytes[10+i]).toString(10));
  var string='['+bb+','+cc+','+dd+','+ee+']'+',';

  return string;
}

function getzf(c_num){
  if(parseInt(c_num) < 10)
    c_num = '0' + c_num;

  return c_num;
}

function getMyDate(str){
  var c_Date;
  if(str > 9999999999)
    c_Date = new Date(parseInt(str));
  else
    c_Date = new Date(parseInt(str) * 1000);

  var c_Year = c_Date.getFullYear(),
  c_Month = c_Date.getMonth()+1,
  c_Day = c_Date.getDate(),
  c_Hour = c_Date.getHours(),
  c_Min = c_Date.getMinutes(),
  c_Sen = c_Date.getSeconds();
  var c_Time = c_Year +'-'+ getzf(c_Month) +'-'+ getzf(c_Day) +' '+ getzf(c_Hour) +':'+
getzf(c_Min) +':'+getzf(c_Sen);

  return c_Time;
}

function Decode(fPort, bytes, variables) {
var Ext= bytes[6]&0x0F;
var poll_message_status=((bytes[6]>>6)&0x01);
var retransmission_Status=((bytes[6]>>7)&0x01);
var Connect=(bytes[6]&0x80)>>7;
var decode = {};
```

```
var data = {};
if((fPort==3)&&((bytes[2]==0x01)||(bytes[2]==0x02)||(bytes[2]==0x03)||(bytes[2]==0x04))){
 var array1=[]
 var bytes1="0x"
 var str1=Str1(bytes)
 var str2=str1.substring(0,6)
 var str3=str1.substring(6,)
   var reg=/.{4}/g;
 var rs=str3.match(reg);
 rs.push(str3.substring(rs.join('').length));
 rs.pop()
 var new_arr = [...rs]
 var data1=new_arr
 decode.bat=parseInt(bytes1+str2.substring(0,4)& 0x3FFF)
 if (parseInt(bytes1+str2.substring(4,))==1){
  decode.sensor="ds18b20"
 }
 else if(parseInt(bytes1+str2.substring(4,))==2){
    decode.sensor="tmp117"
 }
 else if(parseInt(bytes1+str2.substring(4,))==3){
  decode.sensor="gxht30"
 }
 else if(parseInt(bytes1+str2.substring(4,))==4){
  decode.sensor="sht31"
 }
 for (var i=0;i<data1.length;i++){
     var temp=(parseInt(bytes1+data1[i].substring(0,4)))/100
     array1[i]=temp
 }
 decode.Temp=array1
 {
     return decode;
   }
 }
 else if(fPort==5)
 {
   var sub_band;
   var freq_band;
```

```
var sensor;

if(bytes[0]==0x0B)
  sensor= "LHT65N";
else if(bytes[0]==0x1A)
  sensor= "LHT65N-PIR";

if(bytes[4]==0xff)
  sub_band="NULL";
else
  sub_band=bytes[4];

if(bytes[3]==0x01)
  freq_band="EU868";
else if(bytes[3]==0x02)
  freq_band="US915";
else if(bytes[3]==0x03)
  freq_band="IN865";
else if(bytes[3]==0x04)
  freq_band="AU915";
else if(bytes[3]==0x05)
  freq_band="KZ865";
else if(bytes[3]==0x06)
  freq_band="RU864";
else if(bytes[3]==0x07)
  freq_band="AS923";
else if(bytes[3]==0x08)
  freq_band="AS923_1";
else if(bytes[3]==0x09)
  freq_band="AS923_2";
else if(bytes[3]==0x0A)
  freq_band="AS923_3";
else if(bytes[3]==0x0B)
  freq_band="CN470";
else if(bytes[3]==0x0C)
  freq_band="EU433";
else if(bytes[3]==0x0D)
  freq_band="KR920";
else if(bytes[3]==0x0E)
```

```
    freq_band="MA869";

  var firm_ver= (bytes[1]&0x0f)+'.'+(bytes[2]>>4&0x0f)+'.'+(bytes[2]&0x0f);
  var bat= (bytes[5]<<8 | bytes[6])/1000;

  return {
    SENSOR_MODEL: sensor,
    FIRMWARE_VERSION: firm_ver,
    FREQUENCY_BAND: freq_band,
    SUB_BAND: sub_band,
    BAT: bat,
  };
}
if (retransmission_Status==0)
{
switch (poll_message_status) {□□
case 0:□
{
if(Ext==0x09)
{
  decode.TempC_DS=parseFloat(((bytes[0]<<24>>16 | bytes[1])/100).toFixed(2));
  decode.Bat_status=bytes[4]>>6;
}
else
{
  decode.BatV= ((bytes[0]<<8 | bytes[1]) & 0x3FFF)/1000;
  decode.Bat_status=bytes[0]>>6;
}

if(Ext!=0x0f)
{
  decode.TempC_SHT=parseFloat(((bytes[2]<<24>>16 | bytes[3])/100).toFixed(2));
  decode.Hum_SHT=parseFloat((((bytes[4]<<8 | bytes[5])&0xFFF)/10).toFixed(1));
}
if(Connect=='1')
{
  decode.No_connect="Sensor no connection";
}
```

```
if(Ext=='0')
{
  decode.Ext_sensor ="No external sensor";
}
else if(Ext=='1')
{
  decode.Ext_sensor ="Temperature Sensor";
  decode.TempC_DS=parseFloat(((bytes[7]<<24>>16 | bytes[8])/100).toFixed(2));
}
else if(Ext=='2')
{
  decode.Ext_sensor ="Temperature Sensor";
  decode.TempC_TMP117=parseFloat(((bytes[7]<<24>>16 | bytes[8])/100).toFixed(2));
}
else if(Ext=='4')
{
  decode.Work_mode="Interrupt Sensor send";
  decode.Exti_pin_level=bytes[7] ? "High":"Low";
  decode.Exti_status=bytes[8] ? "True":"False";
  decode.Exit_count= bytes[9]<<16 | bytes[10]<<8 | bytes[11];
  decode.Exit_duration= bytes[12]<<16 | bytes[13]<<8 | bytes[14];
}
else if(Ext=='5')
{
  decode.Work_mode="Illumination Sensor";
  decode.ILL_lx=bytes[7]<<8 | bytes[8];
}
else if(Ext=='6')
{
  decode.Work_mode="ADC Sensor";
  decode.ADC_V=(bytes[7]<<8 | bytes[8])/1000;
}
else if(Ext=='7')
{
  decode.Work_mode="Interrupt Sensor count";
  decode.Exit_count=bytes[7]<<8 | bytes[8];
}
else if(Ext=='8')
{
```

```
  decode.Work_mode="Interrupt Sensor count";
  decode.Exit_count=bytes[7]<<24 | bytes[8]<<16 | bytes[9]<<8 | bytes[10];
}
else if(Ext=='9')
{
  decode.Work_mode="DS18B20 & timestamp";
  decode.Systimestamp=(bytes[7]<<24 | bytes[8]<<16 | bytes[9]<<8 | bytes[10] );
}
else if(Ext=='11')
{
  decode.Work_mode="SHT31 Sensor";
  decode.Ext_TempC_SHT=parseFloat(((bytes[7]<<24>>16 | bytes[8])/100).toFixed(2));
  decode.Ext_Hum_SHT=parseFloat((((bytes[9]<<8 | bytes[10])&0xFFF)/10).toFixed(1));
}
else if(Ext=='14')
{
  decode.Work_mode="PIR Sensor";
  decode.Exti_pin_level=bytes[7] & 0x01 ? "Activity":"No activity";
  decode.Move_count=bytes[8]<<16 | bytes[9]<<8 | bytes[10];
}
else if(Ext=='15')
{
  decode.Work_mode="DS18B20ID";

decode.ID=str_pad(bytes[2])+str_pad(bytes[3])+str_pad(bytes[4])+str_pad(bytes[5])+str_pad(byte
s[7])+str_pad(bytes[8])+str_pad(bytes[9])+str_pad(bytes[10]);
}
}
  if((bytes.length==11)||(bytes.length==15))
  {
    return decode;
  }
break;

case 1:
  {
    for(var i=0;i<bytes.length;i=i+11)
    {
      var da= datalog(i,bytes);
```

```
      if(i==' 0')
        decode.DATALOG=da;
      else
        decode.DATALOG+=da;
    }
}
{
return decode;
}
break;
default:
    return {
      errors: ["unknown"]
    }
}
}
else
{
switch (retransmission_Status) {
  case 1:
  {
    for(var i=0;i<bytes.length;i=i+11)
    {
      var da= datalog(i,bytes);
      if(i==' 0')
        data.retransmission_message=da;
      else
        data.retransmission_message+=da;
    }
}
{
return data;
}
break;
default:
    return {
      errors: ["unknown"]
    }
}
```

```
}
}
```

Créer l'application (si nécessaire)

- Sélectionner la rubrique **Applications**.
- Cliquer sue le bouton ADD Application.
- Saisir le nom de l'application et sa description.

Créer un appareil dans une application

- Sélectionner l'application
- Cliquer sur le bouton **ADD device** pour ajouter un appareil.
- Saisir le nom de l'appareil et sa description.
- Saisir le numéro **EUI de l'appareil** (capteur commercial) ou le générer de manière aléatoire (capteur fait maison).

Home Assistant et ESP32

> On aurait voulu utiliser le serveur YunoHost du FabLab, mais seule la version Core de HomeAssistant est disponible sur YunoHost et celle-ci ne permet pas l'installation de l'add-on ESPHome. Une application en cours de création devrait permettre d'installer le dashboard ESPHome sur YunoHost.

> Par ailleurs le serveur YunoHost du FabLab est configuré pour se connecter au réseau wifi du FabLab qui n'est accessible que depuis les salles du FabLab. Étant donné que les ESP32 que l'on veut piloter doivent être sur le même réseau wifi que la Raspberry ESPHome, il est plus flexible d'utiliser un routeur mobile pour pouvoir faire ça de n'importe où dans ou hors de l'IUT.

Installation de Home Assistant

> **Home Assistant peut être installé sur un Raspberry 3B+ moyennant de rajouter 1Go de SWAP**
> **MAIS à l'usage** l'installation des paquets, la compilation et l'upload des programmes sur les ESP ont tendance à échouer (ou timeout). Il faut donc lancer les opérations à plusieurs reprises pour qu'elles aboutissent, ce qui rend la **solution peu utilisable**

- installer Home Assistant OS sur un Raspberry 4 ou 5 de préférence
- Connecter le Raspberry et un PC au même routeur, avec serveur DHCP activé
- Se connecter depuis le navigateur du PC à l'interface Web :

  http://homeassistant.local:8123
  - Déconnecter le PC de tout autre réseau pour éviter les conflits d'IP
  - Attention, connexion en http sur le réseau local
  - En dernier recours, chercher l'IP de la Raspberry sur l'interface du routeur et se connecter à, par ex. http://192.168.1.100:8123
- Configurer l'installation
  - Ajout du premier utilisateur
  - Géolocalisation
  - etc.

Configuration d'un HotSpot Wifi

> Home Assistant étant installé sur un Raspberry, on peut exploiter sa carte Wifi pour émettre un réseau wifi plutôt que de dépendre du routeur externe.

> Le réseau wifi émis par le Raspberry n'est pas très puissant, il faudra donc que les objets connectés soient dans la même pièce ou à proximité immédiate.

- Suivre ces instructions : Paramètres --> modules complémentaires --> Boutique des modules complémentaires
  - ou cliquer simplement sur : http://homeassistant.local:8123/hassio/store
  - puis --> trois points --> dépôts
- Ajouter https://github.com/mattlongman/hassio-addons-directory
- Attendre la mise-jour du store
- Installer le module `Hass.io Access Point`
- Aller dans Paramètres --> modules complémentaires --> Hass.io Access Point --> Configuration
- Définir un SSID et mot-de-passe
- Laisser les autres paramètres par défaut
- Activer le serveur DHCP pour que votre PC et les ESP32 récupèrent une adresse IP automatiquement
- Autoriser l'accès internet pour que vous puissiez avoir internet via le Raspberry lorsque vous connectez votre PC au Hotspot

> Le HotSpot wifi ne se monte que si la Raspberry est connectée à Internet via RJ45

Configuration de ESPHome

- Paramètres --> Modules Complémentaires (add-on) --> Boutique des modules complémentaires --> Installer ESPHome

> Les ESP32 avec Home Assistant ne supportent pas le Wifi 5GHz, rester sur du 2,4GHz

- Pour changer le réseau WIFI auquel se connectent les ESP32 --> modifier SECRETS.yaml

Ajout d'un ESP32 et d'un capteur DHT22

>

L'ESP32-E firebeetle 2 de DFRobot n'est pas officiellement supporté car son schéma n'a pas été ajouté à platformio.
Les I/O ne peuvent être désignés par leur nom A6.

De manière générale sur ESPHome, préférer les numérotations GPIO plutôt que les noms de pin type D1, A2, etc.

- Connecter l'ESP32 en USB-C au Raspberry
- Ajouter un appareil "New Device"
- Cliquer sur continue
- donner un nom parlant à l'ESP32, par ex. `station_eau_1`
- Sélectionner ESP32
- Valider la clé de chiffrement, elle pourra être récupérée par la suite au besoin
- Choisir une connexion par USB au Raspberry "Plug into the computer running ESPHome Dashboard"
- Sélectionner le port USB où est connecté l'ESP32
- La configuration initiale est créée
- cliquer sur EDIT pour la modifier pour ajouter les capteur
- Voir la discussion https://community.home-assistant.io/t/esphome-on-the-firebeetle-2-esp32-e/601747/15
- Trouver le numéro GPIO correspondant au port I/O utilisé, voir ce tableau
  - Ex. : D6 --> GPIO14
- Trouver le code pour le capteur à ajouter, par exemple pour le DHT22 :

```
# Configuration DHT22
sensor:
  - platform: dht
    pin: GPIO14
    temperature:
      name: "Température FabLab"
    humidity:
      name: "Humidité FabLab"
    update_interval: 60s
```

- Update --> Validate --> Install l'ESP32
- Voici les grandes étapes de compilation et déploiement

```
INFO ESPHome 2024.6.3
INFO Reading configuration /config/esphome/station-eau-1.yaml...
INFO Generating C++ source...
```

```
INFO Compiling app...
Processing station-eau-1 (board: esp32dev; framework: arduino; platform:
platformio/espressif32@5.4.0)
--------------------------------------------------------------------------------
Platform Manager: Installing platformio/espressif32 @ 5.4.0
INFO Installing platformio/espressif32 @ 5.4.0
Downloading  [####################################]  100%
Unpacking  [####################################]  100%
Platform Manager: espressif32@5.4.0 has been installed!
INFO espressif32@5.4.0 has been installed!
...
Library Manager: Resolving dependencies...
INFO Resolving dependencies...
Library Manager: Installing esphome/libsodium @ 1.10018.1
INFO Installing esphome/libsodium @ 1.10018.1
Downloading  [####################################]  100%
Unpacking  [####################################]  100%
Library Manager: libsodium@1.10018.1 has been installed!
INFO libsodium@1.10018.1 has been installed!
HARDWARE: ESP32 240MHz, 320KB RAM, 4MB Flash
 - toolchain-xtensa-esp32 @ 8.4.0+2021r2-patch5
Dependency Graph
|-- AsyncTCP-esphome @ 2.1.3
|-- WiFi @ 2.0.0
|-- FS @ 2.0.0
|-- Update @ 2.0.0
|-- ESPAsyncWebServer-esphome @ 3.2.2
|-- DNSServer @ 2.0.0
|-- ESPmDNS @ 2.0.0
|-- noise-c @ 0.1.4
Compiling .pioenvs/station-eau-1/src/esphome/components/api/api_connection.cpp.o
...
Compiling .pioenvs/station-eau-1/src/esphome/components/wifi/wifi_component_pico_w.cpp.o
Compiling .pioenvs/station-eau-1/src/esphome/core/application.cpp.o
...
Compiling .pioenvs/station-eau-1/src/esphome/core/util.cpp.o
Compiling .pioenvs/station-eau-1/src/main.cpp.o
Building .pioenvs/station-eau-1/bootloader.bin
Creating esp32 image...
Successfully created esp32 image.
```

```
Generating partitions .pioenvs/station-eau-1/partitions.bin
Compiling .pioenvs/station-eau-1/libeef/AsyncTCP-esphome/AsyncTCP.cpp.o
Archiving .pioenvs/station-eau-1/libeef/libAsyncTCP-esphome.a
Compiling .pioenvs/station-eau-1/lib64d/WiFi/WiFi.cpp.o
...
Compiling .pioenvs/station-eau-1/FrameworkArduino/wiring_shift.c.o
Archiving .pioenvs/station-eau-1/libFrameworkArduino.a
Linking .pioenvs/station-eau-1/firmware.elf
RAM:   [=         ]  12.4% (used 40648 bytes from 327680 bytes)
Flash: [=====     ]  49.3% (used 904333 bytes from 1835008 bytes)
Building .pioenvs/station-eau-1/firmware.bin
Creating esp32 image...
Successfully created esp32 image.
esp32_create_combined_bin([".pioenvs/station-eau-1/firmware.bin"], [".pioenvs/station-eau-
1/firmware.elf"])
Wrote 0xee320 bytes to file /data/build/station-eau-1/.pioenvs/station-eau-
1/firmware.factory.bin, ready to flash to offset 0x0
esp32_copy_ota_bin([".pioenvs/station-eau-1/firmware.bin"], [".pioenvs/station-eau-
1/firmware.elf"])
========================= [SUCCESS] Took 41.34 seconds =========================
INFO Successfully compiled program.
esptool.py v4.7.0
Serial port /dev/ttyUSB0
Connecting....
Chip is ESP32-D0WD-V3 (revision v3.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 40:22:d8:66:91:7c
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Flash will be erased from 0x00010000 to 0x000eefff...
...
Compressed 8192 bytes to 47...
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.1 seconds (effective 561.8 kbit/s)...
Hash of data verified.
```

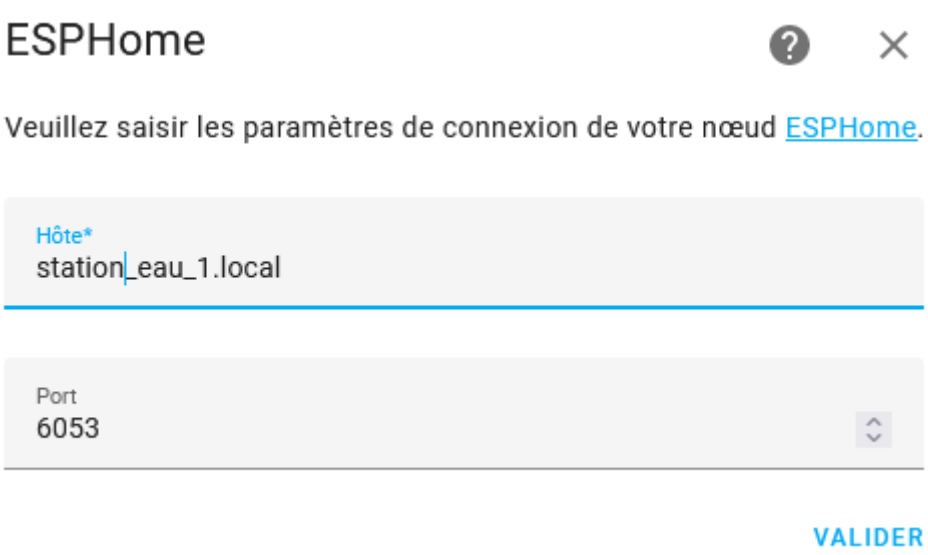
```
Leaving...
Hard resetting via RTS pin...
INFO Successfully uploaded program.
INFO Starting log output from /dev/ttyUSB0 with baud rate 115200
[11:14:57][I][logger:156]: Log initialized
[11:14:57][C][safe_mode:079]: There have been 0 suspected unsuccessful boot attempts
[11:14:57][D][esp32.preferences:114]: Saving 1 preferences to flash...
[11:14:57][D][esp32.preferences:143]: Saving 1 preferences to flash: 0 cached, 1 written, 0
failed
[11:14:57][I][app:029]: Running through setup()...
[11:14:57][C][wifi:047]: Setting up WiFi...
[11:14:57][C][wifi:060]: Starting WiFi...
[11:14:57][C][wifi:061]:   Local MAC: 40:22:D8:66:91:7C
[11:14:57][D][wifi:481]: Starting scan...
[11:14:57][W][component:157]: Component wifi set Warning flag: scanning for networks
[11:15:03][D][wifi:496]: Found networks:
[11:15:03][I][wifi:540]: - 'homeassistant-rasp3-hotspot' (B8:27:EB:46:5C:65) [redacted]▂▄▆
[11:15:03][D][wifi:541]:     Channel: 6
[11:15:03][D][wifi:542]:     RSSI: -48 dB
...
[11:15:09][C][logger:185]: Logger:
[11:15:09][C][logger:186]:   Level: DEBUG
[11:15:09][C][logger:188]:   Log Baud Rate: 115200
[11:15:09][C][logger:189]:   Hardware UART: UART0
[11:15:09][C][captive_portal:088]: Captive Portal:
[11:15:09][C][mdns:115]: mDNS:
[11:15:09][C][mdns:116]:   Hostname: station-eau-1
[11:15:09][C][esphome.ota:073]: Over-The-Air updates:
[11:15:09][C][esphome.ota:074]:   Address: station-eau-1.local:3232
[11:15:09][C][esphome.ota:075]:   Version: 2
[11:15:09][C][esphome.ota:078]:   Password configured
[11:15:09][C][safe_mode:018]: Safe Mode:
[11:15:09][C][safe_mode:020]:   Boot considered successful after 60 seconds
[11:15:09][C][safe_mode:021]:   Invoke after 10 boot attempts
[11:15:09][C][safe_mode:023]:   Remain in safe mode for 300 seconds
[11:15:09][C][api:139]: API Server:
[11:15:09][C][api:140]:   Address: station-eau-1.local:6053
[11:15:09][C][api:142]:   Using noise encryption: YES
[11:15:57][I][safe_mode:041]: Boot seems successful; resetting boot loop counter
```

```
[11:15:57][D][esp32.preferences:114]: Saving 1 preferences to flash...
[11:15:57][D][esp32.preferences:143]: Saving 1 preferences to flash: 0 cached, 1 written, 0 failed
```

- Noter l'adresse de l'ESP32 sur le réseau wifi local :

```
[11:15:09][C][api:140]:   Address: station-eau-1.local:6053
```

- Aller dans `Paramètres` --> `Appareils et Services` -->`ESPHome` -->



- Aller dans `Paramètres` --> `Appareils et Services` -->`ESPHome` -->sélectionner l'ESP32 par son nom, par ex. `station_eau_1`
- Les données du capteur s'affichent sur tout dashboard qui affiche la salle où le capteur est positionné

Débuggage

- Impossible de flasher un ESP32 `Error: Could not find one of 'package.json' manifest files in the package`
  - Les dépendances ce sont mal installées/téléchargées
    - Solution 1 : désinstaller et réinstaller ESPHome
    - Solution 2 : supprimer le dossier `/home/esphome/.platformio/packages`

Sources

- https://randomnerdtutorials.com/esp32-pinout-reference-gpios/
- https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf

- https://rdr-it.io/domotique/home-assistant-configuration-en-tant-que-point-dacces-wifi/

Home Assistant et traitement de données

Bases de données

- Home Assistant supporte InfluxDB

http://homeassistant.local:8123/hassio/addon/a0d7b954_influxdb/info