WiFi / MQTT :

Exemple d'utilisation simple :
Capteur SPI , écran OLED & Wemos

DeltaLab

Espace Maison Milon
2 Place E.Colongin
84600 Grillon

deltalabprototype.fr

Qu’est-ce que DeltaLab ?

DeltaLab est une association ‘loi 1901’ d’intérêt général, dont l’objectif est la création d’un espace dédié à l’innovation, à la production numérique au prototypage et à l’«expression artistique».

Le principe fondateur de l’associtation est **d’apprendre à faire soi-même, pour passer de l’dée à l’objet.**

Deltalab se spécialise en **Objets Connectés**, et est en train de créer un vaste «écosystème digital» entre Drôme et Vaucluse, pour répondre à des besoins non-couverts, mettre à disposition ressources et équipements pour usage professionnels et instaurer des partenariats avec les autres structures et initiatives numériques existantes.

Deltalab est aussi un **FabLab** (*Fabrication Laboratory* / *Laboratoire de Fabrication*), un tiers-lieu de type makerspace ou se trouve un atelier qui dispose de machines de fabrication comme des Imprimantes 3D ou des découpeuses Laser.

Deltalab se veut ouvert à tous publics : étudiants, professionnels, associations, inventeurs, designers,artistes, ...

Contexte de cette Documentation

Ce projet est une présentation de l'utilisation de MQTT dans un exemple concret : la mise en place d'un capteur de température / humidité DHT

Ce projet peut être considéré comme une base pour réaliser une station de capteurs plus ambitieuse (CF : doc "station météo")

Ce projet présente aussi l'ajout d'un écran OLED pour afficher les données en direct.

Table des matières

I - Introduction

Ce projet a pour but de présenter l'utilisation de MQTT et de la Wemos D1 en utilisant un exemple basique : un capteur SPI de température et humidité emboité sur la wemos, sur lequel s'emboitera un écran OLED

Le projet se décompose en 2 parties distinctes : le capteur SPI , l'écran OLED et la carte Wemos associée et le serveur Node-RED , ici très basique pour la présentation.

Les fonctionnalités proposées par ce projet sont les suivantes :

- Récupération des données du capteur (t° , , hum%) et affichage dans Node-RED, en passant par TTN , ainsi que sur l'écran OLED

Logiciels :

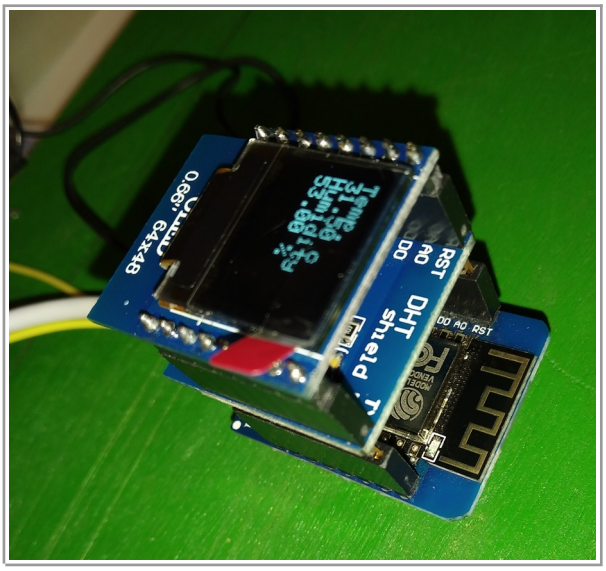
- Arduino IDE **:** téléchargeable à : https://www.arduino.cc/en/main/software
- Node-RED : à deltalab , entrez l'@ip ***192.168.1.45:1880*** dans un navigateur
- Wemos : ***Fichier > Préférences > URL du gestionnaire de cartes***. Entrez l'URL http://arduino.esp8266.com/stable/package_esp8266com_index.json
  Puis ***Outils> Type de Carte > Gestionnaire de cartes*** , rechercher **ESP8266** et installer. Choisissez ensuite **Wemos_D1_mini_pro** comme Type de Carte.
- Librairies :
  - **Librairie ESP8266WiFi** : à télécharger à https://github.com/esp8266/Arduino
    Puis à ajouter à Arduino (*Croquis > Inclure une Bibliothèque > Ajouter une Bibliothèque .zip*)
  - Librairie **DHT** : pour le capteur
    dans Arduino : ***insérer une bibliothèque > gérer les bibliothèques***
    entrez **DHT** et installez la librairie **DHT_Sensor_library**
  - **Librariie GFX :** pour l'écran
    dans Arduino : ***insérer une bibliothèque > gérer les bibliothèques***
    entrez **GFX** et installez la librairie **Adafruit_GFX_library**
  - **Librairie SSD1306 :** pour l'écran
    télécharger : https://github.com/mcauser/Adafruit_SSD1306/tree/esp8266-64x48
    Puis à ajouter à Arduino (***Croquis > Inclure une Bibliothèque > Ajouter une Bibliothèque .zip***)

II - Présentation du Circuit

Le ciruit est extrèmement simple. La Wemos D1 , le capteur DHT et l'écran OLED sont fixés sur des *shields* qui s'emboitent les uns sur les autres.

Vous devez juste les emboitez dans le bon sens , les mêmes pins les unes sur les autres
(3V avec 3V, RST avec RST, ...)

Des soudures sont à prévoir pour installer les bonnes *pins* pour l'emboitage



III - Codes Arduino

Librairies utilisées :

- **ESP8266WiFi - version 1.0**
- **PubSubClient - version 2.7.0**
- **WiFiClient - version 1.0**
- **AdafruitGFX - version 1.8.4**
- **DHT_Sensor - version 1.3.10**
- **Adafruit_SSD1036 - version 1.1.0 modifiée**

Code :

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include "Adafruit_SSD1306.h"
#include "Adafruit_SSD1306.cpp"
#include <DHT.h>
#include "PubSubClient.h"
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
```

```
#define OLED_RESET 0  // GPIO1
#define DHTPIN D4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
Adafruit_SSD1306 display(OLED_RESET);
MFRC522 mfrc522(SS_PIN, RST_PIN);
WiFiClient cli;
PubSubClient mqttClient(cli);
const char* ssid = "DeltaLab-Public";   // SSID du  point wifi
const char* password =  "Milon!Lab"; // Mot de passe du point Wifi

const char* mqttServer = "192.168.1.45"; //serveur MQTT
const int mqttPort = 1883;

// Reconnexion au serveur mqtt si déconnecté
boolean reconnect() {
  if (mqttClient.connect("DHT_MQTT")) {
    mqttClient.loop();
  }
  return mqttClient.connected();
}

// SetUp : WiFi , display & DHT
void setup()  {
  Serial.begin(115200);
  dht.begin();
  WiFi.hostname("DHT_MQTT");
  WiFi.begin(ssid,password);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // I2C addr 0x3C (for the 64x48)
  display.display();
  while(WiFi.status() != WL_CONNECTED){
        delay(500);
  }
  mqttClient.setServer(mqttServer,mqttPort);

}

//boucle : affichage et envoi
void loop()  {

  // Efface l'écran et positionne le curseur dans le coin supérieur gauche
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
```

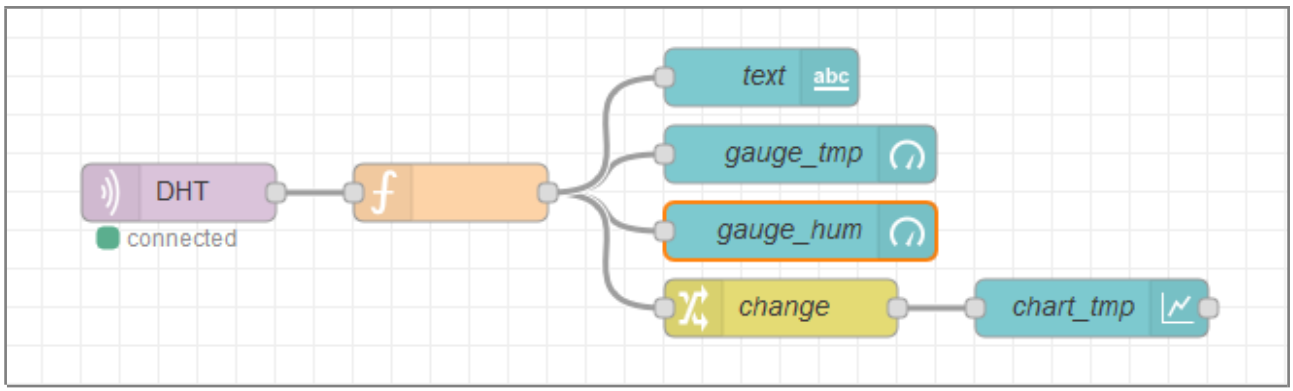
```
  //lecture du capteur DHT
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }else{
    Serial.print(t);
  }

//Affichage sur l'écran OLED
  display.println("Temp.");
  display.print(t);
  display.println(" c");
  display.println("Humidity");
  display.print(h);
  display.println(" %");
  display.display();

//Donneés pour MQTT
  String contenu = "";
  contenu +=  t;
  contenu += "-";
  contenu += h;
  char payload[11];
  unsigned int len = contenu.length()+1;
  contenu.toCharArray(payload,len);

//Envoi sur MQTT
  if(mqttClient.connect(ClientID)){
    Serial.println(payload);
    mqttClient.publish("DHT",payload);
    mqttClient.loop();
  }
  delay(10000);
}
```

Dans Arduino, pour choisir la bonne carte , allez dans **Outils > Type de carte**
On utilise ici une Wemos_D1_mini_pro. Vérifiez le Port et la vitesse d'écriture

Pour téléverser le code sur la carte, allez dans **croquis > Téléverser** , ou cliquez sur la flèche.

IV - Serveur Node-RED

Flow :



Explications :

- Le capteur envoie ses données et MQTT les récupère. Node-RED les lit ensuite grâce à la node MQTT in.
- La node function permet de décoder le message et de placer les données dans des champs distincts.
- Les données sont ensuite récupérées par les nodes de dashboard qui permettent de les afficher : des jauges pour l'humidité et la température , ainsi qu'un graphe pour l'évolution de cette dernière.
- On peut y associer une node 'file' pour enregistrer les données dans un fichier sur le serveur, ou une node de base de données pour y enregistrer les données choisies.

Pour le refaire :

1. **MQTT in**
   - Server :
     - ip: localhost
     - port : 1883
   - Topic : DHT

2. **Function Format**
   code de la function :

```
var id = [];
id = msg.payload.split('-')

return{
  payload:{
     tmp:id[0],
     press:id[1]
  }
}
```

3. **Gauges (Température , Humidité , Pression)**

   - UI group : créez un nouveau group pour la première et mettez les autre dedans
     - ui_tab : créez une nouvelle
     - width (largeur) : comme vous voulez

   - Type : gauge

   - Label :
     - pour la température :
     - pour l'humidité :

   - Value Format :
     - pour la température : {{msg.payload.tmp.toFixed(2)}}
     - pour l'humidité : {{msg.payload.hum.toFixed(2)}}

   - Echelle
     - pour la température : -5 à 45 , unité : °C ou degrés
     - pour l'humidité : 0 à 100 , unité : % ou pourcents

4. **Change**

   - première règle : SET
     - champs : msg.topic
     - valeur : tmp ou température

   - deuxième règle : MOVE
     - champs : msg.payload.tmp
     - destination : msg.payload

5. **Chart**

   - Ui_group : le même que les gauges
   - Type : line
   - axe Y : min = -15 , max = 45 (ou 0 / 40 - pour la beauté de la grille)

Dashbaord :

Pour ce résultat , changez le thème à **sombre** (**panneau de droite > onglet dashboard - icone de graphe > theme > dark(sombre)**) et réglez la width du groupe et le positionnement des éléments (**panneau de droite > dashboard > votre ui tab > layout**) Ici , le groupe à une largeur de **12** , le texte à une largeur de **12** et une hauteur de **1**, chaque gauge à une largeur de **6** et une hauteur de **4** et le graphe à une largeur de **12** et une hauteur de **8.**