Utilisation d'un capteur de de Température Dallas et envoi des données via WiFi

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 présente l'utilisation d'un capteur de température de liquide. Ces capteurs sont utilisés dans beaucoup de machines (lave-linges , lave-vaisselle,...) et dans de nombreux projets autour de l'Eau (monitoring d'une rivière, d'un chateau d'eau,...).

A DeltaLab, un projet d'Aquaponie est en projet, et ces capteurs seront évidemment utilisés dès le début.

Table des matières

I - Introduction

Ce projet a pour but de présenter l'utilisation d'un capteur de température aquatique, dans le but de l' intégrer à un système plus grand.

Le projet se décompose en 2 parties distinctes : le capteur et la carte Wemos associée, et le serveur Node-RED.

Les fonctionnalités proposées par ce projet sont les suivantes :

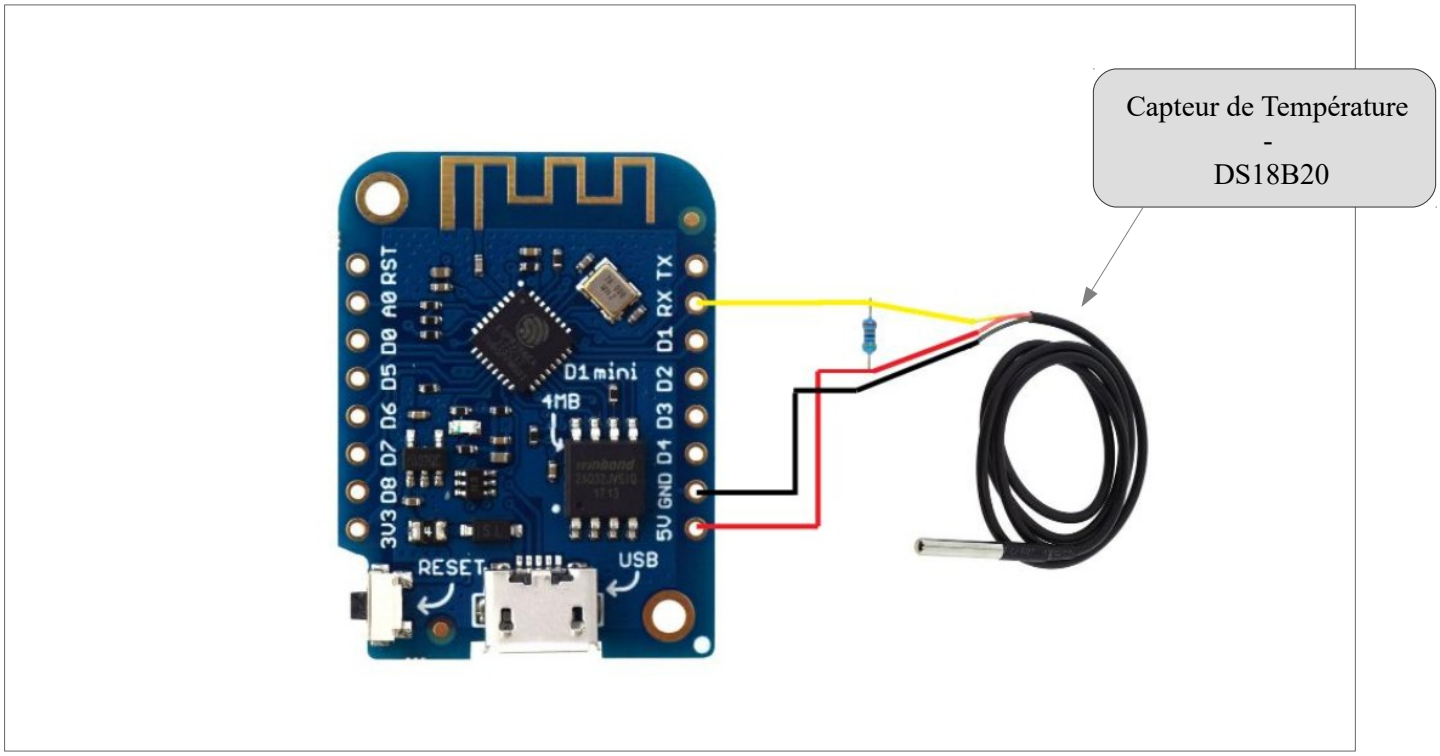
- Récupération des données des capteurs et affichage dans Node-RED, en passant par MQTT.
- Possible stockage dans une BdD.

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*)
  - **OneWire & DallasTemperature** : dans l'IDE Arduino , allez dans Croquis > Inclure une Bibliothèque > Gérer les Bibliothèques, et entrez le nom des 2 bibliothèques.

II - Présentation du Circuit

Circuit :



Connexions :

wemos	Capteur de Température	Couleur
5V	Rouge	Rouge
RX	Jaune	Jaune
GND	Noir	Noir

Explication :

- Le capteur de Température DS18B20 est entièrement Water-Proof. Il en existe de différentes longueurs, les plus longs mesurant plusieurs mètres. La longueur n'est pas problématique pour la qualité des données grâce au support et au protocole utilisé. Il utilise la librairie OneWire, qui permet d'avoir plusieurs capteurs indentiques sur une seule Pin et de pouvoir récupérer leurs données individuellement. Il renvoie une tempéraure en degré celsius et fonctionnne entre -50 et 80°C.

- Le capteur à besoin d'une résistance de **4,7 kΩ** entre l'alimentation positive et le cable de données.

III - Codes Arduino

Librairies utilisées :

- **ESP8266WiFi**
- **PubSubClient - version 2.7.0**
- **WiFiClient - version 1.0**
- **OneWire**
- **DallasTemperature**

Code :

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include "PubSubClient.h"
#include <ESP8266WiFi.h>
#include <WiFiClient.h>

#define pin_tmp RX

double tmp;
OneWire Wir(pin_tmp);
DallasTemperature sensor(&Wir);

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();
}

// Récupération des donneés des capteurs
void getData(){
  sensor.requestTemperatures();
  tmp = sensor.getTempCByIndex(0);
  Serial.println(sensor.getTempCByIndex(0));
}
```

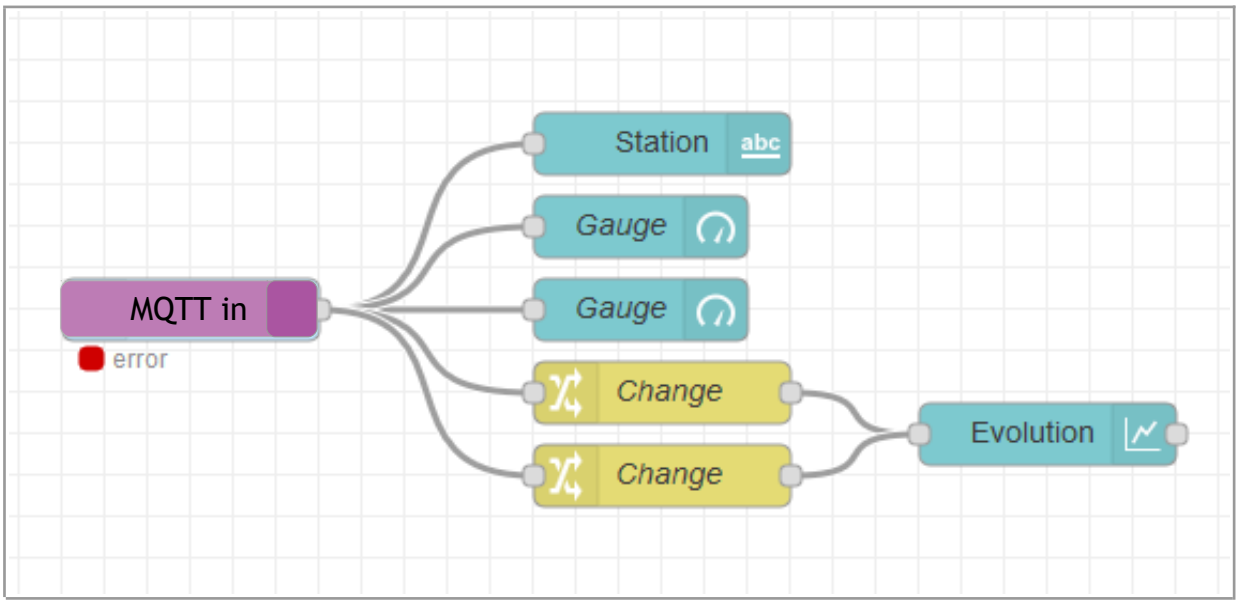
```
void setup() {
    Serial.begin(115200);
    Serial.println(F("Starting"));
    pinMode(pin_tmp , INPUT_PULLUP);
    sensor.begin();
    pinMode(pin_turbi , INPUT);

  WiFi.hostname("DHT_MQTT");
  WiFi.begin(ssid,password);
  while(WiFi.status() != WL_CONNECTED){
        delay(500);
  }
  mqttClient.setServer(mqttServer,mqttPort);
}


void loop() {

   getData();

  //Donneés pour MQTT
  String contenu = "";
  contenu += tmp;
  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("TT",payload);
    mqttClient.loop();
  }
  delay(10000);
}
```

Dans arduino, pour choisir la bonne carte , allez dans **Outils > Type de carte**
On utilise ici une TTGO_LORA32_OLED_V1. Vérifiez le Port et la vitesse d'écriture

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

III - 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

  - Les données sont ensuite récupérées par les nodes de dashboard qui permettent de les afficher : une jauge ainsi qu'un graphe.

  - 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**

     - Serveur :
       - id : l'id de votre application sur TTN
       - acces key : l'application_key de votre application sur TTN

     - Topic : le topic utilisé (TMP dans le code);

2. **Gauges**

   - UI group : créez un nouveau group pour la première et mettez les autre dedans
     - ui_tab : créez une nouvelle
     - width (largeur) : comme vous voulez
     - vous pouvez rendre son nom visible ou non dans le dashboards

   - Type : gauge

   - Label :
     - (icone thermomètre)

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

   - Echelle
     - pour la température : -50 à 80, unité : °C ou degrés

3. **Changes**

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

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

4. **Chart**

   - Ui_group : le même que les gauges
   - Type : line
   - axe Y : min = -50 , max = 100
   - Réglez le nombre de points max ou la durée représentée comme vous le souhaitez

Dashboard :

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 **18** , chaque texte à une largeur de **6** et une hauteur de **1**, la gauge à une largeur de **6** et une hauteur de **4** et le graphe à une largeur de **18** et une hauteur de **8**.