LoRaWAN :

Exemple d'utilisation simple : Capteur I2C & TTGO

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 LoRaWAN dans un exemple concret : la mise en place d'un capteur de température / humidité BME280

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

Table des matières

I - Introduction

Ce projet a pour but de présenter l'utilisation de LoRaWAN et de The Things Network (TTN) en utilisant un exemple basique : un capteur I2C de température , humidité et pression relié à une carte TTGO

Le projet se décompose en 2 parties distinctes : le capteur I2C et la carte TTGO associée et le serveur Node-RED , ici très basique pour la présentation.

Les fonctionnalités proposées par ce projet sont les suivantes :

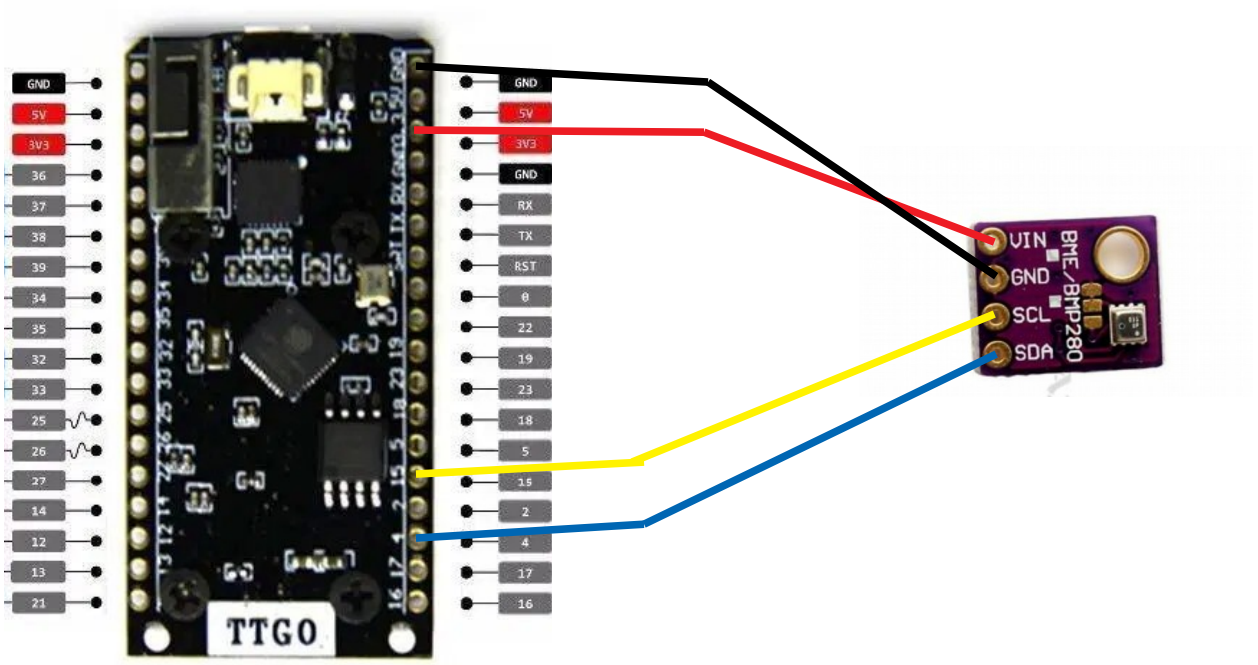
- Récupération des données du capteur (t° , press, hum%) et affichage dans Node-RED, en passant par TTN.

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
- TTGO : ***Fichier > Préférences > URL du gestionnaire de cartes***. Entrez l'URL https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
  Puis ***Outils> Type de Carte > Gestionnaire de cartes ,*** rechercher **ESP32** et installer. Choisissez ensuite **TTGO_LoRa_32** comme Type de Carte.
- Librairies :
  - **AdafruitBME280** : pour le capteur
    - dans Arduino : ***croquis > insérer une bibliothèque > gérer les bibliothèques***
      entrez **BME280** et installez la librairie **Adafruit_BMP280_Library**
  - **MCCI_LoRaWAN_LMIC** : pour LoRaWAN
    - dans Arduino : ***insérer une bibliothèque > gérer les bibliothèques***
      entrez **LMIC** et installez la librairie **MCCI_LoRaWAN_LMIC_Library**
    - **important** : allez dans votre dossier arduino (défaut : ***Documents\Arduino***) et allez dans la configuration de la librairie pour activer la bonne fréquence (***libraries > MCCI_LoRaWAN_LMIC_Library > project_config > lmic_project_config.h***). Dans ce fichier , décommentez (enlevez le ' **//** ') la ligne **#define CFG_eu868 1** (fréquence européenne) et commentez (mettez un ' **//** ') la ligne qui était décommentée (par défaut **#define CFG_us915 1** , fréquence américaine)

II - Présentation du Circuit

- **Circuit :**



- **Connexions :**

Port - Capteur BMP280	Port - TTGO	Couleur
Vin	3,3V	Rouge
GND	GND	Noir
SDA	04	Bleu
SCL	15	Jaune

- **Explication :**

Le capteur BMP280 communique avec la TTGO via le bus I2C.

La TTGO communique régulièrement les données du capteur à TTN, et Node-RED les y récupère et les met en forme.

III - Codes Arduino

Librairies utilisées :

- **MCCI_LoRaWAN_LMIC - version 1.4.6**
- **AdafruitBMP280 - version 2.7.0**

Code :

```
#include <lmic.h>
#include <hal/hal.h>
#include <Adafruit_BME280.h>
#include <Wire.h>
#include <SPI.h>

#define SCK     5    // GPIO5  -- SX1278's SCK
#define MISO    19   // GPIO19 -- SX1278's MISnO
#define MOSI    27   // GPIO27 -- SX1278's MOSI
#define SS      18   // GPIO18 -- SX1278's CS
#define RST     14   // GPIO14 -- SX1278's RESET
#define DI0 26 // GPIO26 -- SX1278's IRQ(Interrupt Request)
#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme;
double Temp;
double Press;
double Alt;
double Hum;
osjob_t sendjob;
const unsigned TX_INTERVAL = 20 ;

// Données à envoyer
typedef union {
  float f[4];
  unsigned char bytes[16];
} donnees;
donnees datas;

// Application EUI : en 'lsb' (little endian format)
static const u1_t PROGMEM APPEUI[8]={ 0xBF, 0x24, 0x03, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// Device EUI : même chose qu'au dessus
static const u1_t PROGMEM DEVEUI[8]={0x62, 0xBA, 0x24, 0x19, 0x19, 0xE2, 0x6F, 0x00};
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}
```

```
 // Application Key : vous pouvez la copier telle qu'elle depuis TTN
static const u1_t PROGMEM APPKEY[16] = { 0xF1, 0xDA, 0x59, 0xA7, 0x56, 0xD4, 0xF8,
0xF0, 0x02, 0x31, 0x02, 0x62, 0x8D, 0xF1, 0x21, 0xF5  };
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}


//Pin Mapping
const lmic_pinmap lmic_pins = {
  .nss = 18,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 14,
  .dio = {26, 33, 32},
};


// Gestionnaire des évenements de TTN
void onEvent (ev_t ev) {
   switch(ev) {
      case EV_JOINING:
          Serial.println(F("EV_JOINING - antenne cherchée"));
          break;
       case EV_JOINED:
          Serial.println(F("EV_JOINED - antenne connectée"));
          LMIC_setLinkCheckMode(0);
       break;
       case EV_TXCOMPLETE:
          Serial.println(F("EV_TXCOMPLETE - message envoyé"));
          os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), Send);
       break;
       default:
          Serial.print(F("Unknown event: "));
          Serial.println((unsigned) ev);
       break;
    }
}


// Récupération des données du capteur
void getData(){
  Temp = bme.readTemperature();
  Press = bme.readPressure()/100.0F;
  Alt = bme.readAltitude(SEALEVELPRESSURE_HPA);
  Hum = bme.readHumidity();
}
```

```
// Envoi des données à TTN
void Send(osjob_t* j){
      if (LMIC.opmode & OP_TXRXPEND) {
       Serial.println(F("OP_TXRXPEND, not sending"));
   } else {
       getData();
       datas.f[0] = Temp;      //température
       datas.f[1] = Press;       //humidité
       datas.f[2] = Alt;
       datas.f[3] = Hum;
       LMIC_setTxData2(1, datas.bytes, 16, 0);
       Serial.println(F("Packet queued"));
   }
}


// SetUp : démarrage du capteur, initialisation de LMIC et de la boucle d'envoi
void setup() {
   Serial.begin(115200);
   Serial.println(F("Starting"));
   Wire1.begin(4,15);
   while(!bme.begin(0x76 , &Wire1)){
     Serial.println("Problème avec le BME280");
     delay(1000);
   }
   SPI.begin(SCK,MISO,MOSI,SS);
   os_init();
   LMIC_reset();
   LMIC_setClockError(MAX_CLOCK_ERROR * 10/100);
   Send(&sendjob);
}


// Boucle automatique
void loop() {
   os_runloop_once();
}
```

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.

IV - The Things Network

1. Ajout du device

- Sur **TTN (thethingsnetwork.org) ,** connectez-vous (ou créez un compte) , allez dans la **console**, choisissez **Applications** et **Ajoutez** une nouvelle Appli. Reinseignez un ID (nom) et une descritpion (facultatif).

- Dans cette Application , ajoutez un nouveau **Device** (appareil). Renseignez un nom et une description, et cliquez une fois sur les flèches du champs EUI pour qu'il soit auto-généré.

- Dans l'**overview de votre Device** , vous avez accès aux **EUIs** nécéssaires dans le programme. Appuyer sur les flèches pour le rendre lisible , puis sur les crochets pour le changer en **lsb**. Vous pouvez les copier / coller dans votre code.

- Dans l'**overview de l'application** , vous avez accès à l'**App_Key**. Copiez-Collez la telle qu'elle.

2. Décodage du payload

Votre Application > Payload Format. Collez cette fonction dans la zone '**Decoder**'

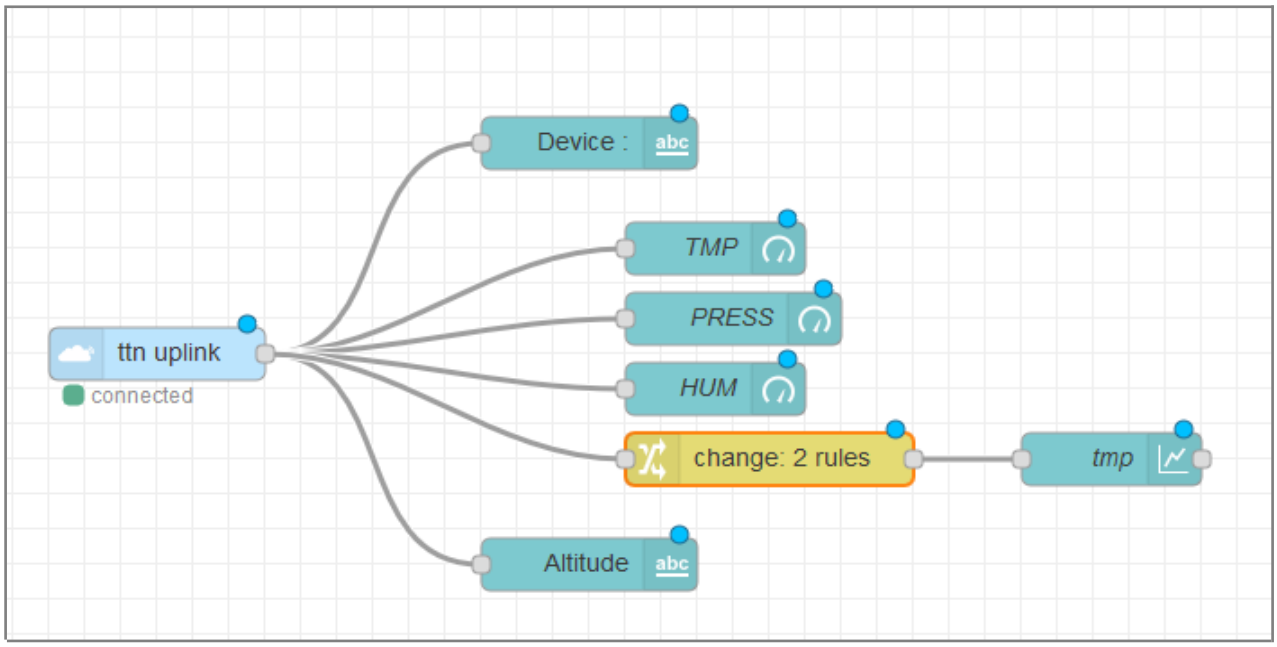
```
function B2F32(bytes) {
	var sign = (bytes & 0x80000000) ? -1 : 1;
	var exponent = ((bytes >> 23) & 0xFF) - 127;
	var significand = (bytes & ~(-1 << 23));
	if (exponent == 128)
		return sign * ((significand) ? Number.NaN : Number.POSITIVE_INFINITY);
	if (exponent == -127) {
		if (significand === 0) return sign * 0.0;
		exponent = -126;
		significand /= (1 << 22);
	} else significand = (significand | (1 << 23)) / (1 << 23);
		return sign * significand * Math.pow(2, exponent);
}

function Decoder(bytes, port) {
	var tmp = bytes[3] << 24 | bytes[2] << 16 | bytes[1] << 8 | bytes[0];
	var pre = bytes[7] << 24 | bytes[6] << 16 | bytes[5] << 8 | bytes[4];
	var alt = bytes[11] << 24 | bytes[10] << 16 | bytes[9] << 8 | bytes[8];
	var hum = bytes[15] << 24 | bytes[14] << 16 | bytes[13] << 8 | bytes[12];

	return{
		tmp: B2F32(tmp) , press: B2F32(pre), alt: B2F32(alt), hum: B2F32(hum)
	};  }
```

IV - Serveur Node-RED

Flow :



Explications :

- Le capteur envoie ses données et TTN les récupère. Node-RED les lit ensuite grâce à la node TTN-Uplink
- Les données sont ensuite récupérées par les nodes de dashboard qui permettent de les afficher : un texte pour l'altitude , des jauges pour la pression , 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. TTN_Uplink

- Application :
  - id : l'id de votre application sur TTN
  - acces key : l'application_key de votre application sur TTN
- Device (dev_id) : l'id de votre ttgo , si vous voulez filtrer les messages entrants

2. **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
     - vous pouvez rendre son nom visible ou non dans le dashboards

   - Type : gauge

   - Label :
     - pour la température : (icone thermomètre)
     - pour la pression : (icone nuage)
     - pour l'humidité : (icone goutte)

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

   - Echelle
     - pour la température : -5 à 45 , unité : °C ou degrés
     - pour la pression : 900 à 1200 , unité : hPa ou hectoPascals
     - pour l'humidité : 0 à 100 , unité : % ou pourcents

3. **Change**

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

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

4. **Chart**

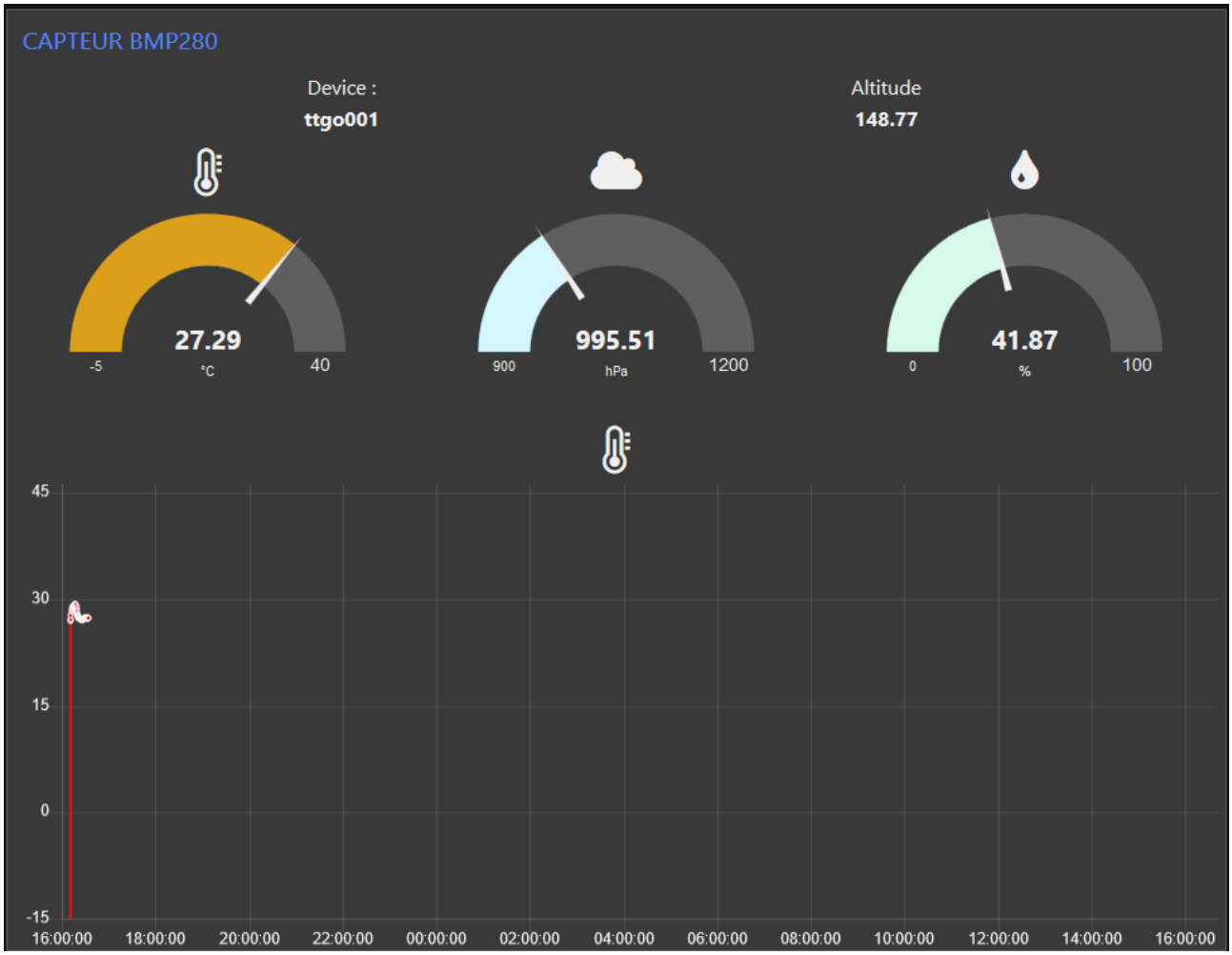
   - Ui_group : le même que les gauges

   - Type : line

   - axe Y : min = -15 , max = 45 (ou 0 / 40 - pour la beauté de la grille)

   - Réglez le nombre de points max ou la durée représentée comme vous le souhaitez

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