

Projet ARNG

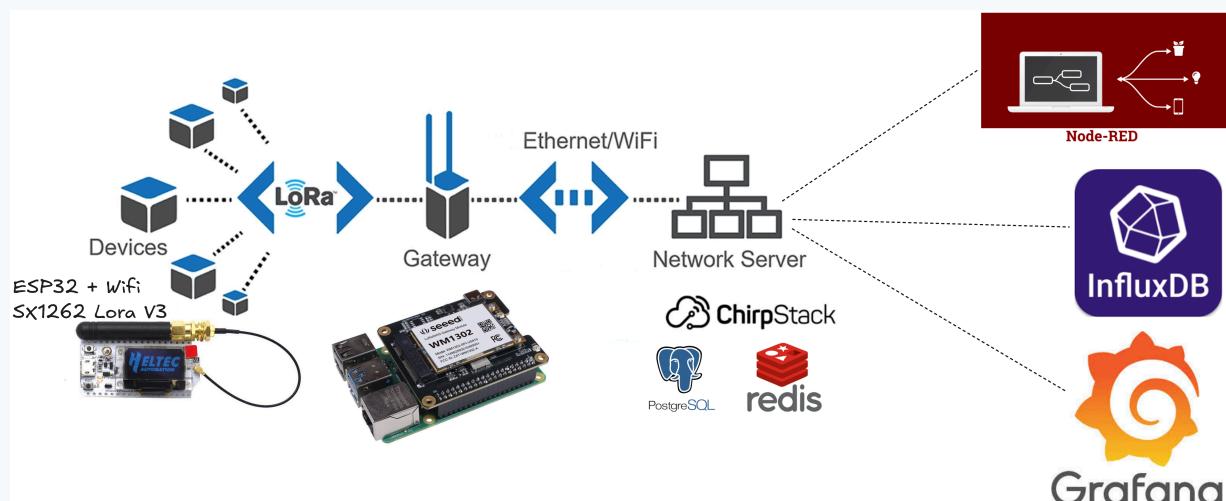
Contexte Mise en œuvre et validation de la suite de la plateforme IoT LoRaWAN avec un device réel basé sur la carte de développement Heltec ESP32 WiFi LoRa V3.

Supports

- Docker Compose fourni en annexe + fichiers hérités des autres TP
- Arduino IDE (Linux): <https://docs.arduino.cc/software/ide-v1/tutorials/Linux/>
- Heltec ESP32 (Arduino IDE): https://docs.heltec.org/en/node/esp32/esp32_general_docs/quick_start.html#esp32-via-arduino-ide
- InfluxDB ↔ Grafana: <https://docs.influxdata.com/influxdb/v2/tools/grafana/>
- Grafana (getting started InfluxDB): <https://grafana.com/docs/grafana/latest/getting-started/get-started-grafana-influxdb/>
- Node-RED (user guide): <https://nodered.org/docs/user-guide/>

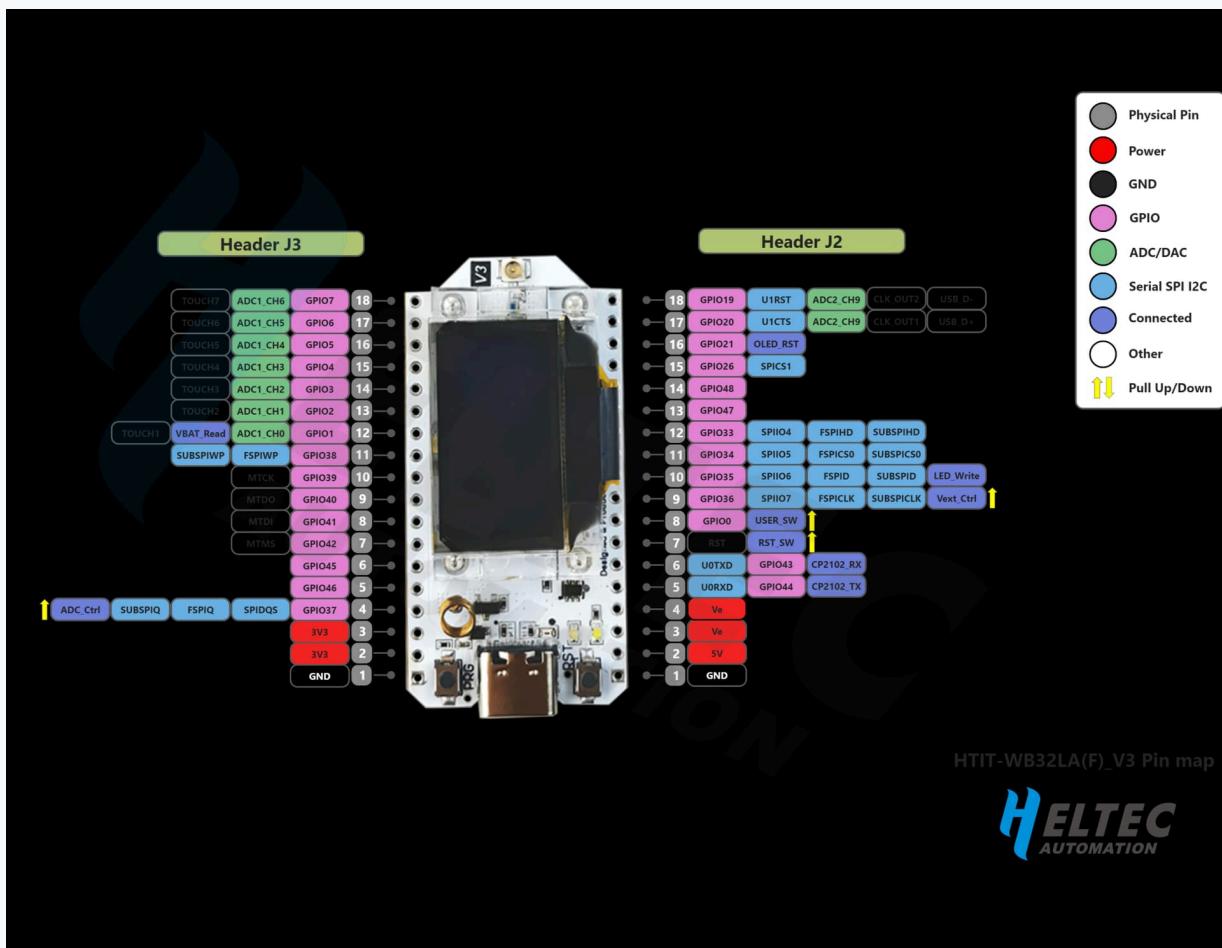
Rappel – Préparation

Infrastructure visée



Device réel: carte Heltec ESP32 WiFi LoRa V3 (Arduino IDE). Passerelle physique: Seeed WM1302 + Raspberry Pi 4 (fournie et préconfigurée).

Carte de développement – Heltec ESP32 WiFi LoRa V3

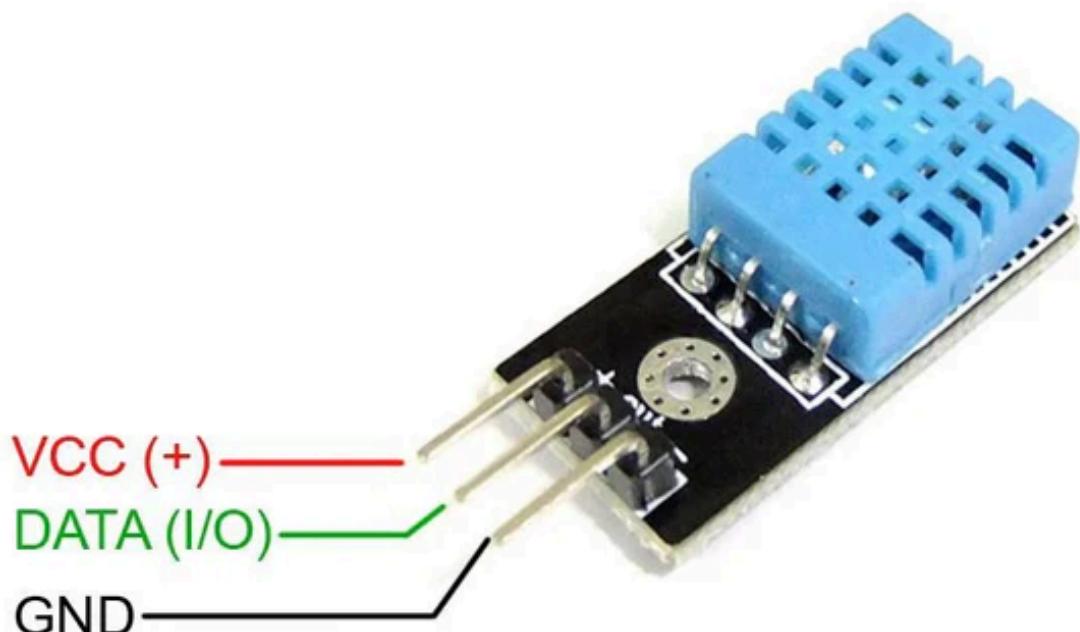


Rôle La carte Heltec WiFi LoRa V3 est le device LoRaWAN du Projet: elle mesure, encode et transmet les données (uplink) via LoRaWAN.

À retenir

- Programmation via Arduino IDE (USB).
- Arduino IDE: installer/configurer le package de cartes Heltec_ESP32 (Heltec ESP32 dev-boards), puis sélectionner la carte Heltec V3 correspondante.
- GPIO: le capteur DHT11 est câblé sur le GPIO 42 (à définir dans le code).
- LoRa: la radio est intégrée à la carte; pas de câblage externe côté LoRa.

Capteur – DHT11



Résumé Le DHT11 est un capteur numérique de température et d'humidité relative. Il fournit une mesure à faible cadence (typ. 1 Hz), adaptée à une démonstration de bout en bout (capteur → LoRaWAN → application).

Arduino IDE – Bibliothèques

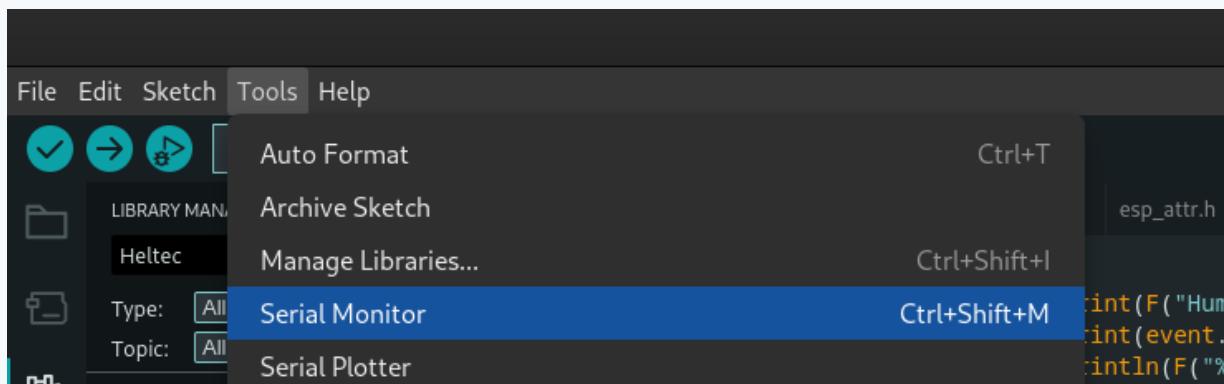
- Installer `DHT sensor library` (Adafruit) via le Library Manager.
- Installer aussi `Adafruit Unified Sensor` (dépendance).

Câblage avec la Heltec

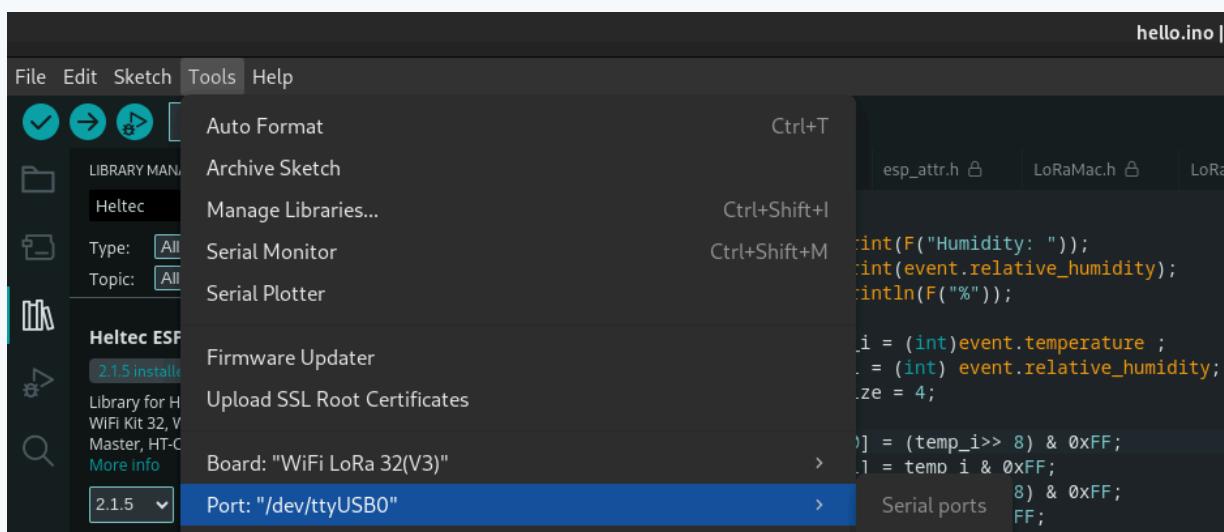
- VCC → 3V3, GND → GND.
- DATA → GPIO 42 (câblage déjà effectué).
- Aucune résistance à ajouter: le DHT11 est déjà monté sur un support/module adéquat (pull-up intégrée).

Guide – Visuels

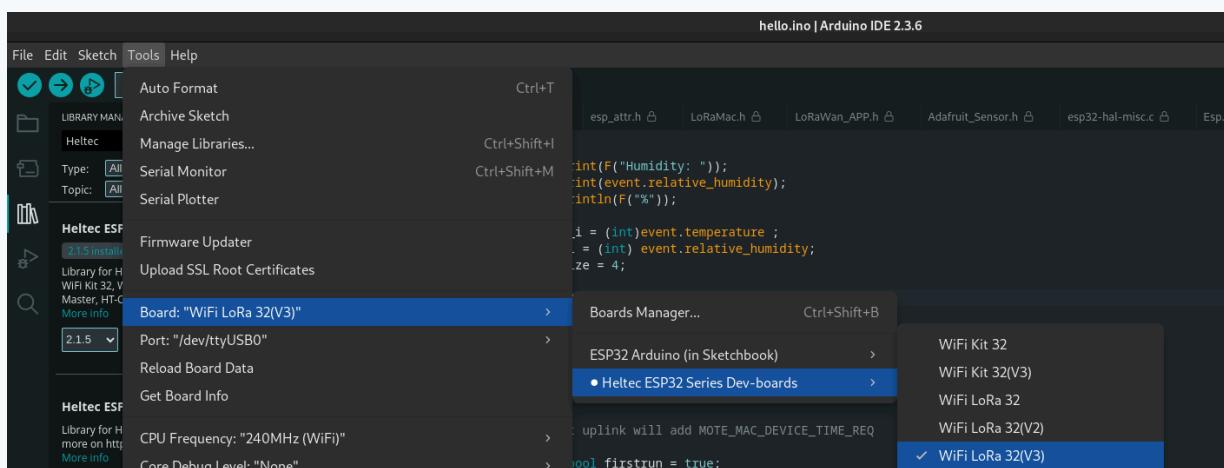
Serial Monitor



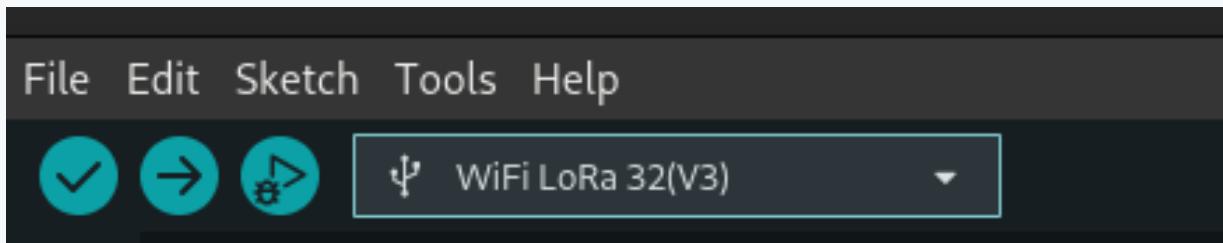
Selectionner le Port USB



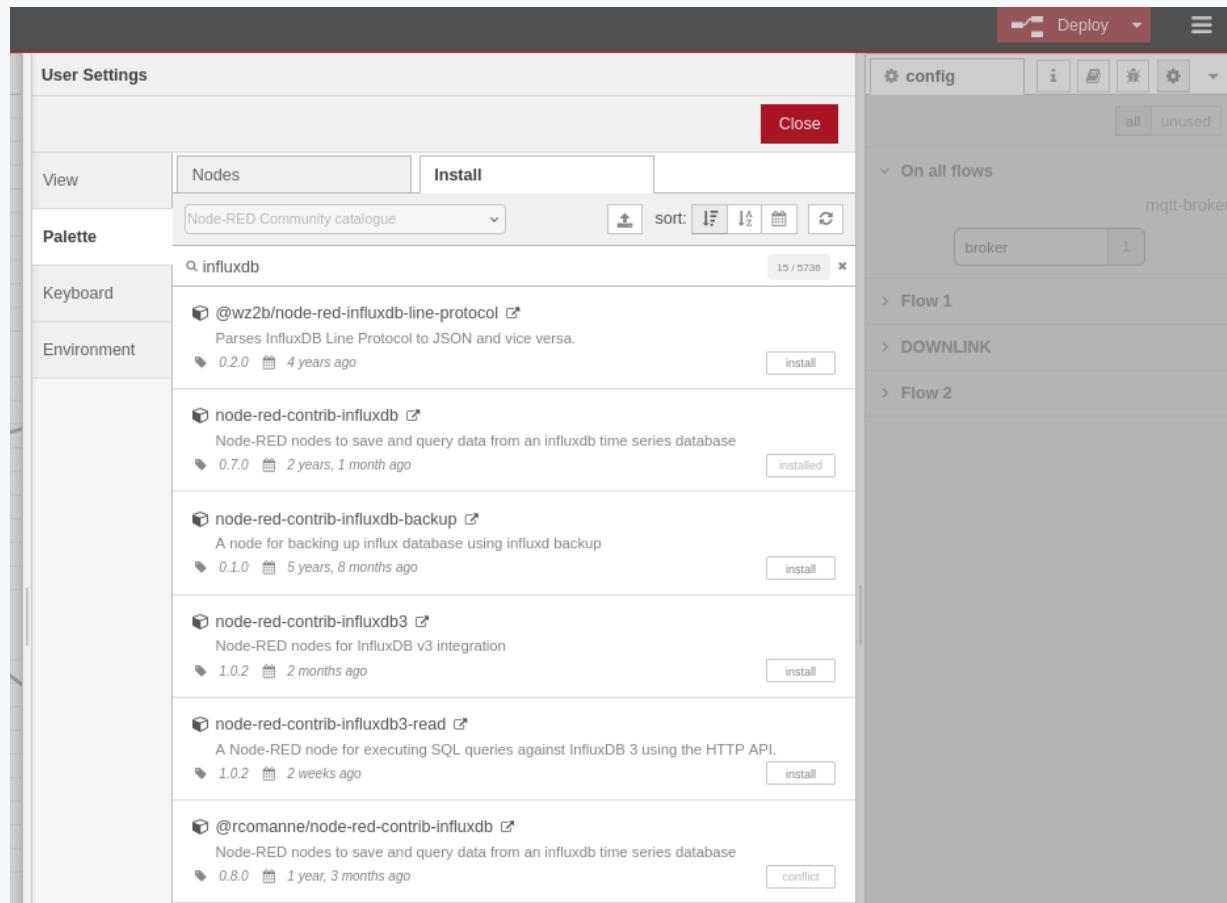
Selectionner la carte



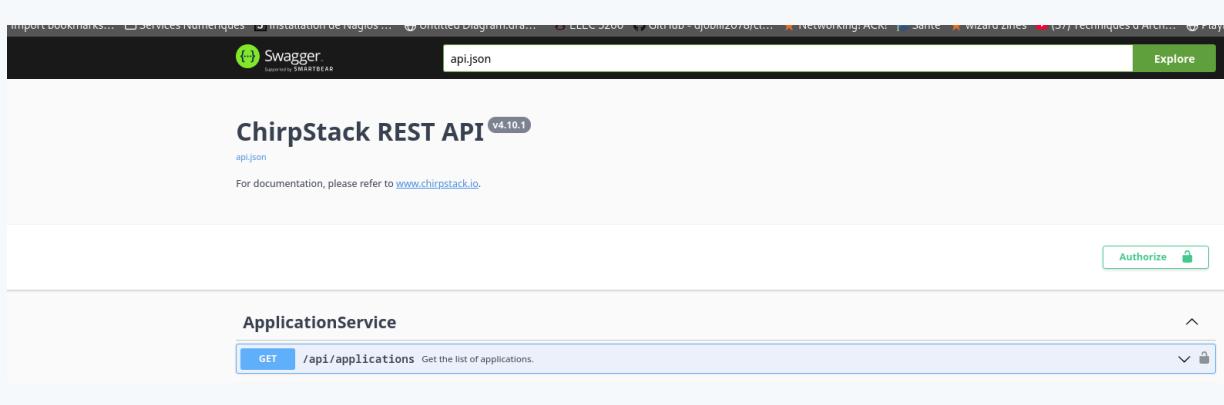
Compiler et flasher



Installer un plugin Node-RED



ChirpStack REST-API



The screenshot shows the ChirpStack REST API documentation. At the top, there's a navigation bar with the ChirpStack logo, a 'Swagger' button, a 'api.json' link, and a 'Explore' button. Below the header, the title 'ChirpStack REST API v4.10.1' is displayed, along with a note: 'For documentation, please refer to www.chirpstack.io'. A large central box contains the API documentation for the 'ApplicationService'. It shows a 'GET /api/applications' endpoint with the description: 'Get the list of applications.' There are also 'Authorize' and lock icons at the top right of this box.

Sensor - Packet Payload



The screenshot shows an Arduino IDE code editor with several files open in tabs: hello.ino, Arduino.h, esp_attr.h, LoRaMac.h, LoRaWan_APP.h (the active tab), Adafruit_Sensor.h, esp32-hal-misc.c, and Esp.h. The LoRaWan_APP.h file contains C code for a LoRaWAN application. A red dashed box highlights the `extern uint8_t appData[LORAWAN_APP_DATA_MAX_SIZE];` line. To the right of the code, handwritten text in red says: "cette variable contient la donnée à émettre vous devez la programmer dans le fichier projet.ino".

Encoded-data



Étape – Configuration du device (OTAA/ABP)

Objectif

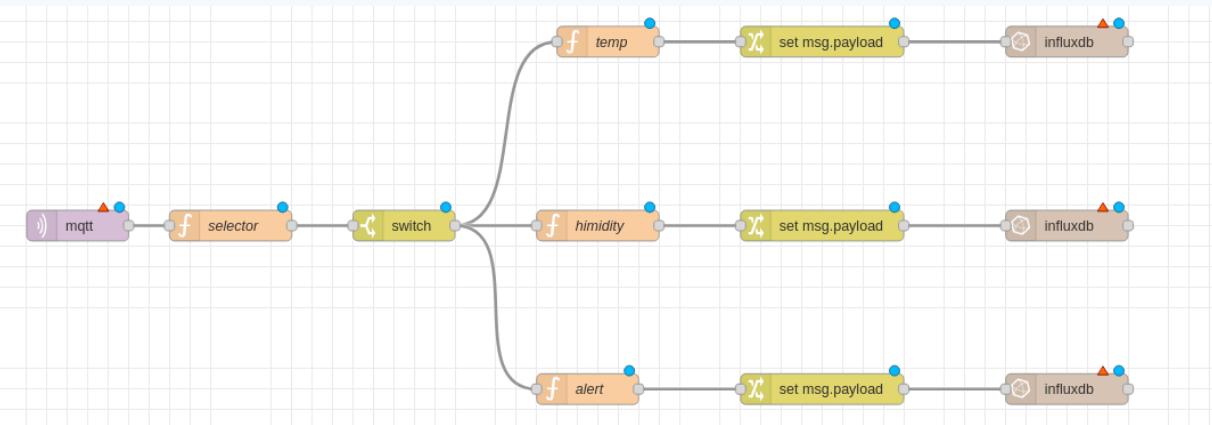
Configurer le device en mode OTAA (Over-the-air activation) ou ABP (Activation by personalization).

Spécification minimale (à respecter)

- Le device démarre avec une période par défaut (ex: 5s).
- Le device journalise la configuration choisie (Serial).
- Les données sont transmises via LoRaWAN.
- Node-RED log les données du capteur.

Étape – Traitement et ingestion des données

Objectif



Spécification minimale (à respecter)

- Les données sont sélectionnées et traitées par Node-RED.
- Créer les buckets par type de données (temperature, humidity, etc.) et les visualisations correspondantes.

Étape – Downlink: changer la fréquence d'envoi

Objectif Mettre en œuvre un downlink applicatif permettant de modifier, à la demande, la période d'envoi des mesures (intervalle entre deux uplinks).

Spécification minimale (à respecter)

- Le device démarre avec une période par défaut (ex: 15 s).

- L'utilisateur déclenche l'action via un POST sur une API REST exposée par Node-RED.
- Node-RED envoie ensuite un downlink vers le device (via l'API ChirpStack).
- Le downlink modifie la période utilisée par le device dès le cycle suivant.
- Le device journalise la nouvelle valeur (Serial) pour faciliter la validation.

API REST Node-RED (exemple)

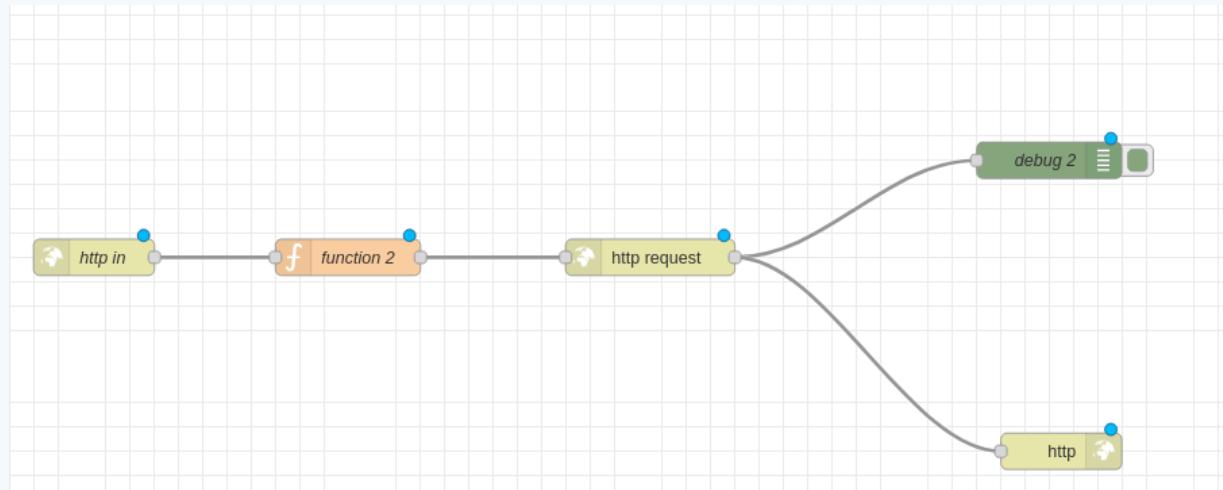
- Endpoint: POST /api/device/period
- Body JSON:
 - devEui: DevEUI du device cible (hex, sans séparateurs)
 - period_s: nouvelle période en secondes (entier)
- Exemple (machine hôte): curl -X POST http://localhost:1880/api/device/period -H "Content-Type: application/json" -d '{"devEui":"0102030405060708","period_s":30}'
- Réponse: 200 OK avec la période appliquée ou un message d'erreur.

Format de downlink proposé

- FPort: 3 (config).
- Payload: 2 octets uint16 big-endian = période en secondes.
- Exemple: 0x00 0x1E → 30 s.

Flow Node-RED attendu (principe)

- http in (POST /api/device/period) → json → function (validation + conversion en 2 octets)
→ http request (enqueue downlink ChirpStack) → http response.
- Côté http request, utilisez le service chirpstack-rest-api (exposé sur le RPI4 en 192.168.200.1:8090, accessible depuis Node-RED via http://192.168.200.1:8090).



Validation attendue

- Déclencher le changement via un curl sur l'API Node-RED.
- Vérifier côté logs série que la nouvelle période est prise en compte.
- Observer la variation effective du rythme d'uplinks dans ChirpStack et/ou via souscription MQTT.

Étape – Alerte émise par le capteur

Objectif Déclencher une alerte côté device lorsque la température et/ou l'humidité dépasse un seuil, puis remonter cette alerte dans la stack (ChirpStack → MQTT → Node-RED → InfluxDB/Grafana).

Spécification minimale (à respecter)

- Définir 2 seuils: `temp_max` et `hum_max` (valeurs par défaut au choix).
- Si un seuil est dépassé, le device émet une trame d'alerte (uplink) et journalise l'événement (Serial).
- Node-RED transforme le message reçu en un JSON applicatif incluant un champ `alert_level` (ex: `normal/warning/critical`).

Indications d'implémentation

- Côté device: vous pouvez distinguer mesure périodique vs alerte par un FPort dédié (ex: 2 = mesure, 4 = alerte) ou par un champ(flag dans le payload.
- Côté Node-RED: déduire `alert_level` et publier vers votre pipeline habituel (InfluxDB / Grafana).

Validation attendue

- Provoquer une alerte (souffler sur le DHT11, chauffer légèrement, etc.) et observer la trame côté ChirpStack.
- Vérifier dans Node-RED l'apparition de `alert_level` et la persistance dans InfluxDB.
- Ajouter une visualisation (tableau ou courbe) des alertes dans Grafana/InfluxDB UI.

Annexe – code



```
/*
TP2 – Projet LoRaWAN (base à trous)
```

Matériel

- Device: Heltec ESP32 WiFi LoRa V3
- Capteur: DHT11 (déjà câblé sur GPIO 42)
- Gateway: Seeed WM1302 + Raspberry Pi 4 (fournie/préconfigurée)

À faire (vous)

- 1) Configurer l'IDE Arduino: "Heltec_ESP32 dev-boards"
- 2) Installer les libs: "DHT sensor library" (Adafruit) + "Adafruit Unified Sensor"
- 3) Renseigner les identifiants OTAA (DevEUI/JoinEUI/AppKey) ci-dessous
- 4) Compléter l'encodage du payload (fonction encodePayload)

```
*/
```

```
#define LORAWAN_DEVEUI_AUTO 0
```

```
#include "Arduino.h"
#include "LoRaWan_APP.h"
#include "heltec.h"
```

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
```

```
// -----
// CONFIGURATION CAPTEUR
// -----
static constexpr uint8_t DHTPIN = 42; // DATA DHT11 (déjà câblé)
static constexpr uint8_t DHATYPE = DHT11;
```

```
DHT_Unified dht(DHTPIN, DHATYPE);
uint32_t dhtMinDelayMs = 1000;
```

```
// -----
// CONFIGURATION LORAWAN (OTAA)
// -----
// TODO: Remplacer avec les valeurs fournies par le serveur LoRaWAN (format hex).
// Rappel: certains serveurs appellent appEui = joinEui.
```

```
uint8_t devEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
uint8_t appKey[] = { 0x00, 0x00 };
```

```
// TODO: configuration ABP
uint8_t nwkSKey[] = {};
uint8_t appSKey[] = {};
uint32_t devAddr = 0x00000000;
```

```
// N'autoriser QUE le canal 5.
// Indice : canal 5 => bit 5 (valeur 1<<5) dans userChannelsMask[0].
uint16_t userChannelsMask[6] = { 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000 };
```

```
// Région LoRaWAN: dépend du choix dans l'IDE Arduino.
LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;
DeviceClass_t loraWanClass = CLASS_A;
```

```
// Périodique d'envoi par défaut (en ms).
uint32_t appTx DutyCycle ;
```

```
// TODO OTAA
bool overTheAirActivation = false;
```

Ceci est un devoir

No chatgpt



```
// TODO:ADR
bool loraWanAdr = true;

// TODO: Choisir confirmed/unconfirmed .
bool isTxConfirmed = false;

uint8_t appPort = 2;

//TODO: configurer les trials
uint8_t confirmedNbTrials = -1;

// -----
// DOWNLINK (CHANGEMENT DE PÉRIODE)
// -----
static constexpr uint8_t DOWNLINK_PORT_PERIOD = 3;
static constexpr uint16_t MIN_PERIOD_S = 5;
static constexpr uint16_t MAX_PERIOD_S = 3600;

// -----
// ALERTES (SEUILS)
// -----
static constexpr uint8_t UPLINK_PORT_MEASUREMENT = 2;
static constexpr uint8_t UPLINK_PORT_ALERT = 4;
static constexpr float TEMP_MAX_C = 30.0f;
static constexpr float HUM_MAX_PCT = 80.0f;
static bool alertActive = false;

static bool readDht(float &temperatureC, float &humidityPct) {
    delay(dhtMinDelayMs);

    sensors_event_t tempEvent;
    dht.temperature().getEvent(&tempEvent);
    if (isnan(tempEvent.temperature)) {
        Serial.println(F("Erreur lecture temperature (DHT11)."));
        return false;
    }

    sensors_event_t humEvent;
    dht.humidity().getEvent(&humEvent);
    if (isnan(humEvent.relative_humidity)) {
        Serial.println(F("Erreur lecture humidite (DHT11)."));
        return false;
    }

    temperatureC = tempEvent.temperature;
    humidityPct = humEvent.relative_humidity;
    return true;
}

static void encodePayload(float temperatureC, float humidityPct) {
    // TODO: Adapter l'encodage (ex: x100, CayenneLPP, JSON si gateway/app le supporte, etc.)
    // Encodage par défaut: int16 big-endian, température/humidité * 100.
    int16_t temperatureCent = (int16_t)(temperatureC * 100.0f);
    int16_t humidityCent = (int16_t)(humidityPct * 100.0f);

    appDataSize = 5;
}
```

Ceci est un devoir

NO chatgpt



```

static int prepareTxFrame(uint8_t port) {
    (void)port;

    float temperatureC = NAN;
    float humidityPct = NAN;
    if (!readDht(temperatureC, humidityPct)) {
        return -1;
    }

    const bool alertNow = (temperatureC > TEMP_MAX_C) || (humidityPct > HUM_MAX_PCT);

    Serial.print(F("Temperature: "));
    Serial.print(temperatureC);
    Serial.println(F(" °C"));
    Serial.print(F("Humidite: "));
    Serial.print(humidityPct);
    Serial.println(F(" %"));

    //TODO: gérer l'alerte
    if (alertNow) {
        if (!alertActive) {
            Serial.println(F("ALERTE: seuil depasse (envoi uplink alerte)."));
        }
    } else {
        if (alertActive) {
            Serial.println(F("Retour a la normale."));
        }
    }
}

encodePayload(temperatureC, humidityPct);
return 0;
}

// Heltec LoRaWAN: callback appelé à la réception d'un downlink.
void downLinkDataHandle(McpsIndication_t *mcpsIndication) {
    if (mcpsIndication == nullptr) {
        return;
    }

    if (!mcpsIndication->RxData) {
        return;
    }

    if (mcpsIndication->Port != DOWNLINK_PORT_PERIOD) {
        return;
    }

    if (mcpsIndication->BufferSize < 2) {
        Serial.println(F("Downlink periode invalide: payload < 2 octets."));
        return;
    }

    const uint16_t periodSeconds =
        ((uint16_t)mcpsIndication->Buffer[0] << 8) | (uint16_t)mcpsIndication->Buffer[1];

    if (periodSeconds < MIN_PERIOD_S || periodSeconds > MAX_PERIOD_S) {
        Serial.print(F("Downlink periode hors limites: "));
        Serial.print(periodSeconds);
        Serial.println(F(" s."));
        return;
    }

    appTxDutyCycle = (uint32_t)periodSeconds * 1000UL;

    Serial.print(F("Nouvelle periode d'envoi: "));
    Serial.print(periodSeconds);
    Serial.println(F(" s."));
}

```

Ceci est un devoir

No chatgpt



```

void setup() {
    Serial.begin(115200);

    // Init MCU (LoRaWAN system)
    Mcu.begin(HELTEC_BOARD, SLOW_CLK_TPYE);

    dht.begin();

    sensor_t sensor;
    dht.temperature().getSensor(&sensor);
    dhtMinDelayMs = sensor.min_delay / 1000;
    if (dhtMinDelayMs == 0) {
        dhtMinDelayMs = 1000;
    }
}

void loop() {
    switch (deviceState) {
        case DEVICE_STATE_INIT: {
#ifndef LORAWAN_DEVEUI_AUTO
            LoRaWAN.generateDeveuiByChipID();
#endif
            LoRaWAN.init(loraWanClass, loraWanRegion);
            LoRaWAN.setDefaultDR(3);
            break;
        }

        case DEVICE_STATE_JOIN: {
            LoRaWAN.join();
            break;
        }

        case DEVICE_STATE_SEND: {
            const int result = prepareTxFrame(appPort);
            if (result >= 0) {
                LoRaWAN.send();
            }

            deviceState = DEVICE_STATE_CYCLE;
            break;
        }

        case DEVICE_STATE_CYCLE: {
            txDutyCycleTime = appTxDutyCycle + randr(-APP_TX_DUTYCYCLE_RND, APP_TX_DUTYCYCLE_RND);
            LoRaWAN.cycle(txDutyCycleTime);
            deviceState = DEVICE_STATE_SLEEP;
            break;
        }

        case DEVICE_STATE_SLEEP: {
            LoRaWAN.sleep(loraWanClass);
            break;
        }

        default: {
            deviceState = DEVICE_STATE_INIT;
            break;
        }
    }
}

```

Ceci est un devoir

no chatgpt

No chatgpt

Jalon

- 3 séances max

Rendu

Livrable pour le groupe

- projet.ino fonctionnel
- capture démontrant le bon fonctionnement
- export flow Node-RED
- export données InfluxDB (csv)
- export dashboard Grafana

Evaluation individuelle

- Une orale de 5 minutes sur le projet.
- l'ordre de passage sera déterminé suivant l'ordre d'arrivée de vos rapports.