un nœud de capteurs autonome

transmission de données via LoRa et alimentation par cellules solaires

Saad Imtiaz (Elektor)

Ce projet de nœud de capteurs, exploite la technologie LoRa pour la transmission de données à faible consommation énergétique et à longue portée, le rendant idéal pour la surveillance environnementale à distance. Utilisant le module LoRa WIO E5 et une carte contrôleur ESP32-C3 XIAO, il intègre un capteur de CO2 SCD30 et un capteur d'humidité du sol, tous alimentés par énergie solaire. Les données collectées sont transmises au réseau The Things Network (TTN) et peuvent être consultées via Datacake ou Home Assistant, offrant des informations en temps réel et des possibilités d'automatisation. Ce guide détaillé vous accompagnera à travers toutes les étapes, depuis la mise en place de votre passerelle LoRaWAN jusqu'à son intégration fluide dans TTN, Datacake, et Home Assistant.

Source: Adobe Stock

Figure 1. Nœud de capteurs LoRaWAN alimenté par l'énergie solaire.

La surveillance à distance de l'environnement est essentielle pour relever les défis du changement climatique et de la gestion des ressources. Cependant, de nombreux nœuds de capteurs LoRa disponibles sur le marché sont coûteux et manquent d'options de personnalisation, ce qui les rend peu pratiques pour des applications spécifiques et des zones éloignées. Pour surmonter ces limitations, ce projet présente un nœud de capteur LoRa polyvalent et rentable. Ce système personnalisé offre une solution efficace pour la collecte de données à distance, en utilisant la communication LoRa à longue portée et l'énergie solaire pour assurer un fonctionnement continu. En intégrant des capteurs personnalisables, le système offre une grande flexibilité pour répondre aux différents besoins de surveillance environnementale.

L'une des caractéristiques de ce projet est son intégration dans la plateforme d'automatisation open-source Home Assistant, qui offre des capacités d'automatisation avancées. Par exemple, en mesurant l'humidité du sol, il est possible d'automatiser l'irrigation du jardin, ce qui peut soulager les propriétaires du fardeau de l'arrosage manuel.

Dans cet article, nous allons explorer le processus d'installation d'un nœud de capteurs LoRaWAN, illustré à la **figure 1**, depuis la configuration d'une passerelle LoRaWAN et l'intégration au réseau The Things Network (TTN) jusqu'à la visualisation des données sur des plateformes telles que Datacake et Home Assistant. Ce projet montre comment une technologie abordable et adaptable peut offrir des solutions de surveillance complètes. Il pallie les insuffisances des options actuellement disponibles sur le marché et ouvre de nouvelles perspectives pour des applications plus étendues.

Aperçu du système

Le cœur de ce projet est la carte microcontrôleur XIAO ESP32-C3 de Seeed Studio. Vous pouvez acquérir cette carte ainsi que les autres modules nécessaires pour ce projet chez Elektor (voir l'encadré Produits). Le choix de la carte XIAO a été fait en raison de sa taille compacte et de ses capacités d'E/S adéquates pour ce projet. Pour étendre ses options d'E/S, nus avons utilisé la carte eXpansion d'Elektor [1] qui augmente les fonctionnalités du microcontrôleur avec

LoRa, LoRaWAN, une passerelle et TTN ?

LoRa : LoRa (Long Range) est une technologie de communication sans fil spécialement conçue pour permettre la transmission de données à longue distance tout en minimisant la consommation énergétique. Opérant sur des fréquences sub-gigahertz (868 MHz en Europe et 915 MHz en Amérique du Nord), cette technologie est capable de transmettre des données jusqu'à 15 km dans des zones rurales et jusqu'à 5 km en milieu urbain. Elle est particulièrement adaptée pour des applications IoT telles que la surveillance environnementale ou l'agriculture intelligente.

LoRaWAN : LoRaWAN (Long Range Wide Area Network) est un protocole de communication qui exploite la technologie LoRa pour établir des connexions entre les appareils LoRa et les passerelles. Ce protocole est idéal pour les déploiements IoT à grande échelle, car il assure une communication sécurisée et bidirectionnelle, supporte des débits de données adaptatifs et optimise l'utilisation du réseau.

Passerelle : Une passerelle LoRaWAN sert de pont entre les appareils LoRa et l'internet, recevant les données des appareils et les relayant à un serveur central via des connexions Wi-Fi, Ethernet ou cellulaires. Ces passerelles peuvent couvrir de vastes zones et gérer de multiples transmissions simultanées d'appareils.

TTN (The Things Network) : The Things Network est une initiative mondiale qui propose un réseau LoRaWAN open-source. Il fournit une infrastructure complète pour connecter des appareils LoRaWAN, incluant l'enregistrement des appareils, le routage des données et l'intégration avec des plateformes telles que Datacake et Home Assistant. Grâce à son large soutien communautaire, TTN constitue une ressource précieuse pour le déploiement d'applications IoT via LoRaWAN.

six connexions I²C. Bien que ce projet n'utilise qu'une seule de ces connexions I²C, la flexibilité de ce nœud de capteurs permet l'intégration d'autres capteurs au besoin. La **figure 2** présente le schéma fonctionnel du projet.

Pour la communication LoRa, nous avons choisi le module E5 WIO LoRa de Seeed Studio, contrôlable via une connexion UART et des commandes AT pour l'envoi de données et la configuration. Sa facilité de configuration et son coût raisonnable en ont fait une option appropriée. Ce module assure une transmission fiable des données à longue distance. Au cours de mes essais, j'ai pu obtenir une portée d'environ 700 m dans un environnement urbain malgré la présence d'arbres et de bâtiments. Il est important de noter que ma passerelle était située à l'intérieur, près d'une fenêtre, ce qui est considéré comme une bonne performance. Comme pour tout système de communication sans fil, la ligne de vue est cruciale. La portée pourrait être considérablement augmentée en plaçant la passerelle en hauteur avec une antenne appropriée. L'efficacité du module pourrait également être améliorée en utilisant une meilleure antenne.

Le système de recharge solaire utilise un panneau solaire 3 W de Seeed Studio, connecté à un module de gestion de l'énergie solaire de Waveshare qui est un tracker MPPT avec quelques circuits de protection. Cette configuration permet de charger deux batteries 18650 en parallèle. Le tracker MPPT (*Maximum Power Point Tracking*) est essentiel pour optimiser la puissance produite par le panneau solaire.

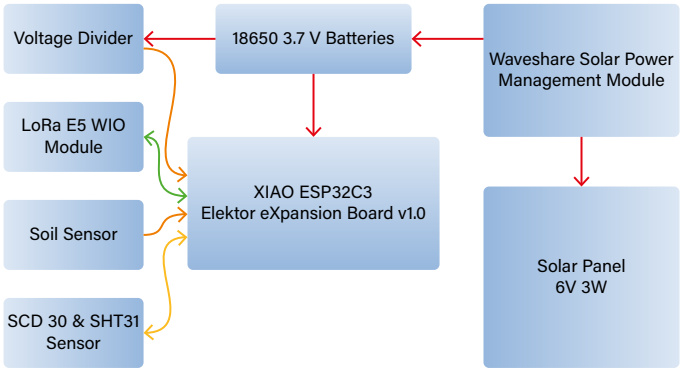

Figure 2. Schéma fonctionnel du projet.

Il ajuste continuellement le point de fonctionnement électrique du panneau solaire, pour maximiser son efficacité. Après avoir testé plusieurs modules de suivi MPPT, le module Waveshare s'est avéré être le plus performant et le plus fiable pour cette application.

Nous avons choisi le capteur SCD30 de Seeed Studio sa grande précision dans la détection du CO_2. Le SCD30 est un capteur NDIR, ce qui en fait un véritable capteur de CO_2, capable de mesurer la concentration en CO_2 en ppm dans l'air ambiant. Un autre aspect intéressant de ce capteur est qu'il est livré avec un capteur de température et d'humidité SHT31 déjà intégré. Le SCD30 n'étant pas conçu pour être utilisé à l'extérieur, un boîtier Stevenson imprimé en 3D a été créé pour le protéger des éléments extérieurs.

Le capteur d'humidité du sol utilisé dans ce projet est également un produit de de Seeed Studio. Bien qu'il ait été choisi en raison de sa disponibilité, il n'est pas idéal pour les applications extérieures car ses connecteurs et son circuit imprimé ne sont pas étanches, comme le montre la **figure 3**. Dans les prochaines itérations du projet, il sera envisageable de choisir un capteur d'humidité du sol plus adapté aux conditions extérieures.

Figure 3. Le capteur de sol. Il peut provoquer un court-circuit s'il attrape de l'humidité.

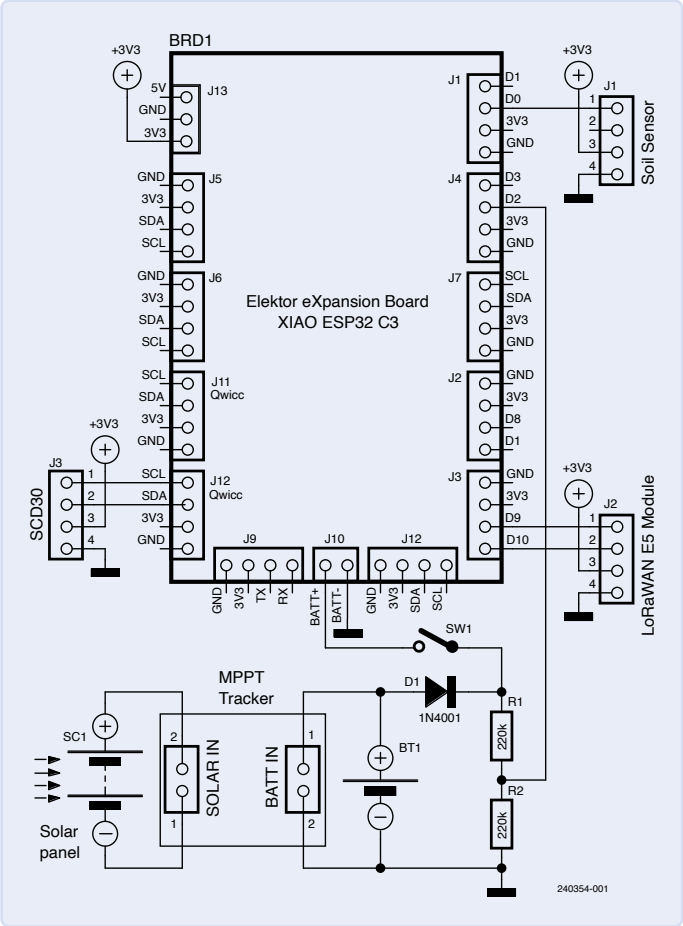

Figure 4. Schéma du projet.

Le microcontrôleur XIAO ESP32-C3 de Seeed Studio communique avec le capteur SCD30 via I²C et avec le module LoRa E5 via UART. Le capteur d'humidité du sol est connecté à une broche analogique et la tension de la batterie est régulée avec un circuit diviseur de tension. Le schéma de circuit du projet est représenté dans la **figure 4**.

Configuration de la passerelle LoRaWAN

Avant d'aborder les aspects techniques plus complexes du projet, commençons par l'élément principal : la configuration de la passerelle LoRaWAN. Comme nous utilisons LoRaWAN pour transmettre les données des capteurs, il est essentiel d'avoir une passerelle LoRaWAN à proximité afin que notre nœud de capteurs puisse envoyer ses données collectées sur Internet. Bien qu'il soit possible de trouver des passerelles LoRaWAN disponibles autour de vous, il se peut que vous ayez besoin d'installer votre propre passerelle si aucune n'est présente à proximité. La configuration de votre propre passerelle LoRaWAN est simple. Tout d'abord, vous devez acquérir une passerelle ; j'ai choisi la passerelle LoRaWAN LPS8v2 de Dragino. Pour utiliser votre passerelle, vous devez la connecter à un réseau LoRaWAN. J'ai choisi le réseau The Things Network (TTN), mais il est également possible de se connecter à d'autres réseaux tels que Helium ou Datacake, et même utiliser deux réseaux simultanément.

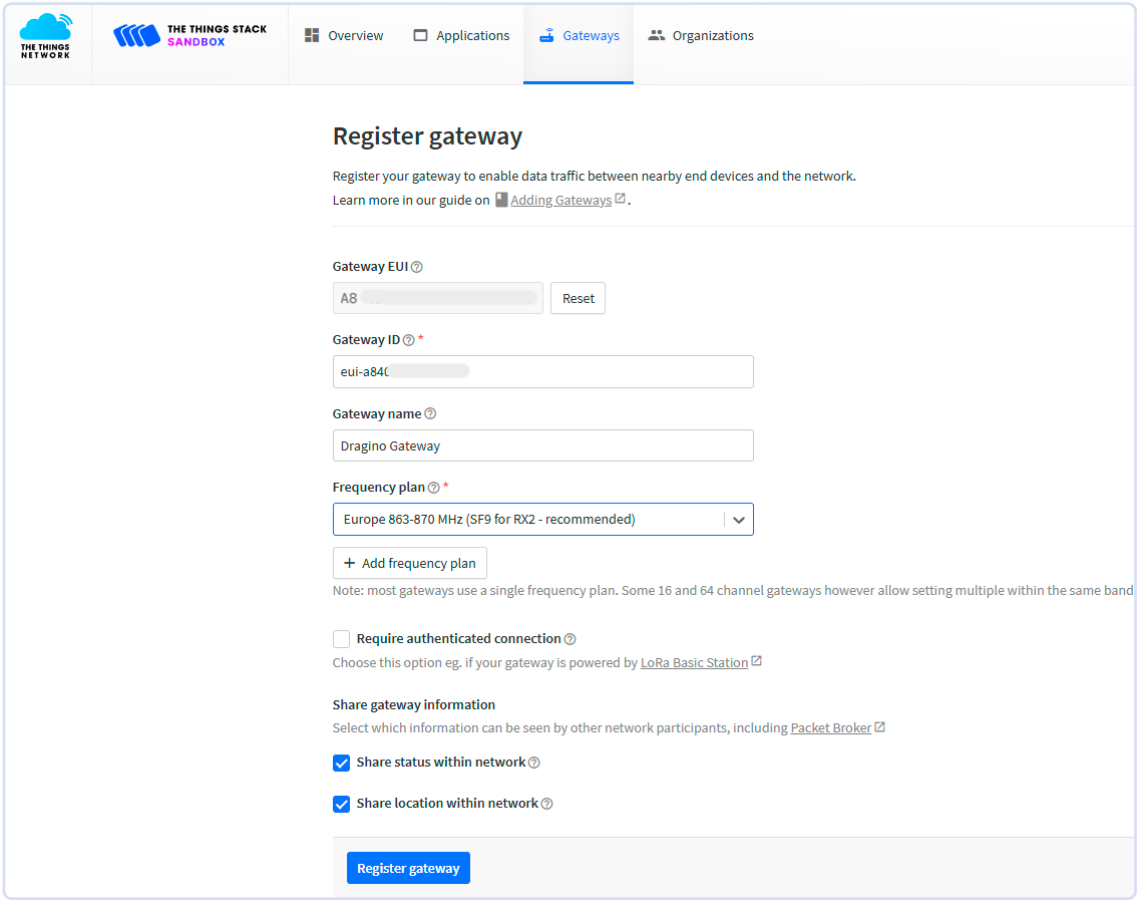

Figure 5. Ajout de votre passerelle au réseau.

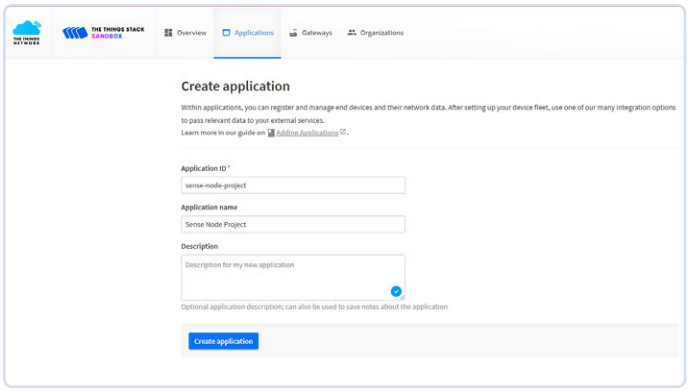

Figure 6. Création de l'application de la console TTN.

Voici un bref aperçu de la procédure de configuration :

- **Connexion au WiFi** : Commencez par mettre votre passerelle LoRaWAN sous tension et connectez-la au réseau Wifi de votre domicile. Cela permet à la passerelle d'accéder à Internet.
- **S'enregistrer sur TTN** : Ensuite, rendez-vous sur la console The Things Network et créez un compte si vous n'en avez pas déjà un. Une fois connecté, enregistrez votre passerelle en fournissant les informations nécessaires telles que l'EUI (identifiant unique de votre passerelle) et en sélectionnant votre région, comme le montre la **figure 5**.
- **Configuration de la passerelle** : Suivez les instructions fournies par TTN pour configurer votre passerelle. Cette étape inclut généralement la saisie des paramètres réseau, la sélection du plan de fréquences approprié et la configuration de la passerelle pour qu'elle communique avec les serveurs de TTN.
- **Dernières étapes** : Après la configuration, la passerelle devrait se connecter au TTN et être prête à recevoir des données de vos nœuds LoRa. Vous pouvez surveiller l'état de votre passerelle sur la console TTN pour vous assurer qu'elle fonctionne correctement.

Il existe de nombreux guides détaillés disponibles en ligne [2][3] qui fournissent des instructions étape par étape pour configurer différents types de passerelles LoRaWAN. Suivre ces guides peut vous aider à résoudre les problèmes et à vous assurer que votre passerelle est correctement configurée et connectée. Une fois la passerelle LoRaWAN configurée, nous passerons à l'enregistrement de notre nœud de capteurs sur le réseau The Things Network.

IIntégration de votre projet au réseau The Things Network

Une fois votre passerelle configurée, l'étape suivante consiste à intégrer le module LoRa E5 au réseau The Things Network (TTN) [4]. Pour ce faire, commencez par enregistrer votre appareil sur la console TTN, vous devez d'abord créer une nouvelle application, puis ajouter votre appareil. Sur la console TTN cliquez sur *Create application*, donnez un ID et un nom à votre application, puis de cliquez sur *Create application*, comme le montre la **figure 6**.
Ensuite, il est nécessaire d'enregistrer votre appareil au sein de l'application que vous venez de créer. Rendez-vous dans votre application puis cliquez sur *Register end device.* Au cours de cette étape, vous obtiendrez des informations d'identification essentielles telles que *Device EUI*, *Application EUI*, et *App Key*, comme le montre la **figure 7**. Ces informations sont indispensables pour configurer votre appareil afin qu'il puisse communiquer avec le TTN et seront utilisées dans le code Arduino.

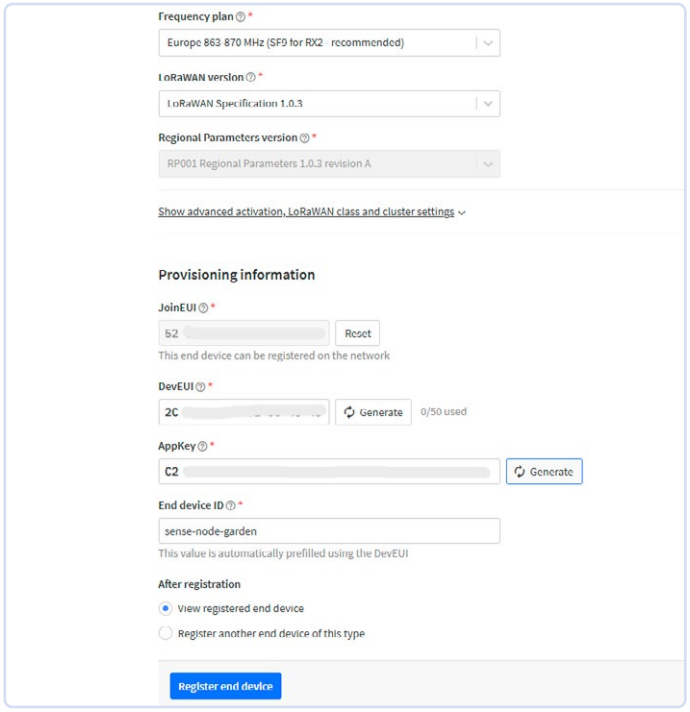

Figure 7. Enregistrement de l'appareil final sur l'application.

Après avoir enregistré votre appareil dans la console TTN, vous devez configurer le format de la charge utile afin que le TTN interprète correctement les données envoyées par votre appareil. TTN vous permet de définir un formateur de charge utile personnalisé en JavaScript, qui décode les données brutes envoyées par votre appareil.
Voici un formateur de charge utile en JavaScript :

```
function Decoder(bytes, port) {
  var decoded = {};
  if (port === 8) {
    decoded.soilMoisture = (bytes[0] << 8) | bytes[1];
    decoded.temp = ((bytes[2] << 8) | bytes[3]);
    decoded.humi = (bytes[4] << 8) | bytes[5];
    decoded.co2 = (bytes[6] << 8) | bytes[7];
    decoded.battery = (bytes[8] << 8) | bytes[9];
  }
  return decoded;}
```

Pour mettre cela en œuvre :

- **Accédez à Payload Formats** : Dans la console TTN, accédez à votre application et sélectionnez l'onglet *Payload Formats*.
- **Sélectionnez Decoder Function** : Choisissez le type de fonction *Decoder*.
- **Insérez le code** : Copiez et collez le code JavaScript fourni dans l'éditeur de la fonction de décodage.
- **Enregistrer les modifications** : Enregistrez les modifications pour appliquer le décodeur à votre application.

Cette fonction de décodage traite les octets reçus de votre nœud de capteur et les convertit en valeurs lisibles, telles que l'humidité du sol, la température, l'humidité, les niveaux de CO_2 et la tension de la batterie.

Logiciel

Le logiciel est conçu pour collecter et transmettre efficacement les données des capteurs tout en gérant la consommation d'énergie grâce à des modes de veilles. Examinons de plus près les principaux composants du code et la manière dont ils interagissent.

Listage 1. Croquis Arduino (extrait).

```
#include <Arduino.h>
#include <SCD30.h>
#include <HardwareSerial.h>
#include <config.ino>

// Defining Hardware the second internal UART -
// Serial2 for the LoRaWAN E5 Module - Pin 9 and 10
HardwareSerial Serial2(1);

...
//*** Initializing variables ***//

void setup() {

  ...
  //*** Initializing Sensors, Serial 1 & Serial 2 ***//

  // Check if the AT command returns OK
  if (at_send_check_response("+AT: OK", 100, "AT\r\n")) {
    // Set the flag to indicate the LoRa module exists
    is_exist = true;
    // Send AT command to get AppEUI and check the response
    at_send_check_response("+ID: AppEui", 1000, "AT+ID\r\n");
    // Set the LoRa module to LWOTAA mode
    at_send_check_response("+MODE: LWOTAA", 1000, "AT+MODE=LWOTAA\r\n");
    // Set the data rate to EU868
    at_send_check_response("+DR: EU868", 1000, "AT+DR=EU868\r\n");
    // Set the channel number range
    at_send_check_response("+CH: NUM", 1000, "AT+CH=NUM,0-2\r\n");
    // Set the APP Key for authentication, replace with your generated APP Key from TTN
    at_send_check_response("+KEY: APPKEY", 1000,
             "AT+KEY=APPKEY,\"C2XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\"\r\n");
                         // Enter your generated APP Key here.
    // Set the LoRa module to Class A
    at_send_check_response("+CLASS: C", 1000, "AT+CLASS=A\r\n");
    // Set the port number to 8
    at_send_check_response("+PORT: 8", 1000, "AT+PORT=8\r\n");
    // Delay to ensure all commands are processed
    delay(200);
    // Print confirmation that the LoRaWAN setup is complete
    Serial.println("LoRaWAN");
    // Set the flag to indicate the module has joined the network
    is_join = true;
  }
  else {
    is_exist = false;
    Serial.print("No LoRa E5 module found.\r\n");
  }
}

void loop() {
  getSensors();
  sendData();
  // Configure the wake-up source and duration for deep sleep
  esp_sleep_enable_timer_wakeup(10 * 60 * 1000000);
                   // 10 minutes in microseconds
  // Enter deep sleep mode
  esp_deep_sleep_start();
}
```

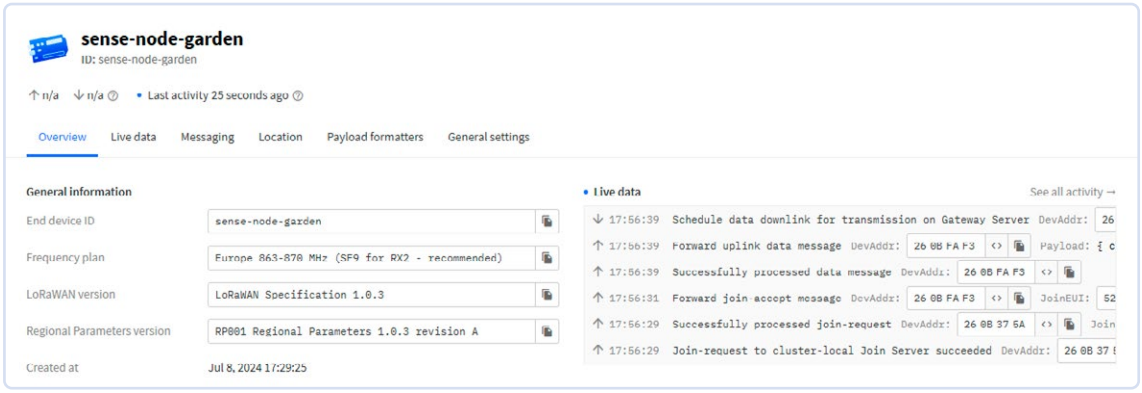

Figure 8. Les données envoyées par le nœud de capteurs sont affichées dans la section d'aperçu du dispositif de TTN.

Le code commence par l'inclusion des bibliothèques nécessaires et définir les configurations matérielles. Dans la fonction `setup`, nous initialisons la communication série, configurons les capteurs et le module LoRa. Ceci implique de définir les paramètres du module LoRa E5, notamment d'entrer l'*App Key* générée sur la console TTN (voir ci-dessous). Le **listage 1**, présente une version simplifiée du code, tandis que l'intégralité du code et des fichiers matériels est disponible sur le dépôt GitHub du projet [5].

Les principales opérations se déroulent dans la fonction `loop`. Elle recueille les données des capteurs, les envoie au TTN, puis met le microcontrôleur en mode « deep sleep » pour économiser de l'énergie. L'utilisation du mode « deep sleep » réduit considérablement la consommation d'énergie en désactivant la plupart des fonctionnalités du microcontrôleur et en ne le réveillant uniquement pour effectuer les tâches essentielles. Lorsqu'il est en veille profonde, le XIAO ESP32-C3 consomme environ 142 µA au lieu de 8 mA, ce qui augmente significativement l'autonomie de la batterie. En mode « deep sleep », l'ESP32C3 consomme environ 5 µA et le reste du courant est consommé par le circuit intégré de gestion de la tension et de la batterie integré. Le microcontrôleur s'active à intervalles réguliers pour collecter et transmettre des données avant de retourner en veille.

La fonction `getSensors()` lit les valeurs du capteur d'humidité du sol et du capteur SCD30. Elle calcule également la tension de la batterie en utilisant un diviseur de tension et effectue la moyenne de plusieurs lectures pour améliorer la précision. En effectuant plusieurs lectures, le système assure des résultats de mesure stables et précis.

La fonction `sendData()` formate les données de capteur collectées dans une charge utile et les envoie au TTN via le module LoRa E5. Elle convertit essentiellement les données du capteur en une chaîne hexadécimale prête à être transmise via LoRa, comme le montre l'extrait de code ci-dessous. Cette fonction garantit que les données sont correctement formatées et transmises via LoRa en établissant d'abord une connexion à la passerelle la plus proche et en envoyant les données du capteur, ce qui permet une surveillance en temps réel. Après l'envoi des données, le module LoRa E5 est mis en mode veille pour réduire la consommation énergétique.

```
// Prepare and send the sensor data
// as a hexadecimal string via LoRa
char cmd[128];
sprintf(cmd, "AT+CMSGHEX=
   \"%04X%04X%04X%04X%04X\"\r\n",
   (int)soilMoisturePercent, (int)temp,
   (int)humi, (int)co2, (int)battery);
```

App Key

Avant de charger le code, vous devez ajouter la clé d'application générée lors de l'ajout de votre appareil à l'application sur The Things Network (TTN). Cette clé est essentielle car elle sert à authentifier votre appareil auprès du réseau TTN. Elle est saisie dans la fonction `setup` et utilisée dans la fonction `at_send_check_response`.

```
at_send_check_response("+KEY: APPKEY", 1000,
"AT+KEY=APPKEY,\"XXXXXXXXXXXXXXXXX\"\r\n");
// Enter your generated APP Key here
```

This section of the code ensures that your device is registered and authenticated, enabling it to send data to TTN. if all is done correctly, you can see the forward uplink messages from your LoRa Sense Node on your TTN device live data section, as shown in **Figure 8**. You can find the entire project repository on the GitHub [5] which includes all the files for this project.

Consommation d'énergie

Tout système fonctionnant sur batterie nécessite une gestion rigoureuse de la consommation d'énergie afin d'éviter les recharges fréquentes, toutes les deux ou trois jours par exemple. De plus, les systèmes destinés à un usage extérieur devraient nécessiter un minimum d'entretien. Mesurer la consommation d'énergie, tester le courant consommé par chaque composant et évaluer le courant moyen du système, puis calculer les coulombs pour estimer la durée de vie de la batterie sont des tâches complexes. Pour faciliter ces tâches, il est possible d'utiliser des outils tels que les profileurs de puissance ou les analyseurs d'énergie DC pour obtenir des mesures précises et analyser la consommation d'énergie de chaque composant.

J'ai récemment mis la main sur un Joulescope JS200 [6], qui a grandement simplifié ce processus. Sa taille compacte et son interface conviviale en font l'outil idéal pour tous les tests de consommation électrique.

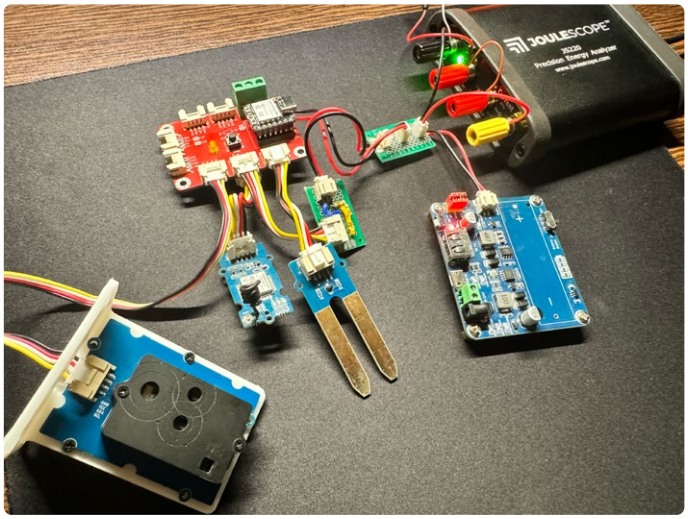

Figure 9. Consommation d'énergie du projet testé avec un JouleScope.

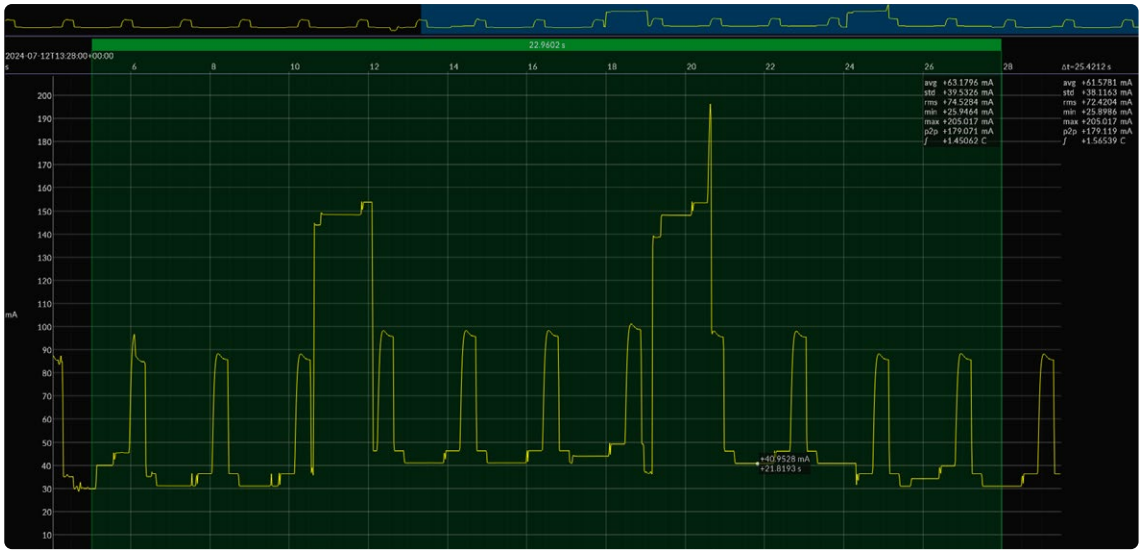

Figure 10. Forme d'onde du courant du SCD30 et du module LoRa.

Initialement, comme tout ingénieur, mon système n'était pas optimisé en termes de consommation d'énergie ; la priorité était de vérifier que tous les composants fonctionnaient correctement ensemble. L'optimisation du code et de la consommation d'énergie vient ensuite.

Après avoir assemblé le système initial, je l'ai connecté au Joulescope, comme le montre la **figure 9**. J'ai immédiatement remarqué que la LED IR du capteur SCD30, utilisée pour la mesure du CO_2, consommait près de 68 mA toutes les 2 secondes, ce qui résultait en un courant moyen de 25,3 mA sur 30 secondes, comme le montre la **figure 10**. En examinant la fiche technique et la bibliothèque fournie, j'ai trouvé que l'intervalle de mesure par défaut était réglé sur 2 s. C'était une solution facile car l'intervalle de mesure peut être contrôlé, de sorte que la LED IR ne s'allume que lorsqu'une mesure est nécessaire. Cela réduit les pics de courant, diminuant ainsi l'appel de courant moyen. Cependant, le capteur SCD30 consomme toujours environ 4,5 mA même lorsqu'il est inactif, et je n'ai pas encore trouvé de solution logicielle pour le réduire.

Ensuite, je me suis concentré sur le module LoRa E5 WIO, qui consomme un courant significatif de la batterie. Lors des opérations LoRa, le module consomme jusqu'à 110 mA, ce qui est acceptable, car cela ne dure qu'une seconde. Mais l'objectif est de minimiser la consommation de courant lorsque le système est en veille et que l'ESP32-C3 est en mode deep sleep. Lorsque le module est inactif, il peut consommer jusqu'à 10 mA, ce qui représente une valeur élevée pour un système alimenté par batterie sur de longues périodes. En explorant différentes ressources en ligne, j'ai découvert l'existence d'une commande AT qui peut être utilisée, pour mettre le module en mode veille, la consommation de courant de 51,7 µA : une amélioration notable. La **figure 11** illustre la forme d'onde de courant du module LoRa E5 avant et après l'activation du mode veille. Il est également possible de réduire davantage la consommation de courant du module LoRa peut être encore réduit jusqu'à 3 µA, en retirant le LDO intégré sur le module. Lors de tests ultérieurs, j'ai constaté que le capteur d'humidité du sol augmentait la consommation de courant du système jusqu'à 4 mA lorsqu'il mesurait l'humidité. Ce capteur utilise la capacité pour mesurer l'humidité, ce qui entraîne une consommation d'énergie excessive à mesure que le sol devient plus conducteur. Cela provoque également de la corrosion et l'accumulation de dépôts minéraux sur les sondes, du fait que les pistes nues du capteur fonctionnent comme un dispositif d'électrolyse. Pour atténuer le problème de la corrosion, il est possible d'utiliser un capteur de sol revêtu. Cependant, pour éliminer le courant excessif lorsque le niveau d'humidité augmente, il faut couper l'alimentation électrique du capteur lorsqu'il n'est pas utilisé, une solution principalement matérielle. Dans la première version du projet, j'ai privilégié une approche simplifiée, axée sur les optimisations logicielles. Les versions futures prévoiront l'intégration de solutions matérielles.

Autonomie de la batterie

Cette étape nous a permis de calculer la durée de vie de la batterie du système, ce qui a été grandement facilité grâce à l'utilisation du Joulescope. Après avoir appliqué toutes les optimisations de consommation d'énergie via le logiciel, j'ai connecté le système au Joulescope. Dans la **figure 12**, vous pouvez voir la forme d'onde de courant de l'ensemble du système après ces optimisations. Notez que le SCD30 allume la LED IR uniquement lorsque l'ESP32-C3 est éveillé, puis arrête de prendre des mesures une fois que l'ESP32-C3 passe en mode veille profonde. En utilisant sa fonction avancée dans son mode multimètre, j'ai mesuré le courant moyen et les coulombs consommés par le système sur 10 minutes, période durant laquelle de nouvelles valeurs des capteurs étaient envoyées via LoRa chaque minute. Comme le montre la **figure 13**, le courant moyen était de 19,10 mA, ce qui est élevé pour mon objectif de maintenir le système en fonctionnement pendant une semaine sur une seule charge. La réduction de la fréquence de lecture des

Figure 13. Capture d'écran du mode Multimètre dans JouleScope.

Figure 11. Consommation de courant du module LoRa E5 Modoule, avant (à gauche) et après (à droite) la mise en œuvre de la fonction sommeil.

Figure 12. Forme d'onde de courant du système entier, après les optimisations de puissance.

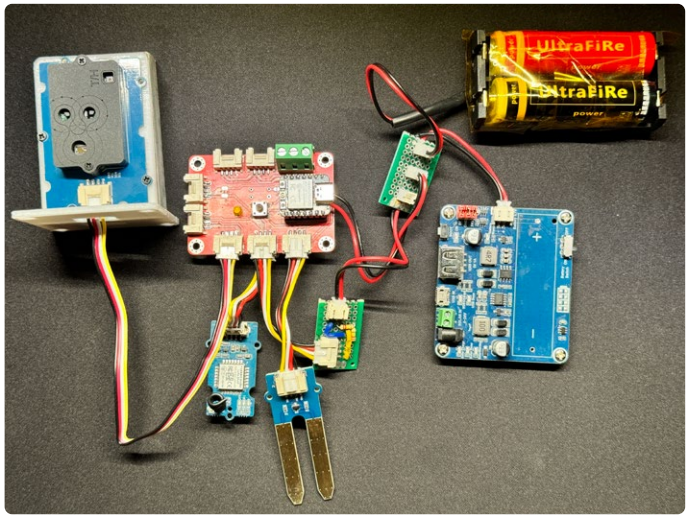

Figure 14. Tous les composants du projet assemblés et prêts à être placés dans le boîtier.

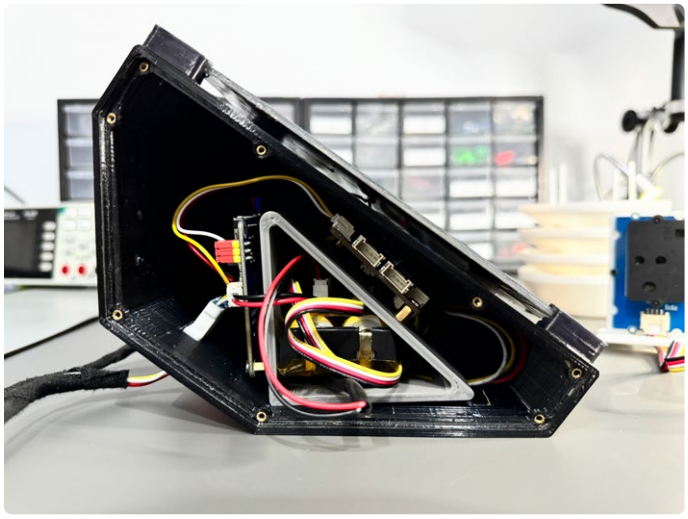

Figure 15. Composants placés dans e boîtier principal avec le panneau solaire monté sur le dessus.

données à toutes les 10 minutes a réduit le courant moyen à 4,905 mA, rendant ainsi la consommation plus acceptable pour cette application. Le nombre de coulombs consommés au cours du temps est nécessaire pour mesurer la durée de vie de la batterie, qui était de 28,1403 C en 1 h et 35 min (5700 s), cela a été mesuré sur le Joulescope. Ensuite, il faut convertir la capacité de nos batteries en coulombs. J'utilisais deux batteries Li-ion 18650 en parallèle ayant chacune une capacité de 1800 mAh.

Capacité de la batterie en C (coulombs) :

3.6 Ah × 3600 s/h = 12960 C

Comme la tension de coupure est d'environ 3,3 V, ce qui est plus élevé que les 3 V (qui est la tension lorsque la batterie est complètement vidée), nous devons supposer environ 80 à 90% de capacité utilisable. En gardant à l'esprit le taux d'autodécharge des batteries, utilisons 85% comme approximation :

0.85 × 12960 C = 11016 C

Calculons maintenant le taux de consommation en coulombs par seconde :

28.1403 C / 5700 s = 0.004936 C/s

Enfin, pour calculer l'autonomie de la batterie, il faut diviser la capacité utilisable de la batterie par le taux de consommation pour obtenir l'autonomie de la batterie en heures :

(11016 C / 0.004936 C/s) / 3600 s/h = 619.93 h.

Ce qui représente environ 25 jours d'autonomie sur une seule charge. Sans aucune optimisation de la consommation d'énergie, l'ensemble du système n'allait durer que six jours ! Une fois que j'ai été satisfait de la consommation d'énergie après avoir éliminé d'autres LED inutiles, j'ai préparé le système pour le déploiement, comme on peut le voir sur la **figure 14**.

Composants et boîtier

J'ai conçu un boîtier personnalisé imprimé en 3D [5] pour abriter tous les composants du projet de nœud de capteurs LoRa. Résistant aux intempéries, ce boîtier principal contient efficacement tous les éléments,

Figure 16. Vue latérale de la nœud de capteur LoRa d'installation, montrant le connecteur étanche de type USB Type et les connecteurs étanches JST.

comme l'illustre la **figure 15.** Plusieurs facteurs ont dû être pris en compte lors de sa conception. Le boîtier assure l'étanchéité de toutes les connexions électriques, utilisant pour cela des connecteurs JST JWPF étanches spéciaux pour les capteurs externes et l'alimentation du panneau solaire. De l'époxy a été utilisé pour sceller les espaces entre les connecteurs et la structure du boîtier

Un bouton étanche a été intégré pour permettre le redémarrage, ainsi que l'allumage ou l'extinction du système. Un connecteur USB-C étanche a été utilisé pour charger de nouveaux micrologiciels ou déboguer des problèmes sans avoir à ouvrir le boîtier, comme le montre la **figure 16**. Ce port permet également de recharger la batterie en cas d'urgence, si le mauvais temps persiste plusieurs jours et que les batteries s'épuisent faute de soleil.

Pour le capteur de température, d'humidité et de CO_2 SCD30, un boîtier Stevenson personnalisé (**figure 17)**, a été conçu pour protéger le capteur de la pluie, étant donné qu'il n'est pas conçu pour un usage extérieur. Ce boîtier Stevenson peut également abriter d'autres capteurs, grâce à un support de composants interne adapté à la disposition du circuit imprimé du module Grove, permettant l'installation de jusqu'à quatre petits capteurs Grove.

Après l'installation du système, comme le montre la figure 18, ses capacités d'étanchéité ont été rapidement mises à l'épreuve. Malgré une semaine de fortes pluies et une tempête, le système est resté sec à l'intérieur, et tous les composants ont continué à fonctionner parfaitement.

Figure 17. Vue en contre-plongée du nœud de capteur LoRa, présentant le boîtier Stevenson.

Intégration avec Datacake

L'intégration de votre nœud de capteur LoRa avec Datacake permet une visualisation et une gestion efficaces des données. Le processus commence dans la console The Things Network (TTN), où vous devez vous connecter et naviguer vers votre application. Sous l'onglet *Integrations*, sélectionnez *Webhooks* et cliquez sur *Add Webhook*. Choisissez *Datacake* dans la liste des modèles de webhook disponibles et configurez-le en entrant les détails nécessaires, y compris votre clé API Datacake, que vous pouvez trouver dans votre compte

Figure 18. Nœud de capteurs LoRa déployé.

Datacake sous la section *API*.
Ensuite, créez un compte sur Datacake si ce n'est pas déjà fait, et configurez un nouveau dispositif correspondant à votre nœud de capteur LoRa. Lors de la configuration du dispositif, fournissez des détails tels que le Device EUI et connectez-le à The Things Network. Cela garantit que les données envoyées depuis TTN sont reçues par Datacake. Dans la console TTN, naviguez vers votre application et allez à l'onglet *Payload Formats*. Pour Datacake, nous utiliserons le même format de charge utile pour décoder les données reçues de TTN. Collez simplement le même code de formatage de charge utile JavaScript dans la section de décodeur de charge utile dans l'onglet de configuration du dispositif sur Datacake et enregistrez les modifications une fois la configuration terminée.
Revenez ensuite à Datacake et naviguez vers le tableau de bord du dispositif que vous avez créé. Ajoutez des champs pour afficher les valeurs des capteurs, telles que l'humidité du sol, la température, l'humidité, les niveaux de CO_2 et l'état de la batterie. Définissez le type de données approprié pour chaque champ et mappez les champs de charge utile de TTN aux champs correspondants sur Datacake comme illustré à la **figure 19**. Cette cartographie assure que les données reçues de TTN sont correctement affichées sur votre tableau de bord Datacake.

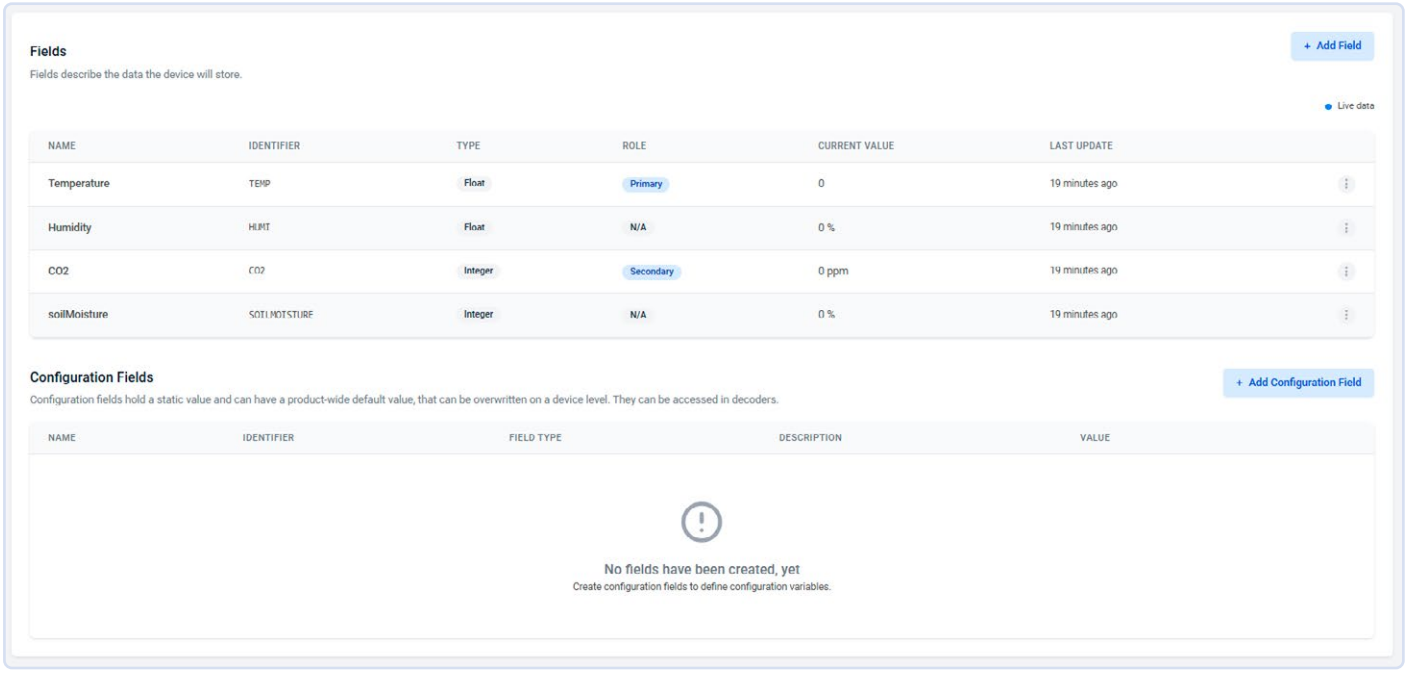

Figure 19. Configuration sur le terrain de Datacake des données de capteurs reçues du TTN.

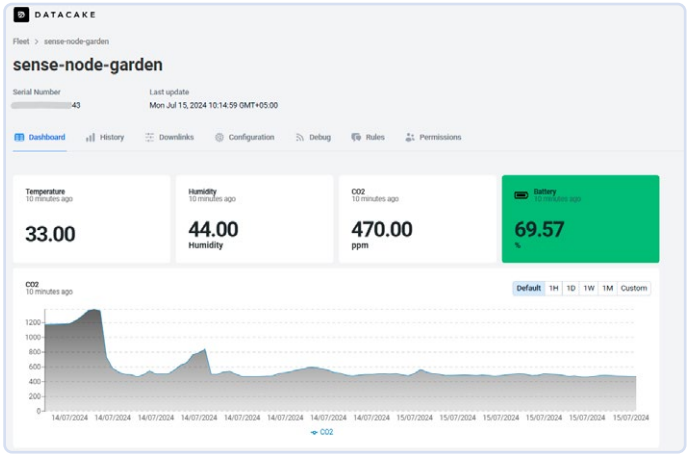

Figure 20. Tableau de bord Datacake montrant toutes les données de capteurs collectées.

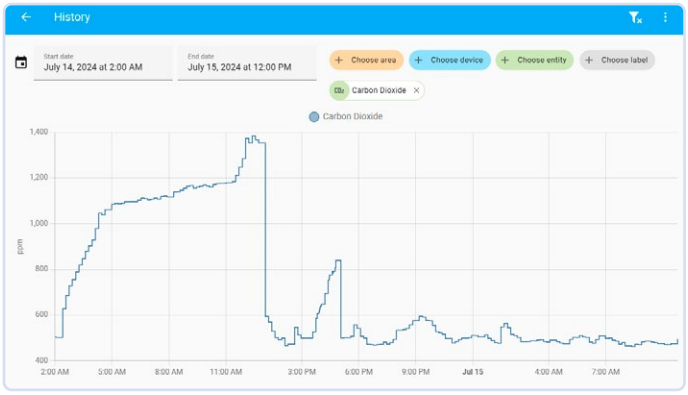

Figure 21. Graphique linéaire de l'historique dans Home Assistant.

Enfin, vérifiez que les données de votre nœud de capteur LoRa sont bien transmises à TTN et redirigées vers Datacake. Votre tableau de bord Datacake devrait maintenant afficher les valeurs des capteurs en temps réel, vous permettant de surveiller et d'analyser les données efficacement, comme le montre la figure 20. Cette intégration offre une plateforme puissante pour visualiser et gérer vos données de capteurs, facilitant le suivi des conditions environnementales et la prise de décisions éclairées basées sur les données collectées.

Intégration dans Home Assistant

Intégrer votre nœud de capteur LoRa dans Home Assistant [8] offre une excellente solution pour le stockage et l'analyse des données à long terme, sans les limitations de stockage imposées par des plateformes telles que Datacake. Puisque Home Assistant fonctionne sur votre propre installation domestique, il n'y a pas de limites en termes d'espace de stockage ou de points de données, ce qui le rend idéal pour une surveillance continue.

Pour intégrer Home Assistant avec The Things Network (TTN), commencez par générer une clé API dans la console TTN. Naviguez vers votre application, accédez à la section *API Keys*, et créez une nouvelle clé avec les permissions nécessaires. Assurez-vous également d'activer l'intégration de stockage dans les intégrations du tableau de bord de l'application TTN ; sinon, les données du capteur ne s'afficheront pas dans Home Assistant.

Ensuite, dans Home Assistant, allez dans *Devices & Services* et ajoutez l'intégration de The Things Network. Lorsqu'on vous le demande, entrez le nom de votre application dans TTN et la clé API que vous avez générée. Cela reliera votre instance Home Assistant à votre application TTN, permettant ainsi le flux de données de votre nœud de capteur LoRa vers Home Assistant.

Pour visualiser les valeurs des capteurs dans un graphique linéaire comme vu dans la **figure 21**, vous devez définir l'unité de mesure pour chaque entité de capteur reçue de l'intégration The Things Network. Ajoutez les entrées suivantes à votre fichier configuration.yaml dans Home Assistant :

```
sensor.lora_sense_node_temperature:
  unit_of_measurement: "°C"
sensor.lora_sense_node_co2:
  unit_of_measurement: "ppm"
sensor.lora_sense_node_humidity:
  unit_of_measurement: "%"
sensor.lora_sense_node_soilmoisture:
  unit_of_measurement: "%"
```

Dans ces entrées, `sensor.lora_sense_node_temperature` est le nom de l'entité où Home Assistant reçoit les valeurs de température du TTN. De même, définissez l'unité de mesure pour les capteurs de CO_2, d'humidité et d'humidité du sol. Cette configuration permet de s'assurer que les valeurs des capteurs sont affichées correctement sur le tableau de bord de Home Assistant.

En complétant ces étapes, vous pouvez avoir une carte détaillée et interactive sur le tableau de bord de votre Home Assistant, comme le montre la **Figure 22,** montrant toutes les valeurs des capteurs que votre nœud de capteur LoRa envoie. Cette configuration fournit une solution robuste et flexible pour la surveillance et l'analyse des données environnementales, en exploitant les capacités de stockage et l'interface personnalisable de Home Assistant.

Améliorations futures et potentiel pour diverses applications

Bien que le système actuel de nœud de capteur LoRa soit efficace, il présente un potentiel important d'amélioration, notamment en termes d'efficacité énergétique. Au repos, le système consomme environ 4,32 mA, ce qui est relativement élevé pour des applications alimentées par batterie sur le long terme. En intégrant une horloge en temps réel (RTC) externe avec un circuit de verrouillage de puissance, nous pourrions réduire considérablement la consommation au repos jusqu'à atteindre 50 nA.

Dans cette configuration améliorée, l'horloge RTC externe serait contrôlée par le microcontrôleur, et cette RTC gérerait le circuit de verrouillage de puissance. Ce circuit couperait l'alimentation de l'ensemble du système lorsqu'il n'envoie pas activement des valeurs de capteurs et la réactiverait à des intervalles prédéfinis. Cette approche prolongerait considérablement la durée de vie de la batterie, rendant le système plus adapté pour des déploiements à distance et de longue durée. Cette solution pourrait être mise en œuvre dans la prochaine version du tableau d'expansion ou sous forme d'un module séparé pouvant être ajouté à la configuration existante. Restez à l'écoute pour un article à ce sujet !

De plus, d'autres optimisations matérielles peuvent être explorées. Par exemple, remplacer les composants à haute consommation d'énergie par des alternatives plus efficaces, optimiser le firmware pour assurer une utilisation minimale de l'énergie, et améliorer l'efficacité du système de charge solaire sont toutes des améliorations potentielles.

La nature modulaire de ce système permet l'ajout de divers capteurs pour étendre sa fonctionnalité. En intégrant davantage de capteurs, cette configuration peut être utilisée pour une large gamme d'applications, y compris, mais sans s'y limiter :

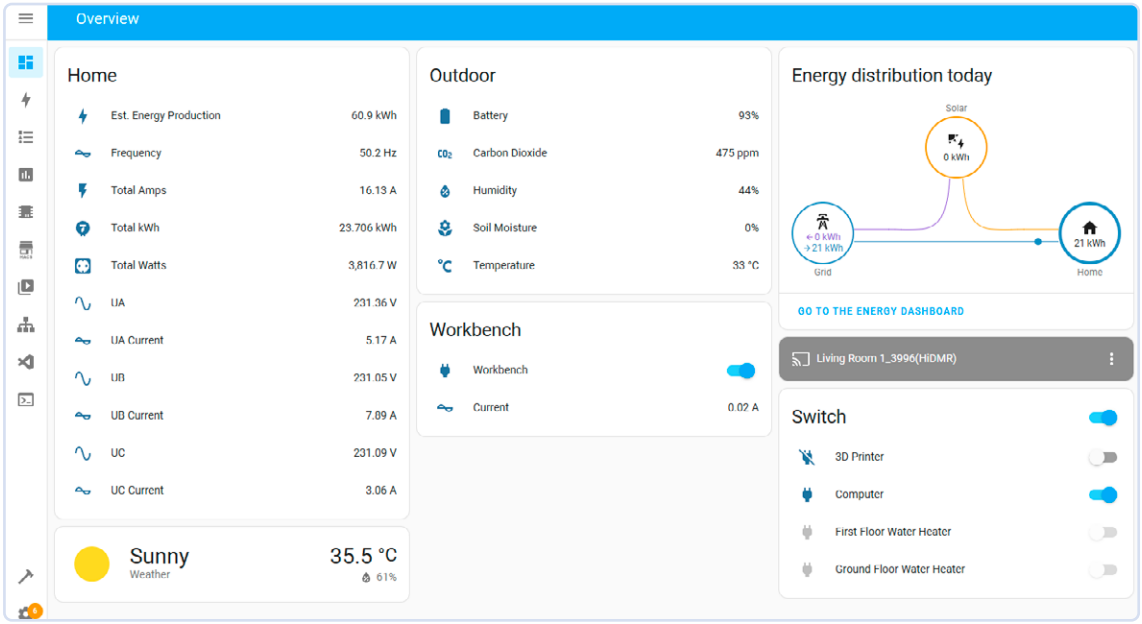

Figure 22. Données du nœud de détection LoRa affichées sur le tableau de bord de Home Assistant.

- **Surveillance environnementale :** L'ajout de capteurs de qualité de l'air, d'intensité lumineuse et de niveaux sonores peut faire de ce système une station de surveillance environnementale complète.
- **Applications agricoles :** L'intégration de capteurs de pH du sol, de capteurs de nutriments et de capteurs météorologiques peut fournir des données précieuses pour l'agriculture de précision.
- **Surveillance industrielle :** Inclure des capteurs de fuite de gaz, de vibration et de pression peut aider à surveiller les environnements industriels pour la sécurité et l'efficacité

En conclusion, le système actuel de nœud de capteur LoRa offre une plateforme robuste et flexible pour la surveillance environnementale à distance. Avec des améliorations futures axées sur l'efficacité énergétique et l'intégration de capteurs supplémentaires, ce système a le potentiel d'être adapté à diverses applications, en faisant un outil inestimable pour divers besoins de surveillance.

240354-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (saad.imtiaz@elektor.com), ou contactez elektor (redaction@elektor.fr).

À propos de l'auteur

Saad Imtiaz (ingénieur senior, Elektor), est un ingénieur mécatronicien avec une expertise approfondie en systèmes embarqués et en développement de produits. Son expérience lui a permis de collaborer avec une variété d'entreprises, allant de startups innovantes à des multinationales renommées, en menant des projets de prototypage et de développement à la pointe de la technologie. Fort d'une riche expérience dans l'industrie aéronautique et ayant dirigé une startup technologique, Saad apporte à Elektor un mélange unique d'expertise technique et un esprit entrepreneurial affirmé. Il contribue au développement de projets dans les domaines du logiciel et du matériel.

Produits

- **Seeed Studio XIAO ESP32C3**
  www.elektor.fr/20265
- **Seeed Studio Grove SCD30 CO2**
  www.elektor.fr/20012
- **Seeed Studio LoRa-E5 STM32WLE5JC**
  www.elektor.fr/19956
- **Seeed Studio Solar Panel (3 W)**
  www.elektor.fr/19131
- **Waveshare Solar Power Management Module**
  www.elektor.fr/20488
- **Dragino LoRa/LoRaWAN IoT Kit v3 (EU868)**
  www.elektor.fr/20775

LIENS

[1] Saad Imtiaz, "Elektor eXpansion Board v1.0," Elektor 7-8/2024: https://elektormagazine.com/240250-01
[2] Setting up The Things Network V3 on Dragino: http://wiki.dragino.com/xwiki/bin/view/Main/Notes%20for%20TTN/
[3] Dragino LPS8N - Setup with The Things Network: https://www.thethingsindustries.com/docs/gateways/models/dragino-lps8/
[4] The Things Network: https://www.thethingsnetwork.org/
[5] LoRa Sensor Node Github Repository: https://github.com/ElektorLabs/lora-sensor-node
[6] Joulescope JS220: Precision Energy Analyzer: https://www.joulescope.com/products/js220-joulescope-precision-energy-analyzer
[7] Datacake: https://datacake.co/
[8] Home Assistant: https://www.home-assistant.io/