Ecole des Mines de Saint-Etienne

Campus Georges Charpak Provence - Gardanne

Conception d'une Application IoT de Suivi en Temps Réel pour la Prévention des Maladies Respiratoires

BREATH SAFE

PROJET IOT

Préparé par : **Hajar ZOUGGARI**

Yassmina BARA

Supervisé par : M .Elias KHARBOUCHE

M. Marques ACACIO

M .Francois BERNIER

Janvier 2025

Table de matière

1 Introduction

La qualité de l'air joue un rôle crucial dans la santé publique, en particulier en ce qui concerne les maladies respiratoires. Avec l'urbanisation croissante, l'industrialisation et le changement climatique, l'exposition aux polluants atmosphériques est devenue une préoccupation majeure. Des études ont démontré que des facteurs environnementaux tels que la température, la pression atmosphérique, l'humidité et les particules fines ont un impact direct sur la santé respiratoire. De plus, l'Indice de Qualité de l'Air (IAQ) permet d'évaluer le niveau de pollution auquel les individus sont exposés et de déterminer les risques associés.

Dans ce contexte, il devient impératif de mettre en place des dispositifs permettant un suivi précis et en temps réel de ces paramètres afin de prévenir les complications respiratoires et d'améliorer la qualité de vie des populations à risque. L'évolution des technologies IoT (Internet of Things) et des capteurs intelligents ouvre aujourd'hui de nouvelles perspectives pour la surveillance de la qualité de l'air et l'anticipation des problèmes de santé.

1.1 Problématique :

Les maladies respiratoires constituent aujourd'hui un enjeu majeur de santé publique. L'exposition prolongée à un air de mauvaise qualité peut entraîner des complications pour les personnes vulnérables, notamment les enfants, les personnes âgées et les patients souffrant d'affections respiratoires chroniques comme l'asthme et la BPCO. Cependant, les systèmes de surveillance de la qualité de l'air sont souvent coûteux, peu accessibles ou ne fournissent pas une analyse en temps réel.

La question qui se pose est donc la suivante : **Comment développer un dispositif IoT efficace et accessible permettant de surveiller en temps réel la qualité de l'air et de prévenir les risques liés aux maladies respiratoires ?**

1.2 Notre solution

Le projet **BreathSafe** s'inscrit dans cette démarche en développant une solution innovante qui permet de mesurer en continu les paramètres environnementaux clés, d'analyser leur impact sur la santé et d'alerter les utilisateurs en cas de conditions défavorables. Grâce à une approche combinant capteurs de haute précision, transmission des données en temps réel. Cette solution vise à offrir un outil de prévention efficace et accessible.

BREATH
SAFE

Ses objectifs :

1. Mesurer les paramètres environnementaux clés (température, pression atmosphérique, humidité, IAQ) avec une grande précision à l'aide de capteurs intelligents.
2. Intégrer la connectivité IoT en exploitant The Things Network (TTN) et Datacake pour assurer la transmission et le stockage des données.
3. Analyser en continu les données recueillies pour détecter les conditions susceptibles d'aggraver les problèmes respiratoires.
4. Alerter les utilisateurs en cas de détection de risques via une interface intuitive, des notifications ou des recommandations adaptées.

2 Synoptique

Notre solution est modelisee dans le synoptique suivant :

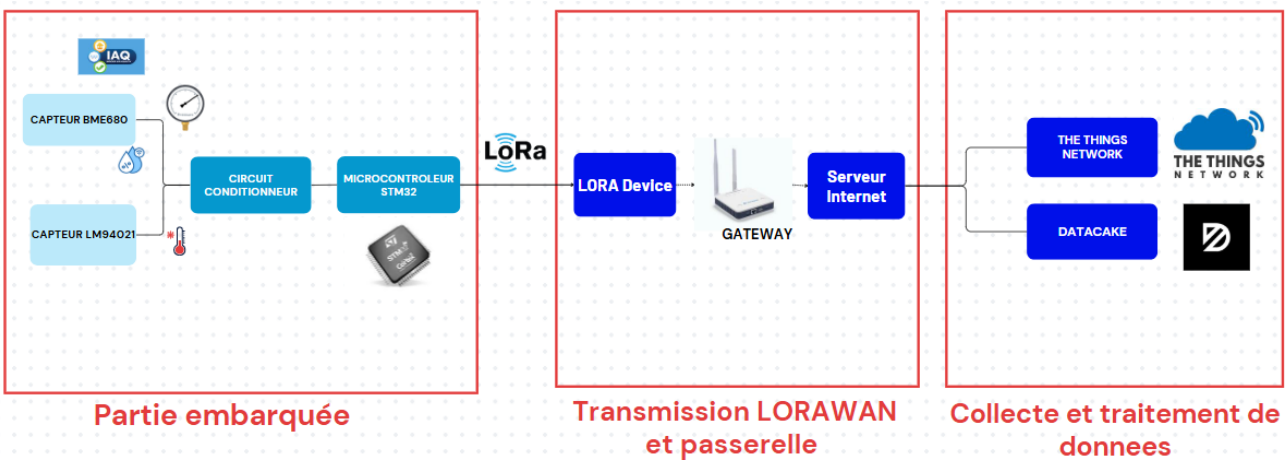


FIGURE 1 – Le synoptique du projet

— On a utilise le capteur LM94021 pour la mesure de la température, et le capteur de BME680 pour le mesure de la pression, l'humidité, et la qualité de l'air.
— Les signaux bruts des capteurs sont traités par **un circuit conditionneur** pour adapter les niveaux de tension et améliorer la qualité des signaux.
— Les données conditionnées sont envoyées au microcontrôleur STM32 pour extraction et prétraitement.
— Transmission des mesures via le protocole LoRaWAN et le Gateway qui convertit la trame LORA en trame IP, ce Gateway utilise le protocole MQTT.
— le serveur internet contient l'application THE THINGS NETWORK ou les données sont collectées et stockées, et DATACAKE pour analyser et visualiser les mesures sous forme de graphiques et histogrammes.

3 Développement :

L'objectif principal de cette phase est de créer un système fonctionnel qui surveille et analyse en temps réel les paramètres environnementaux, tels que la température, l'humidité, la pression atmosphérique et la qualité de l'air. Ces paramètres sont cruciaux pour la prévention des risques respiratoires. Le système devra être capable de détecter des conditions défavorables pour la santé, d'envoyer des alertes en cas de danger et d'afficher les données sous forme compréhensible pour l'utilisateur

3.1 Choix des Composants et Plateformes

3.1.1 Les composants

STM32 :Dans notre cas, nous avons choisi un microcontrôleur STM32 pour sa puissance, sa faible consommation d'énergie et ses nombreuses fonctionnalités, telles que les interfaces de communication et les timers, qui sont idéales pour le traitement des données des capteurs et la gestion du projet

Stlink debugger :Le débogueur ST-Link a été choisi pour sa compatibilité avec les microcontrôleurs STM32, sa facilité d'utilisation, et son intégration avec STM32CubeIDE. Il permet un débogage rapide et efficace à un coût abordable.

BME680 : Ce capteur mesure la température, l'humidité, la pression et la qualité de l'air (avec des indices tels que le gaz de qualité de l'air).

LM94021 : Ce capteur est utilisé pour mesurer la température. Il offre une grande précision et est adapté aux applications IoT en raison de sa faible consommation d'énergie.

Antenne LORA : L'antenne LoRa permet de transmettre et recevoir des données sur de longues distances avec une faible consommation d'énergie, en utilisant la technologie LoRa (Long Range). Elle est essentielle pour établir la communication entre les dispositifs IoT sur des réseaux à faible bande passante, comme LoRaWAN.

Explication des choix de capteurs : Nous avons choisi le LM94021 pour la mesure de la température en raison de sa haute précision et de sa stabilité thermique, ce qui est crucial pour un suivi fiable dans le cadre de notre projet de surveillance des maladies respiratoires. Bien qu'il n'offre pas de mesures multiples, sa spécialisation dans la température assure des données plus précises que celles fournies par le BME680.

3.1.2 Plateformes logiciels

STM32CubeIDE est un environnement de développement basé sur Eclipse, conçu pour les microcontrôleurs STM32. Il intègre un éditeur de code, un compilateur, un débogueur, et le générateur de code STM32CubeMX, permettant une configuration facile des périphériques et la génération automatique du code d'initialisation. Cette solution complète facilite le développement et le débogage des projets STM32.

TTN (The Things Network) : TTN est une plateforme IoT qui permet de transmettre les données collectées par les capteurs vers un serveur central pour leur traitement. Ce réseau utilise la technologie LoRaWAN, qui est idéale pour les systèmes IoT nécessitant de faibles débits de données et une faible consommation d'énergie.

Datacake : C'est une plateforme de gestion de données IoT qui permet de visualiser, analyser et interpréter les données transmises par TTN. Elle offre des outils pour configurer des alertes et des seuils, ce qui est essentiel pour ce projet.

Pourquoi LoRa pour le projet BreathSafe ?

Le choix de LoRa (Long Range) pour la communication IoT dans le cadre de notre projet BreathSafe repose sur plusieurs critères techniques et fonctionnels qui en font une technologie particulièrement adaptée aux besoins spécifiques de notre solution. Voici les principaux points justifiant ce choix :

Le choix de LoRa pour notre projet BreathSafe repose sur plusieurs critères clés :

- **Large couverture géographique** : LoRa offre une portée jusqu'à 15 km en milieu rural, idéale pour surveiller la qualité de l'air sur de grandes distances.
- **Faible consommation d'énergie** : LoRa permet une faible consommation, garantissant une autonomie prolongée des capteurs.
- **Coût réduit** : LoRa est économique, rendant notre solution accessible tout en assurant une large adoption.
- **Adapté aux environnements IoT** : Son faible débit et sa faible latence sont parfaits pour des échanges réguliers de données de qualité de l'air.
- **Technologie ouverte** : LoRaWAN est un standard ouvert qui assure l'interopérabilité avec d'autres dispositifs IoT.
- **Soutien à la mise en réseau décentralisée** : Avec LoRa et TTN, notre projet bénéficie d'une infrastructure décentralisée flexible pour un déploiement à grande échelle.

Pourquoi MQTT choisi par TTN ?

Le choix de **MQTT** par **The Things Network (TTN)** est motivé par plusieurs avantages :

- **Légèreté et efficacité** : MQTT est un protocole léger, idéal pour les réseaux IoT à faible consommation de bande passante comme LoRaWAN.
- **Communication asynchrone** : Permet une transmission de données sans attendre une réponse immédiate, ce qui est adapté aux capteurs IoT.
- **Gestion des messages en cas de perte** : Garantit la fiabilité des messages grâce à la qualité de service (QoS), même en cas de perte de connexion.
- **Modèle publish/subscribe** : Facilite la distribution des données vers plusieurs abonnés tout en réduisant la charge du réseau.

Ces caractéristiques font de MQTT un protocole optimal pour les applications IoT sur LoRaWAN.

3.2 Gestion des Capteurs et Transmission des Données

3.2.1 Bibliothèques choisies pour l'application

Dans notre application, plusieurs bibliothèques ont été choisies pour faciliter l'intégration des capteurs et la communication avec le réseau LoRaWAN, à savoir **LMIC**, **CayenneLPP**, et **BME680**.

- **LMIC (LoRaMAC-in-C)** :
  - Permet de gérer les connexions et les échanges de messages avec un réseau LoRaWAN (activation par réseau ABP ou activation par OTAA).
  - Essentiel pour connecter le dispositif au réseau LoRaWAN, comme **The Things Network (TTN)**, et envoyer les données collectées par les capteurs.
- **CayenneLPP** :
  - Encode les données de capteurs dans le format standardisé **Cayenne Low Power Payload (LPP)**.
  - Compatible avec plusieurs plateformes IoT, y compris **TTN** et **Datacake**.
  - Structure les données sous forme de "payloads" (charges utiles) compactes et efficaces, optimisées pour les transmissions sur des réseaux à faible consommation d'énergie comme LoRaWAN.
  - Facilite l'envoi des valeurs collectées (ex : température, humidité) dans un format facilement interprétable par la plateforme de gestion des données.
- **BME680** :
  - Gère le capteur **BME680**, un capteur environnemental mesurant la température, l'humidité, la pression atmosphérique et la qualité de l'air (IAQ).
  - Particulièrement utile pour la surveillance de l'environnement.
  - Permet de récupérer ces données et de les utiliser pour surveiller les conditions ambiantes (température, humidité et qualité de l'air), essentielles dans le cadre du projet de surveillance des maladies respiratoires.
- **BME68x_necessary_functions** :
  - Inclut les fonctions nécessaires pour interagir avec le capteur BME68x et récupérer les mesures de pression, humidité, température et qualité de l'air (IAQ).
  - Ces fonctions permettent de lire les registres du capteur, d'effectuer les conversions nécessaires et de renvoyer les données sous un format compréhensible pour l'application.
  - Utilisée pour obtenir des mesures environnementales essentielles dans notre projet.

Ces trois bibliothèques sont complémentaires et jouent un rôle essentiel dans la collecte des données de capteurs, leur encodage dans un format adapté, et leur transmission sur le réseau LoRaWAN vers des plateformes comme TTN et Datacake pour une gestion efficace et un traitement ultérieur.

3.2.2 Acquisition et Mesure des Données des Capteurs

— **LM94021**

```c
uint32_t get_ADC_value(ADC_HandleTypeDef hadc1, uint32_t channel)
{
    uint32_t adc_val = 0;
    HAL_ADCEx_Calibration_Start(&hadc1, ADC_SINGLE_ENDED);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, 100);
    adc_val = HAL_ADC_GetValue(&hadc1);
    return adc_val;
}
float GET_temperature(uint32_t ADC_value, double VDD){
    float TEMP_value = 0.0;
    float voltage = 0.0;
    voltage = (ADC_value*VDD)/4095; // Int => Volts
    TEMP_value = (1034-voltage)/5.48;
    return TEMP_value;
}
```

```c
float readtemperaturesensor(){
    //Il faut alimenter le capteur

    HAL_GPIO_WritePin(Alim_temp_GPIO_Port, Alim_temp_Pin, GPIO_PIN_SET);
    HAL_Delay(100);
    uint value = 0xDF; /// read from evrything ...make your own sensor
    HAL_ADC_Start(&hadc1);
    value = get_ADC_value(hadc1, ADC_CHANNEL_15);
    debug_str("Brut value:\n");
    debug_uint(value);
    debug_str("\r\n");
    float temperature_sensor_value = GET_temperature(value, 3300);
    debug_valfloat("Temperature sensor value = ", temperature_sensor_value, 6);//6 doit etre le nbr de caracteres
    //On éteint le capteur
    HAL_GPIO_WritePin(Alim_temp_GPIO_Port, Alim_temp_Pin, GPIO_PIN_RESET);
    return temperature_sensor_value;
}
```

Fonction	Description
get_ADC_value	Effectue la calibration de l'ADC, démarre la conversion, attend la fin de la conversion, puis récupère et renvoie la donnée numérique échantillonnée.
GET_temperature	Linéarise la valeur numérique provenant de l'ADC en une tension avec une résolution de 12 bits, puis calcule la température en °C en fonction de la référence de tension (VDD).
readtemperaturesensor	Alimente le capteur, récupère la valeur ADC, convertit la donnée en température, affiche les valeurs, puis désactive le capteur.

TABLE 1 – Description des fonctions

— **BME680**

Pour l'utilisation du capteur BME68x dans notre projet, nous avons inclus les bibliothèques nécessaires, telles que 'bme68x_necessary_functions' et 'bme68x.c', afin de gérer la communication avec le capteur et d'extraire les données environnementales cruciales.

Afin d'extraire chaque type de donnée, nous avons défini trois fonctions spécifiques dans le code STM32, permettant de collecter et de traiter les informations fournies par le capteur :

- **Mesure de la pression :**
  - Nous avons utilisé une fonction dédiée pour récupérer les valeurs de pression atmosphérique, ce qui permet de suivre les variations de pression dans l'environnement et d'analyser leur impact sur la santé.
- **Mesure de l'humidité :**
  - Une fonction distincte a été définie pour extraire les données d'humidité. Ces mesures sont cruciales pour surveiller les conditions de l'air, notamment pour les personnes souffrant de maladies respiratoires.
- **Mesure de la qualité de l'air (IAQ) :**
  - Une fonction supplémentaire est utilisée pour obtenir des informations sur la qualité de l'air (IAQ), ce qui permet d'évaluer la pollution de l'air et son effet sur la santé des utilisateurs.

Chaque fonction est conçue pour interagir avec le capteur BME68x, en utilisant les bibliothèques spécifiques pour lire les registres du capteur, effectuer les conversions nécessaires, et récupérer les mesures brutes. Ces données sont ensuite traitées et utilisées dans notre application pour fournir des informations sur l'environnement et aider à la gestion des maladies respiratoires.

```c
double GET_Humidity()
{
    double humidite=0;
    struct bme68x_data data;
    bme68x_start(&data, &hi2c1);

    if (bme68x_single_measure(&data) == 0) {
        humidite=data.humidity;;
    }
    return humidite;
}
```

FIGURE 2 – Extraction de l'Humidité

```
159 double GET_Pressure()
160 {
161     double pression=0;
162     struct bme68x_data data;
163     bme68x_start(&data, &hi2c1);
164 
165     if (bme68x_single_measure(&data) == 0) {
166         pression=data.pressure;;
167     }
168     return pression;
169 }
```

FIGURE 3 – Extraction de la pression

```
171 double GET_IAQ()
172 {
173     double Iaq=0;
174     struct bme68x_data data;
175     bme68x_start(&data, &hi2c1);
176 
177     if (bme68x_single_measure(&data) == 0) {
178         data.iaq_score = bme68x_iaq();
179         Iaq=data.iaq_score;
180     }
181     return Iaq;
182 }
```

FIGURE 4 – Extraction de l'IAQ

3.3 Transfert des données vers TTN

Configuration de la Plateforme TTN (The Things Network

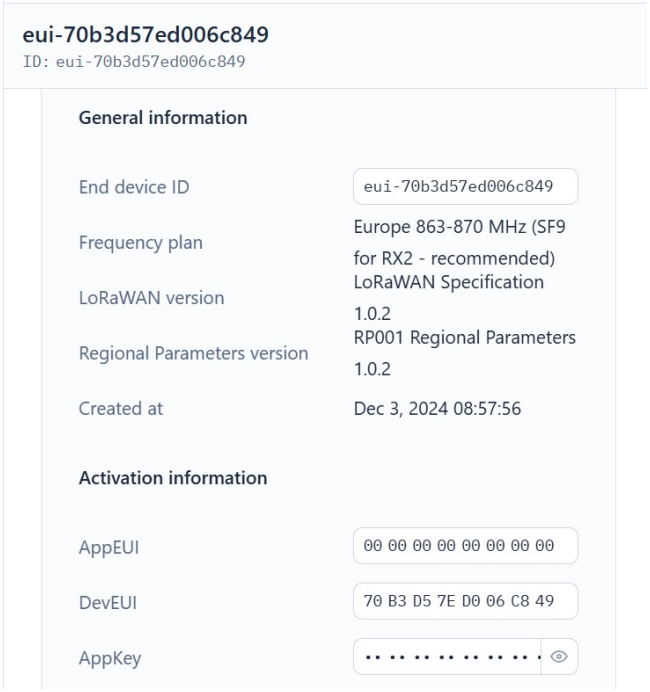


FIGURE 5 – Configuration de notre application sur TTN

Nous avons configuré TTN en créant l'application, en enregistrant le dispositif et en générant les clés nécessaires (DevEUI, AppEUI, AppKey pour OTAA (Over-the-Air Activation) . Ensuite, ces in-

formations ont été intégrées dans le code avec la bibliothèque LMIC pour établir la communication LoRaWAN.

```
// application router ID (LSBF) < ------- IMPORTANT
static const u1_t APPEUI[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
// unique device ID (LSBF) < ------- IMPORTANT
static const u1_t DEVEUI[8] = {0x49, 0xC8, 0x06, 0xD0, 0x7E, 0xD5, 0xB3, 0x70};
// device-specific AES key (derived from device EUI (MSBF))
static const u1_t DEVKEY[16] = {0x0F, 0x6D, 0x92, 0x49, 0x25, 0xA8, 0x89, 0xD9, 0x20, 0x8C, 0x28, 0x6C, 0x3E, 0xBE, 0xCF, 0x32};
```

FIGURE 6 – Insertion des Identifiants dans le code

Les constantes APPEUI, DEVEUI et DEVKEY sont ajoutées pour configurer l'activation OTAA sur le réseau TTN. APPEUI identifie l'application sur TTN, DEVEUI est l'identifiant unique du dispositif, et DEVKEY est la clé secrète utilisée pour sécuriser l'activation. L'ordre des octets MSB (Most Significant Bit) et LSB (Least Significant Bit) est essentiel pour une intégration correcte. Cette configuration permet au dispositif de rejoindre et de communiquer avec TTN de manière sécurisée. Implémentation dans le code

La figure suivante représente l'implémentation de la fonction reportfunc :

```
static osjob_t reportjob;
// report sensor value every minute
static void reportfunc (osjob_t* j) {
    double humidite=0;
    double pression=0;
    double temperature=0;
    double Iaq=0;
    //Extraction des mesures
    humidite=GET_Humidity();
    temperature=readtemperaturesensor();
    pression=GET_Pressure();
    Iaq=GET_IAQ();
    cayenne_lpp_reset(&lpp);
    //Formatter les donnees
    cayenne_lpp_add_temperature(&lpp, 0, temperature);
    cayenne_lpp_add_barometric_pressure(&lpp, 2, pression / 100); // Conversion de Pa à hPa
    cayenne_lpp_add_relative_humidity(&lpp, 1, humidite);
    cayenne_lpp_add_analog_output(&lpp, 3, Iaq);
    //transmission des donnees
    LMIC_setTxData2(1, &lpp, 15, 0);
    os_setTimedCallback(j, os_getTime()+sec2osticks(10), reportfunc);
}
```

FIGURE 7 – Implémentation de reportfunc

Dans la fonction reportfunc, on appelle les différentes fonctions qui permettent d'extraire les différentes mesures provenues des capteurs, puis on formate ces données pour qu'elles soient au bon formats avant d'être transmises grâce a une fonction de la librairies LMIC.
Pour expliquer la valeur du 3 eme parametre de la fonction LMIC_setTxDta, Le Cayenne LPP est conforme à la restriction de taille de la charge utile, qui peut être abaissée à 11 octets, et permet à l'appareil d'envoyer différentes données de capteurs dans la même trame. Pour ce faire, les données de chaque capteur doivent être préfixées de un ou deux octets :

- **Canal de données :** Identification unique de chaque capteur, par exemple "capteur intérieur".
- **Type de données :** Identifie le type de données dans la trame, par exemple "température".

On a obtenu 15 en se basant sur le tableau suivant :

1 Byte	1 Byte	N Bytes	1 Byte	1 Byte	M Bytes	...
Data1 Ch.	Data1 Type	Data1	Data2 Ch.	Data2 Type	Data2	...

TYPE	LPP	Size	Data Resolution per bit
Digital Input	0x00	1	1
Digital Output	0x01	1	1
Analog Input	0x02	2	0.01 Signed
Analog Output	0x03	2	0.01 Signed
Illuminance Sensor	0x65	2	1 Lux Unsigned MSB
Presence Sensor	0x66	1	1
Temperature Sensor	0x67	2	0.1 °C Signed MSB
Humidity Sensor	0x68	1	0.5 % Unsigned
Accelerometer	0x71	6	0.001 G Signed MSB per axis
Barometer	0x73	2	0.1 hPa Unsigned MSB
Gyrometer	0x86	6	0.01 °/s Signed MSB per axis
GPS Location	0x88	9	Latitude : 0.0001 ° Signed MSB
			Longitude : 0.0001 ° Signed MSB
			Altitude : 0.01 meter Signed MSB

```
static void reportfunc (osjob_t* j) {
    // read sensor
    float val = readsensor_temp();
    debug_valdec("val temp = ", val);
    LMIC.frame[0] = 0;
    LMIC.frame[1] = 0x67;//temp
    val = val/100; // en 0.1 °C
    LMIC.frame[2] = val >> 8;
    LMIC.frame[3] = val;
    . . .
    . . .
```

FIGURE 8 – Extrait de datasheet

Une fois la procédure de jonction réussie, la fonction reportFunction est appelée pour transmettre les données collectées.

```
236 void onEvent (ev_t ev) {
237     debug_event(ev);
238     switch(ev) {
239     // network joined, session established
240         case EV_JOINING:
241             debug_str("try joining\r\n");
242             break;
243         case EV_JOINED:
244             // kick-off periodic sensor job
245             debug_str("Joined\r\n");
246             os_clearCallback(&blinkjob);
247             debug_led (1);
248             reportfunc(&reportjob);
249             break;
```

FIGURE 9 – Gestion des événements de jonction et d'envoi des données LoRaWAN

De plus, nous avons amélioré le Spreading Factor (SF) en passant de 7 à 11 dans le lmic.c. Cette modification permet d'augmenter la portée du signal et d'améliorer la réception dans des environnements avec des obstacles ou une faible couverture réseau, au prix d'un débit de transmission plus faible.

```
701     LMIC.adrTxPow = 14;
702     setDrJoin(DRCHG_SET, DR_SF11);//Was DR_SF7 TODO
703     initDefaultChannels(1);
704     ASSERT((LMIC.opmode & OP_NEXTCHNL)==0);
705     LMIC.txend = LMIC.bands[BAND_MILLI].avail + rndDelay(8);
706 }
```

FIGURE 10 – Modification du SF dans lmic.c

3.3.1 Configuration de la liaison entre TTN et DATACAKE

On a cree un webhook sur TTN, ceci afin de permettre l'intégration et le transfert automatique des données depuis **The Things Network (TTN)** vers **Datacake**. Ensuite, lors de la configuration de l'appareil sur Datacake, on a rempli les champs suivants par les differents ID provenus de TTN.

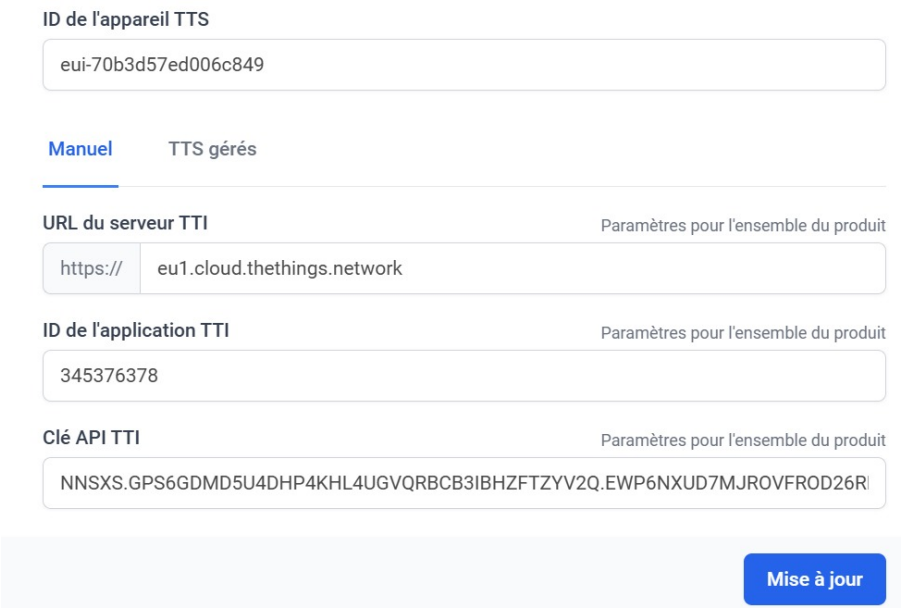
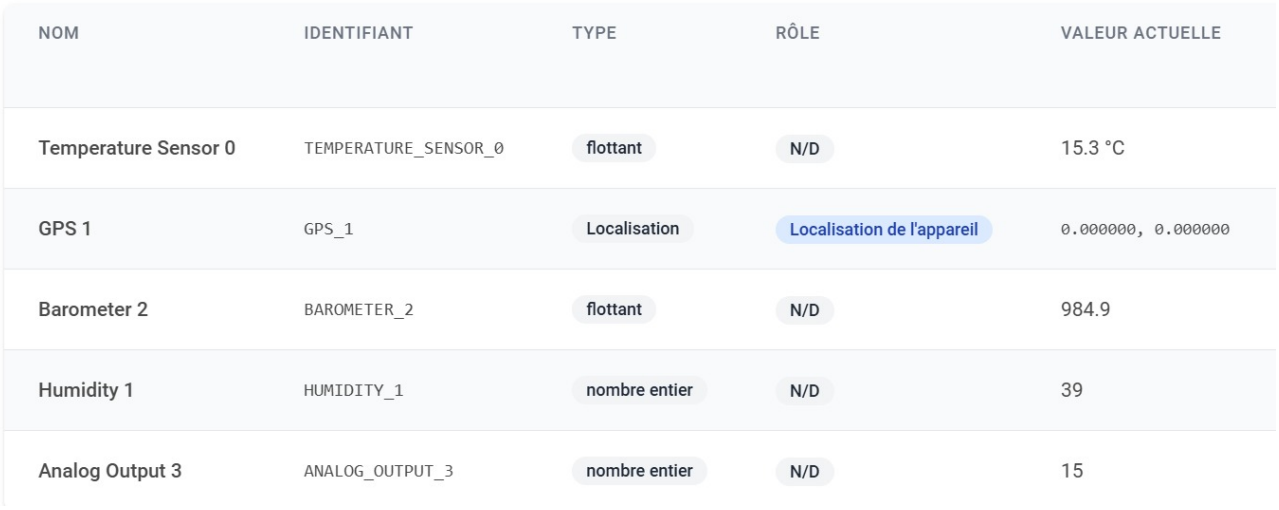


FIGURE 11 – Configuration de DATACAKE

Après avoir configure la liaison entre TTN et DATACAKE, on ajoute les différents champs correspondant aux données provenues de TTN. Ceci est représente dans la figure suivante :



NOM	IDENTIFIANT	TYPE	RÔLE	VALEUR ACTUELLE
Temperature Sensor 0	TEMPERATURE_SENSOR_0	flottant	N/D	15.3 °C
GPS 1	GPS_1	Localisation	Localisation de l'appareil	0.000000, 0.000000
Barometer 2	BAROMETER_2	flottant	N/D	984.9
Humidity 1	HUMIDITY_1	nombre entier	N/D	39
Analog Output 3	ANALOG_OUTPUT_3	nombre entier	N/D	15

FIGURE 12 – Champs à visualiser

4 Tests et Résultats

Dans le cadre de la phase de test du projet, nous avons tout d'abord exécuté le code et déployé l'application sur la carte STM32. Une fois l'activation du réseau (join) réussie, les données collectées par les capteurs sont envoyées via LoRaWAN et affichées sur la plateforme TTN (The Things Network), où nous pouvons suivre l'état de la connexion et la réception des messages. Ensuite, ces données sont transférées en temps réel vers Datacake , où elles sont traitées et visualisées sous forme de graphiques et d'indicateurs, permettant ainsi un suivi précis et continu des conditions environnementales mesurées par les capteurs.

Voici les résultats ci dessous :

FIGURE 13 – Résultats obtenus sur TTN

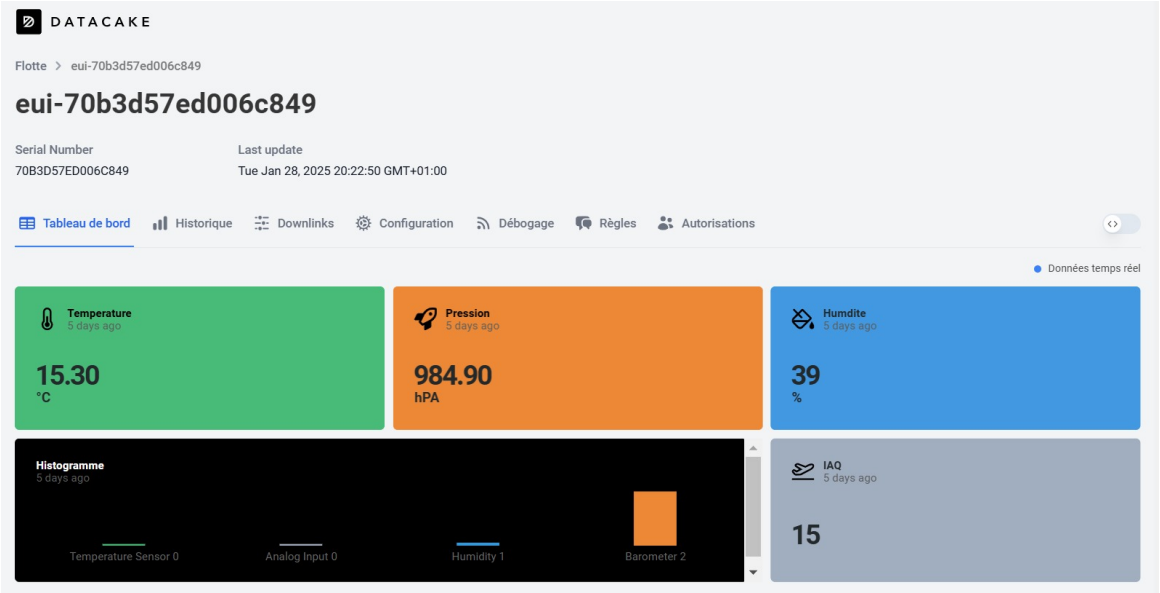


FIGURE 14 – Resultats obtenus sur le Dashboard de DATACAKE

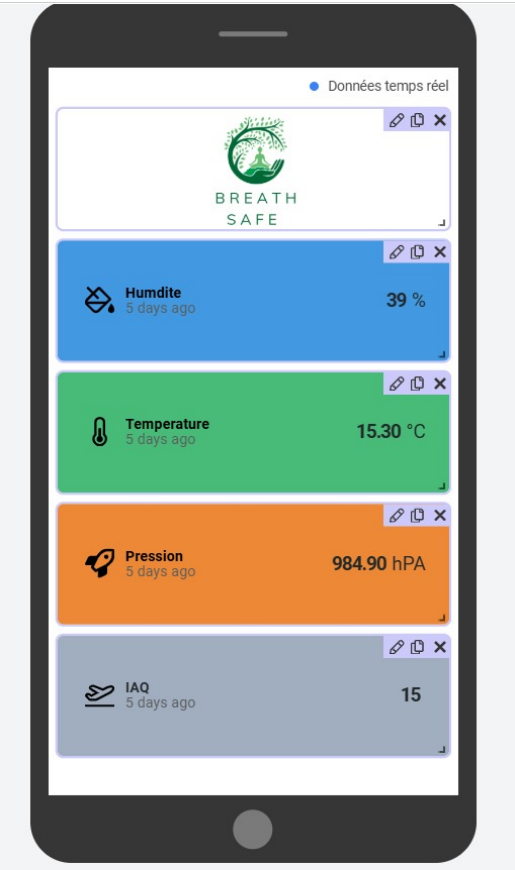


FIGURE 15 – Résultats obtenus sur l'application DATACAKE

- Humidité (39 %) : Dans une plage acceptable (30-60%), bonne gestion de l'humidité.
- Température (15.30℃) : Un peu basse mais acceptable pour un environnement frais, aucun souci majeur.
- Pression (984.90 hPa) : Légèrement en dessous de la moyenne mais dans une plage normale pour des conditions météorologiques variables.
- IAQ (15) : Excellente qualité de l'air, dans la plage idéale (0-50), parfait pour un environnement sain.

5 Conclusion

En résumé, les paramètres mesurés démontrent que l'environnement est globalement sain, validant ainsi l'efficacité du système BreathSafe. Cette surveillance continue et proactive permet non seulement de garantir une qualité d'air optimale, mais aussi d'anticiper et prévenir les risques de maladies respiratoires. Grâce à une détection rapide des variations et à une analyse précise des données, BreathSafe confirme sa capacité à assurer une protection sanitaire robuste, renforçant la confiance des utilisateurs et contribuant activement à la préservation de la santé publique.

6 Synthèse et Améliorations

Nous avons développé un système IoT permettant de collecter et de visualiser des données environnementales à l'aide de capteurs BME680 (température, humidité, pression, qualité de l'air) et LM94021 (température). Les signaux des capteurs sont traités par un circuit conditionneur avant d'être envoyés à un microcontrôleur STM32, qui les transmet via le protocole LoRaWAN

Les données sont ensuite acheminées vers un serveur réseau via une passerelle LoRaWAN, qui convertit les trames LoRa en trames IP. Elles sont stockées sur The Things Network (TTN), puis intégrées à la plateforme Datacake à l'aide d'un webhook, permettant une visualisation sous forme de graphiques et d'histogrammes

Une amélioration possible du projet serait l'intégration d'un système d'alerte Downlink permettant de prévenir l'utilisateur en cas de dépassement de seuil critique (ex. température élevée, mauvaise qualité de l'air), ceci en ajoutant les points suivants :

- Définition de seuils critiques sur Datacake.
- Envoi d'un message **Downlink** via TTN vers le microcontrôleur STM32 lorsqu'une alerte est déclenchée.
- Activation d'un **buzzer, LED, ou notification mobile** pour alerter l'utilisateur.

7 Bibliographie

https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme680-ds001.pdf

http://ecampus.emse.fr/course/view.php?id=1750

https://www.ti.com/lit/ds/symlink/lm94021-q1.pdf?ts=1738527273154&ref_url=https%253A%252F%252Fwww.google.com%252F