Capteur IoT pour la surveillance environnementale en bloc opératoire

Rapport de conception

Alexandra Baron - Ewen Breton

Janvier 2025

Encadrant : E. Kharbouche

ISMIN EI23

Sommaire

Table des figures

Liste des tableaux

1-Introduction

1.1 Contexte

Les blocs opératoires nécessitent une surveillance continue des paramètres environnementaux pour garantir des conditions stériles et sécurisées. Le maintien de ces conditions est très important pour limiter la prolifération des bactéries, réduire les risques d'infections et assurer un environnement optimal pour les interventions chirurgicales.

D'après [1] et [2], la température doit être maintenue entre 20 et 21 °C, tandis que le taux d'humidité doit être compris entre 50 et 70 %. Ces valeurs permettent de limiter la prolifération bactérienne tout en assurant un certain confort du personnel médical. La pression différentielle entre les différentes zones du bloc opératoire doit également être surveillée pour éviter toute contamination croisée, en maintenant une pression positive par rapport aux zones adjacentes. Enfin, la qualité de l'air est un élément fondamental, nécessitant des systèmes de filtration efficaces pour éliminer les particules et les agents pathogènes en suspension.

1.2 Problématique

Il est nécessaire de développer un système permettant de surveiller en temps réel ces paramètres critiques afin d'améliorer la gestion des environnements chirurgicaux et de garantir la sécurité des patients et du personnel médical.

1.3 Objectifs du projet

L'objectif de ce projet est de concevoir un capteur IoT permettant de surveiller en temps réel les paramètres environnementaux critiques d'un bloc opératoire, notamment la température, l'humidité, la pression et la qualité de l'air. Pour cela, le système devra :

- Mesurer avec précision ces paramètres à l'aide de capteurs adaptés.
- Transmettre les données via le protocole LoRaWAN afin d'assurer une communication longue portée et basse consommation.
- Visualiser les données en temps réel sur une plateforme cloud pour permettre un suivi efficace et une alerte rapide en cas d'anomalie.
- Garantir une solution autonome, adaptée aux exigences des environnements médicaux.

1.4 Méthodologie

Le projet repose sur la conception d'un système embarqué intégrant plusieurs composants électroniques pour assurer la surveillance des paramètres environnementaux en bloc opératoire.

Le dispositif est basé sur un microcontrôleur STM32L432, choisi pour sa faible consommation énergétique et sa capacité à gérer les capteurs et la communication sans fil. Deux capteurs sont utilisés :

- **LM94021** : Capteur analogique de température. Sa lecture est effectuée via un convertisseur analogique-numérique (ADC).
- **BME680** : Capteur numérique mesurant l'humidité, la pression et la qualité de l'air. Il communique avec le microcontrôleur via le protocole I2C.

Pour la transmission des données, un transceiver **LoRa RFM95W** est intégré, permettant l'envoi des mesures sur un réseau LoRaWAN. La communication entre les différents composants repose sur le protocole **SPI**, et l'encodage des données suit le format **Cayenne LPP** afin d'optimiser leur transmission.

L'ensemble du système est programmé et débogué à l'aide d'un **ST-Link**, permettant le développement et la mise au point du firmware sur le microcontrôleur.

Les données sont ensuite transmises vers **The Things Network (TTN)**, qui assure la réception des messages LoRaWAN, puis sont visualisées en temps réel sur la plateforme **Datacake**, offrant ainsi une interface de suivi et d'analyse.

2-Vue d'ensemble du système

2.1 Architecture globale du capteur

L'architecture du capteur IoT repose sur une conception modulaire adaptée aux contraintes des blocs opératoires. Chaque bloc opératoire est équipé d'un capteur dédié qui mesure des paramètres environnementaux (température, humidité, pression et qualité de l'air). Ces capteurs individuels sont reliés à une passerelle centrale via le protocole LoRaWAN, permettant une communication longue portée et à faible consommation d'énergie.

Un synoptique clair du système est présenté ci-dessous pour en visualiser les interactions entre les différentes parties.

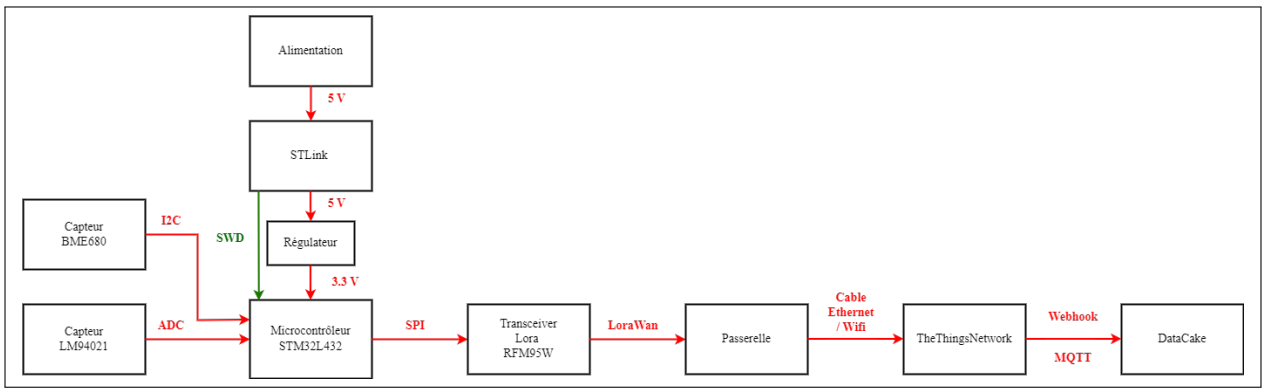

FIGURE 1 – Synoptique du système.

Le synoptique présenté illustre le fonctionnement du système pour un seul bloc opératoire. Cependant, le système est conçu pour être étendu : chaque bloc opératoire dispose de son propre capteur, et tous ces capteurs transmettent leurs données à une même passerelle centrale.

La passerelle collecte l'ensemble des données transmises par les capteurs individuels et les envoie vers la plateforme The Things Network. Les données issues des différents blocs opératoires sont ensuite visualisables sur une interface via Datacake. Ainsi, cela offre un suivi global ou par bloc opératoire.

2.2 Fonctionnement général

Le fonctionnement du capteur peut être résumé en plusieurs étapes :

— Acquisition des données : chaque capteur mesure les paramètres environnementaux spécifiques de son bloc opératoire.

— Traitement et formatage des données : Les données collectées par les capteurs sont traitées par le microcontrôleur STM32L432, puis encodées au format Cayenne LPP.

— Transmission des données : Le transceiver LoRa RFM95W transmet les données au réseau LoRaWAN, jusqu'à la passerelle centrale.

— Visualisation des données : Les données transmises vers TTN sont ensuite rendues accessibles sur Datacake, permettant une visualisation en temps réel des paramètres pour chaque bloc opératoire.

Un synoptique simplifié du système est présenté ci-dessous pour en comprendre le fonctionnement.

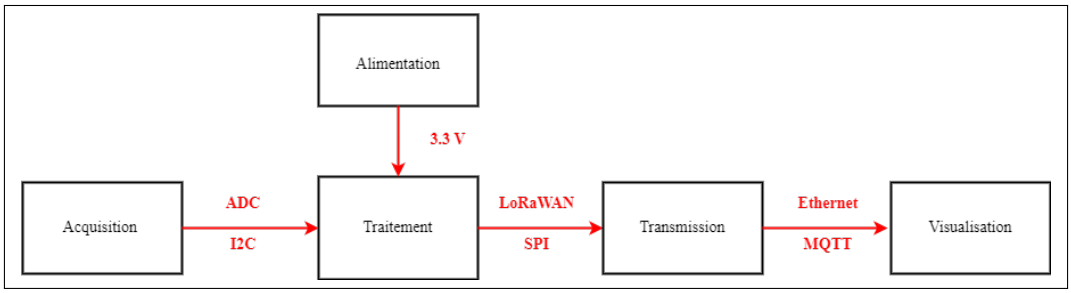

FIGURE 2 – Synoptique simplifié du système.

3-Description détaillée des blocs fonctionnels

3.1 Acquisition des données (Capteurs)

L'acquisition des données environnementales dans un bloc opératoire repose sur l'utilisation de capteurs capables de mesurer avec précision la température, l'humidité, la pression et la qualité de l'air. Ces capteurs sont essentiels pour garantir un environnement stérile, conformément aux normes médicales strictes énoncée précedement. Nous décrivons ici en détail les capteurs utilisés et leurs principes de fonctionnement. Les capteurs sélectionnés, le LM94021 pour la température et le BME680 pour l'humidité, la pression et la qualité de l'air, sont intégrés au système embarqué et communiquent avec le microcontrôleur STM32L432 pour assurer une surveillance en temps réel.

Le tableau ci-dessous résume les caractéristiques techniques des deux capteurs utilisés dans ce projet :

Caractéristique	**LM94021**	**BME680**
Type de capteur	Capteur de température analogique	Capteur numérique 4-en-1 (gaz, pression, température, humidité)
Plage de température	-50°C à +150°C	-40°C à +85°C
Plage de tension	1.5V à 5.5V	1.71V à 3.6V (VDD), 1.2V à 3.6V (VDDIO)
Consommation de courant	9 µA (typique)	2.1 µA à 12 mA (selon le mode)
Précision de la température	±1.5°C (20°C à 40°C)	±0.5°C (à 25°C)
Précision de l'humidité	N/A	±3% r.H.
Précision de la pression	N/A	±0.6 hPa
Précision de la qualité de l'air	N/A	IAQ (0-500)

TABLE 1 – Comparaison des caractéristiques des capteurs LM94021 et BME680

3.1.1 Capteur de température (LM94021)

Objectif

Mesurer la température ambiante du bloc opératoire à l'aide d'une sonde analogique connectée à l'ADC du microcontrôleur.

Méthode

Le capteur LM94021 fournit une sortie analogique dont la tension varie en fonction de la température. Pour obtenir la température en degrés Celsius, la valeur analogique est lue à travers le convertisseur analogique-numérique (ADC) du STM32. La résolution de l'ADC est de 12 bits, ce qui signifie que la valeur retournée est comprise entre 0 et 4095. La tension mesurée est proportionnelle à la température selon la formule :

$$\text{Tension} = \frac{\text{Valeur ADC} \times V_{DD}}{\text{ADC\_RESOLUTION}}$$

Où :

- VDD est la tension d'alimentation du système, ici 3300 mV.
- ADC_RESOLUTION est la résolution de l'ADC (4095 pour un ADC sur 12 bits).

La température est ensuite déduite à partir de la tension mesurée avec la formule suivante, qui est directement issue de la documentation technique du capteur [3] :

$$\text{Température} = \frac{1034 - \text{Tension}}{5.48}$$

Cette formule permet de convertir la tension mesurée en température en °C.

Résultat

Les mesures de température obtenues sont précises et conformes aux valeurs attendues. La combinaison de la résolution de l'ADC et de la formule de conversion permet d'obtenir une précision suffisante pour la plupart des applications environnementales.

3.1.2 Capteur de pression, humidité et qualité de l'air (BME680)

Objectif

Mesurer l'humidité, la pression atmosphérique et la qualité de l'air (IAQ) en utilisant le capteur BME680, qui communique avec le microcontrôleur STM32 via I2C. Une bibliothèque spécifique est utilisée pour interagir avec le capteur et récupérer les données mesurées.

Méthode

Le capteur BME680 fournit plusieurs paramètres environnementaux, à savoir :

- Humidité : Un capteur d'humidité basé sur un polymère absorbe ou relâche l'humidité de l'air, ce qui modifie sa capacité électrique et permet d'en déduire le taux d'humidité.
- Pression atmosphérique : Il utilise un capteur barométrique basé sur une membrane qui se déforme sous la pression, permettant de calculer la pression ambiante.
- Qualité de l'air (IAQ) : Grâce à un capteur de gaz basé sur un oxyde métallique chauffé, il détecte la présence de composés organiques volatils (COV), qui sont des indicateurs de pollution de l'air intérieur.

Le BME680 fonctionne principalement en mode forcé, ce qui signifie qu'il effectue une seule mesure à la demande avant de repasser en veille.

Résultat

Ce capteur permet ainsi de fournir des données environnementales précises, adaptées à une surveillance continue des conditions de l'air.

3.2 Traitement des données (Microcontrôleur STM32L432)

Objectif

Le microcontrôleur STM32L432 joue un rôle central dans le système. Il assure la collecte, le traitement et la préparation des données avant leur transmission via le réseau LoRaWAN. Ses objectifs principaux sont de centraliser les données provenant des capteurs, de les formater selon le protocole Cayenne LPP, et de les envoyer de manière optimisée pour une transmission sans fil.

Méthode

Le traitement des données est réalisé à travers une programmation en C sur STM32CubeIDE. On y réalise la onfiguration et la gestion des interfaces de communication telles que I2C, SPI et ADC. Le protocole I2C est utilisé pour communiquer avec le capteur BME680. La figure ci-dessous illustre les connexions du microcontrôleur STM32L432 avec les différents périphériques et capteurs :

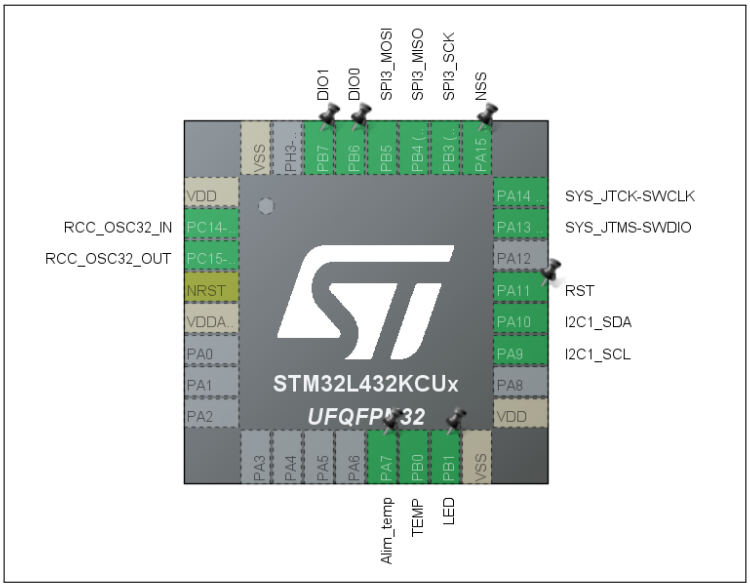

FIGURE 3 – Configuration du microcontrôleur.

Pour le flashage du firmware et le débogage du code embarqué, le programmateur ST-Link V2 a ete utilisé pour le telechargement du programme dans la memoire flash du STM32L432, le débogage en temps réel via l'interface SWD (Serial Wire Debug), offrant la possibilite de placer des points d'arrêt par exemple.

Le Cayenne Low Power Payload (LPP) est un format de données optimisé pour les réseaux LPWAN comme LoRaWAN. Il permet d'envoyer plusieurs types de données de capteurs dans une seule trame, tout en respectant les contraintes de taille de la charge utile. Chaque donnée de capteur est préfixée par deux octets :

— Canal de données : Identifie de manière unique chaque capteur.

— Type de données : Indique le type de données dans la trame.

Résultat

Le microcontrôleur permet de traiter efficacement les données des capteurs. Les données sont formatées de manière optimisée pour la transmission via LoRaWAN, ce qui garantit une communication à faible consommation d'énergie. L'utilisation du protocole Cayenne LPP assure une compatibilité avec la plateforme The Things Network.

3.3 Transmission des données (LoRaWAN)

3.3.1 Communication avec le module LoRa RFM95W

Objectif

Assurer la communication entre le microcontrôleur STM32L432 et le module LoRa RFM95W pour transmettre les données vers The Things Network.

Méthode

La communication entre le STM32L432 et le module RFM95W est réalisée via l'interface SPI. Le module est configuré avec une fréquence de 868 MHz et un Spreading Factor de 9 pour RX2, conformément aux normes européennes. La puissance d'émission est ajustée pour optimiser la portée.

Résultat

La transmission entre le STM32L432 et le module RFM95W est efficace et stable.

3.3.2 Transmission vers The Things Network (TTN)

Objectif

Acheminer les données environnementales capturées par les capteurs vers The Things Network via la passerelle LoRaWAN.

Méthode

L'activation du capteur est réalisée via OTAA (Over-The-Air Activation), ce qui garantit une meilleure sécurité grâce à la génération dynamique des clés de session. Une fois les données encodées en Cayenne LPP, elles sont transmises via LoRaWAN et réceptionnées par TTN.

Résultat

Les données sont correctement acheminées vers TTN et accessibles sur la console de gestion. L'analyse des logs a permis de confirmer la réception régulière des paquets.

3.4 Visualisation et exploitation des données via Datacake

Objectif

Offrir un suivi en temps réel et une analyse des données transmises.

Méthode

Les données qui ont été transmises vers TTN sont ensuite relayées via MQTT vers la plateforme Datacake. Des tableaux de bord personnalisables permettent d'afficher graphiquement l'évolution des paramètres environnementaux et de configurer des alertes en cas de dépassement de seuils critiques.

Résultat

L'utilisateur peut surveiller de façon claire et intuitive grâce à différents graphiques les conditions environnementales en temps réel et être alerté en cas d'anomalie, comme par exemple des dépassements de seuils.

3.5 Alimentation et gestion de l'énergie

Objectif

Assurer une alimentation stable et optimiser la consommation énergétique.

Méthode

L'alimentation s'effectue à 5V via une source externe, puis régularisée en 3,3V pour le microcontrôleur et les capteurs. Les modes basse consommation du STM32 sont activés lorsque les capteurs ne sont pas utilisés.

Résultat

Le fonctionnement est fiable et permet une réduction de la consommation énergétique afin de prolonger l'autonomie.

4-Test et évaluation du système

La vérifiaciton du bon fonctionnement du système s'effectue en plusieurs étapes afin de s'assurer que les données soient bien transmises d'une plateforme à l'autre.

4.1 Transmission à The Things Network

Après avoir établi la connexion avec The Things Network, il faut contrôler la régularité de l'envoi du flot de données, en l'occurence ici The Things Network en reçoit par intervalles de 25 secondes, ainsi que la cohérence de ces données, en les comparant notamment aux ordres de grandeur attendus pour un tel environnement, ou alors en faisant volontairement varier celles-ci, par exemple en positionnant un doigt sur le capteur de température pour augmenter significativement sa valeur. Il faut également s'assurer que le protocole Cayenne LPP a bien été utilisé en comparant le nom correspondant à chaque variable s'affichant sur TTN.

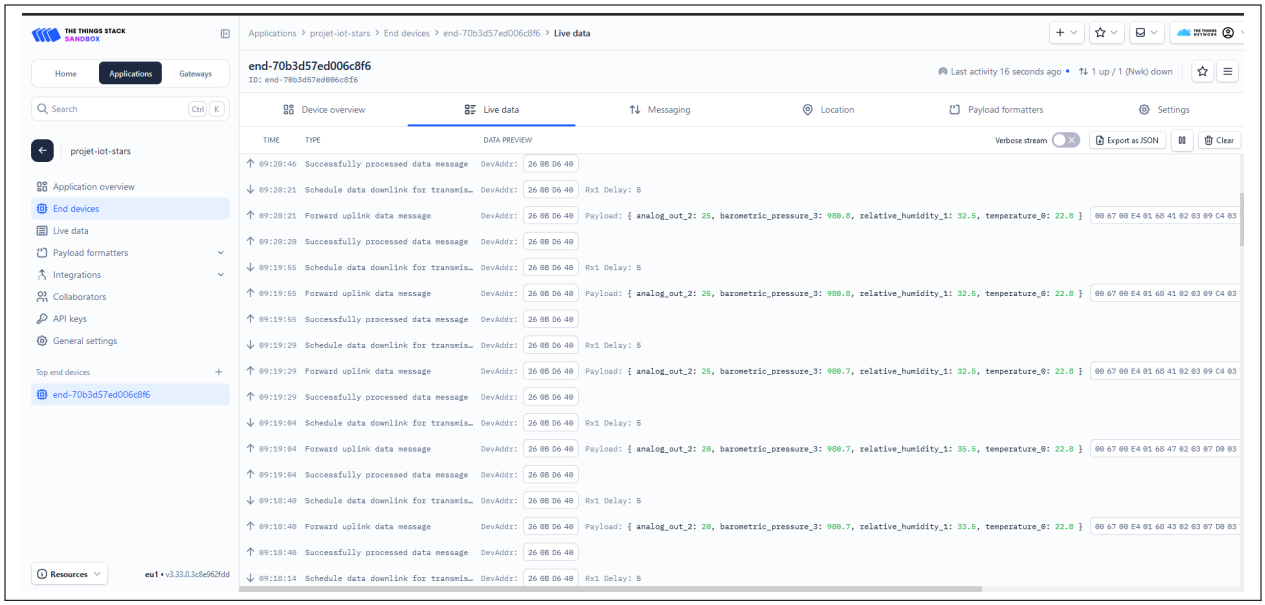

FIGURE 4 – Réception et affichage des valeurs sur TTN

4.2 Transmission à Datacake

Après avoir effectué l'étape précédente, il est à présent possible de les transmettre à Datacake afin de les visualiser. Pour s'assurer que les données sont bien transmises, il est utile d'afficher en permanence la valeur des différentes variables transmises et de tracer les courbes d'évolution de ces dernières au cours du temps. En quelques minutes, la représentation de ces courbes permet d'identifier d'éventuelles valeurs aberrantes.

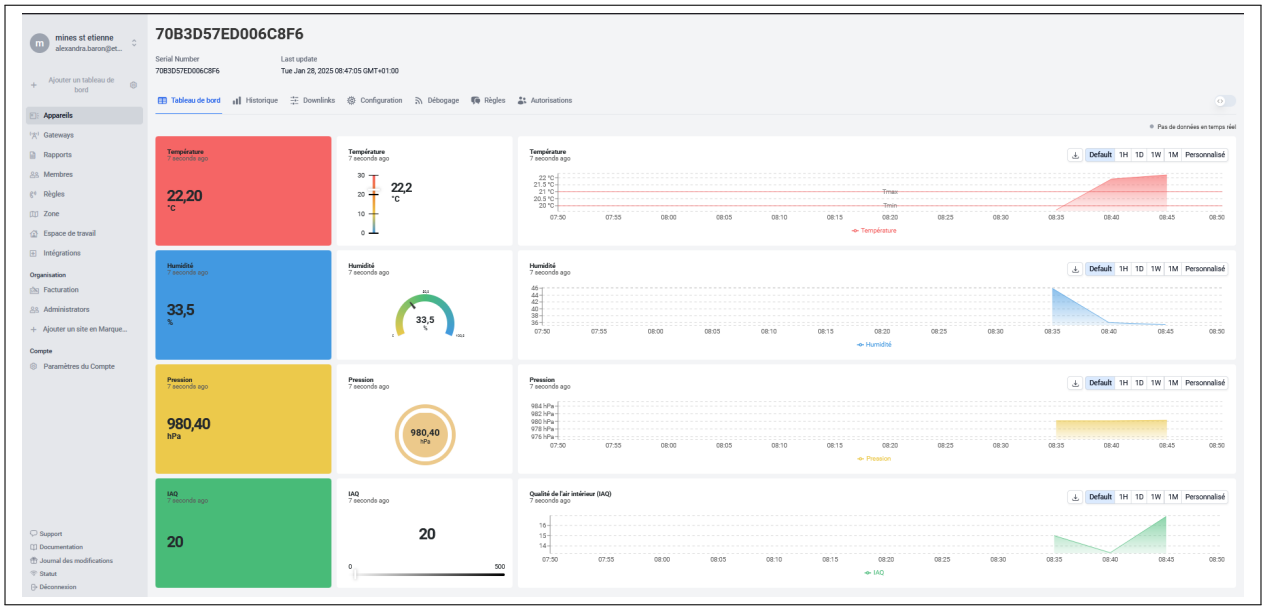

FIGURE 5 – Affichage des données sur DataCake.

Pour une bonne visualisation des limites de température et pression autorisée dans un bloc opératoire, nous avons ajouté des droites - limites qui apparaissent sur les courbes en temps réels.

5-Synthèse

5.1 Résumé des résultats

Ce projet de conception d'un capteur IoT pour la surveillance environnementale en bloc opératoire a permis de mettre en place un système permettant de contrôler différents paramètres tels que la température, l'humidité, la pression et la qualité de l'air.

Le dispositif mis en place repose sur une architecture modulaire intégrant un microcontrôleur STM32L432 et des capteurs spécialisés (LM94021 pour la température, BME680 pour l'humidité, la pression et la qualité de l'air). L'utilisation de la technologie LoRaWAN, couplée à The Things Network (TTN), a permis une transmission efficace et à faible consommation énergétique des données vers une plateforme cloud (Datacake) pour une visualisation en temps réel.

Les tests effectués ont démontré la fiabilité et la régularité de la transmission des données. L'analyse des courbes affichées sur Datacake a confirmé la cohérence des valeurs mesurées. De plus, l'utilisation du format Cayenne LPP a facilité l'interprétation des données et leur affichage structuré.

5.2 Perspectives

Différentes améliorations peuvent être apportées à ce système afin d'en optimiser différents aspects tels que la consommation énergétique, la communication LoRaWAN ou encore le déploiement à une échelle plus large.

- **Amélioration de l'autonomie énergétique :**
  - Mise en place d'un mode repos plus avancé pour le STM32 afin de réduire davantage la consommation d'énergie.
- **Optimisation de la communication LoRaWAN :**
  - Utilisation d'un algorithme adaptatif pour ajuster dynamiquement le Spreading Factor et optimiser la transmission selon les conditions du réseau.
- **Extension des fonctionnalités des capteurs :**
  - Ajout de capteurs supplémentaires pour mesurer d'autres paramètres environnementaux, tels que la détection des composés organiques volatils (COV) ou des niveaux de CO_2, plutôt que d'utiliser seulement la qualité de l'air comme indicateur.
  - Intégration d'un module de stockage local permettant d'enregistrer temporairement les données en cas d'interruption de la communication avec TTN.
- **Déploiement à grande échelle :**
  - Mise en place d'une infrastructure réseau pour la gestion de plusieurs capteurs sur différents blocs opératoires.
  - Intégration avec des systèmes hospitaliers existants pour une gestion centralisée des alertes et des historiques de mesures.
- **Utilisation d'algorithmes d'analyse prédictive :**
  - Développement d'un modèle basé sur l'IA permettant d'anticiper les variations des paramètres environnementaux et d'alerter en cas de conditions anormales.
  - Utilisation des données historiques pour identifier des tendances et recommander des actions préventives.

En conclusion, ce projet constitue une première étape vers une solution efficace de surveillance des environnements hospitaliers. Son évolution future pourrait contribuer à améliorer la gestion des conditions environnementales en milieu médical et renforcer la sécurité des patients et du personnel soignant.

6-Bibliographie

[1] Ladjimi, S., & Maguemoun, L. (2016). *Conception et Réalisation d'un système de contrôle de température, humidité et la qualité de l'air dans un milieu hospitalier* (Mémoire de fin d'études, Master académique en Électronique biomédicale, Université).

[2] Néji, S., Aloulou, M., Trabelsi, H., Sellami, H., Cheikrouhou, F., Triki, Z., Guidara, R., Makni, F., Karoui, A., & Ayadi, A. (2014). *Étude de la flore fongique aérienne des blocs opératoires de l'hôpital de Sfax, Tunisie.* Centre hospitalier universitaire Habib-Bourguiba, Université de Sfax, Tunisie.

[3] Texas Instruments. (2005, June). *LM94021/LM94021Q Multi-Gain Analog Temperature Sensor.*

[4] Tan, H., Wong, K. Y., Othman, M. H. D., Sheng, D. D. C. V., Kek, H. Y., Kuan, G., Lee, K. Q., Wong, S. L., & Deris, M. S. (année inconnue). *Preliminary Assessment of Thermal Comfort in an Operating Room.* Journal of Sustainable Engineering and Built Environment, Faculty of Engineering, Technology, and Built Environment, UCSI University.

7-Annexes

[A] Extrait du Code main.c

```

#include "main.h"
#include "adc.h"
#include "i2c.h"
#include "spi.h"
#include "tim.h"
#include "gpio.h"

#include "hal.h"
#include "lmic.h"
#include "oslmic.h"
#include "lorabase.h"
#include "cayenne_lpp.h"
#include <bme680/bme68x_necessary_functions.h>



#define ADC_RESOLUTION 4095.0  //NEWWWWWWWW
#define myTIMER htim7   //  <--------- change to your setup

// application router ID (LSBF) < ------- IMPORTANT
static const u1_t APPEUI[8]={0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}; // IMPORT FROM
    TTN
// unique device ID (LSBF) < ------- IMPORTANT
static const u1_t DEVEUI[8]={0xF6, 0xC8, 0x06, 0xD0, 0x7E, 0xD5, 0xB3, 0x70};// CREATE RANDOM
    NUMBER
// device-specific AES key (derived from device EUI (MSBF))
static const u1_t DEVKEY[16]={0xE0, 0x35, 0xCD, 0xA6, 0xF8, 0xC2, 0x20, 0xF3, 0xB1, 0x88,
    0x8D, 0x10, 0x96, 0xFF, 0x69, 0xE5};// IMPORT FROM TTN

/* USER CODE END PV */

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;
```

```
}

double GET_temperature(uint32_t ADC_value, double VDD)
{
    double temp_value = 0.0;
    double voltage = 0.0;
    voltage = (ADC_value*VDD)/ADC_RESOLUTION; //int en mV
    temp_value = (1034 - voltage)/5.48; //datasheet p12 a la fin
    return temp_value;
}

double GET_humidite()
{
    struct bme68x_data data;
    bme68x_start(&data, &hi2c1);
    if (bme68x_single_measure(&data) == 0) {
        data.iaq_score = bme68x_iaq(); // Calculate IAQ
        char msgBuffer[120];
        for(uint16_t i = 0; i < 120; i++){
            msgBuffer[i] = ' ';
        }
    }
    return data.humidity;
}

float GET_iaq()
{
    struct bme68x_data data;
    bme68x_start(&data, &hi2c1);
    if (bme68x_single_measure(&data) == 0) {
        data.iaq_score = bme68x_iaq(); // Calculate IAQ
        char msgBuffer[120];
        for(uint16_t i = 0; i < 120; i++){
            msgBuffer[i] = ' ';
        }
    }
    return data.iaq_score;
}

double GET_pression()
{
    struct bme68x_data data;
    bme68x_start(&data, &hi2c1);
    if (bme68x_single_measure(&data) == 0) {

        // Measurement is successful, so continue with IAQ
        data.iaq_score = bme68x_iaq(); // Calculate IAQ

        // Create a message buffer and clear it.
        char msgBuffer[120];
        for(uint16_t i = 0; i < 120; i++){
            msgBuffer[i] = ' ';
        }
    }
    return data.pressure;
}
static int cnt = 0;
//static osjob_t hellojob;

static osjob_t reportjob;
// report sensor value every minute
static void reportfunc (osjob_t* j) {
    double temperature_sensor_val = 25.0;
    double humidity_sensor_val = 0.0;
    double pressure_sensor_val = 0.0;
    double iaq_sensor_val = 0.0;
```

```
104     cayenne_lpp_t lpp = {0};
105     HAL_GPIO_WritePin(Alim_temp_GPIO_Port,Alim_temp_Pin, GPIO_PIN_SET);
106     HAL_Delay(100);
107 
108     // on est aliment , on peut donc lanc  une mesure
109     temperature_sensor_val = GET_temperature(get_ADC_value(hadc1, ADC_CHANNEL_15), 3300);
110     humidity_sensor_val = GET_humidite();
111     iaq_sensor_val = GET_iaq();
112     pressure_sensor_val = GET_pression() / 100;
113 
114     debug_valfloat("Temperature Sensor Value = ", temperature_sensor_val, 6);
115     HAL_GPIO_WritePin(Alim_temp_GPIO_Port, Alim_temp_Pin, GPIO_PIN_RESET);
116     cayenne_lpp_reset(&lpp);
117     cayenne_lpp_add_temperature(&lpp, 0, temperature_sensor_val);
118     cayenne_lpp_add_relative_humidity(&lpp, 1, humidity_sensor_val);
119     cayenne_lpp_add_analog_output(&lpp, 2, iaq_sensor_val);
120     cayenne_lpp_add_barometric_pressure(&lpp, 3, pressure_sensor_val);
121     LMIC_setTxData2(1, &lpp, 15,0); //compter le nombre d'octets qu'on envoie 2 pour dire que
            c'est un capteur et 2 pour la valeur de temp rature
122 
123     os_setTimedCallback(j, os_getTime()+sec2osticks(1), reportfunc);
124 
125 }
126 // void onEvent(ev_t ev) retir  par soucis de presentation
127 int main(void)
128 {
129   HAL_Init();
130   MX_GPIO_Init();
131   MX_SPI3_Init();
132   MX_TIM7_Init();
133   MX_I2C1_Init();
134   MX_ADC1_Init();
135   HAL_TIM_Base_Start_IT(&htim7);
136   __HAL_SPI_ENABLE(&hspi3);
137   osjob_t initjob;
138   os_init();
139   debug_init();
140   os_setCallback(&initjob, initfunc);
141   os_runloop();
142   // (not reached)
143   return 0;
144 }
```