# RAPPORT DU PROJET ÉCOLE À ECOLE DE MINES DE SAINT-ETIENNES

## Réalisation d'un réveil simulateur d'aube

7 Avril 2025 - 20 Juin 2025

Tuteur de stage : Acacio Marques
Promotion : EI23

MINES Saint-Étienne | Institut Mines-Télécom

HALIMA GHANNOUM

# 1
# REMERCIEMENTS

Je tiens à exprimer ma gratitude à mon tuteur, Acacio MARQUES, pour son accompagnement, ses conseils et sa disponibilité tout au long de ce projet. Sa bienveillance et sa pédagogie ont été très appréciées durant cette expérience.

Je remercie également l'École des Mines de Saint-Étienne pour m'avoir offert l'opportunité de réaliser ce projet dans un cadre stimulant, et pour m'avoir permis d'accéder aux ressources et au matériel nécessaires à sa réalisation.

# 2
# INTRODUCTION

Dans le cadre de ma formation d'ingénieur à l'École des Mines de Saint-Étienne, au sein du cycle ISMIN (Ingénieur Systèmes Microélectronique et Informatique), j'ai eu l'opportunité de réaliser un projet école d'une durée de trois mois, axé sur le bien-être et les technologies intelligentes au service du quotidien.

Ce projet s'est fondé sur un constat simple : pour de nombreuses personnes, trouver le sommeil ou se réveiller sans stress est une véritable difficulté. Les réveils brusques, la lumière artificielle soudaine ou un environnement de sommeil mal adapté peuvent nuire à la qualité du repos et rendre les matins pénibles. Pour répondre à cette problématique, nous avons imaginé un simulateur d'aube intelligent, conçu pour accompagner l'utilisateur à la fois lors de l'endormissement et du réveil, en s'adaptant à son rythme et à son environnement.

L'idée de ce dispositif repose sur des bases scientifiques solides. Une étude finlandaise (2003–2004) a montré qu'une lumière simulant le lever du soleil, s'intensifiant progressivement 30 minutes avant l'heure de réveil, améliore significativement la qualité perçue du sommeil. Les effets bénéfiques, ressentis dès la première semaine, disparaissaient à l'arrêt de l'utilisation. Une autre étude, menée en 2014, a démontré que cette lumière douce matinale réduit l'inertie du sommeil et améliore la vigilance, les performances cognitives et physiques dès le réveil. Ces résultats valident notre approche et confirment que la lumière, utilisée intelligemment, peut réellement transformer l'expérience du sommeil.

Concrètement, l'utilisateur programme une alarme depuis son téléphone. Quelques minutes avant l'heure prévue, des lumières douces s'allument progressivement, imitant le lever du soleil et favorisant un réveil naturel. Le soir, une lumière tamisée est diffusée pour aider à l'endormissement. Mais le système va plus loin : durant la nuit, il observe l'environnement et le comportement de l'utilisateur. Il analyse ses mouvements pour évaluer la qualité de son sommeil, surveille la température et

la qualité de l'air, et peut, si nécessaire, activer automatiquement un ventilateur pour améliorer le confort de la pièce.

Pensé pour être discret, autonome et bienveillant, ce simulateur d'aube vise à créer un environnement plus sain et plus naturel, en harmonie avec le rythme biologique de l'utilisateur. Il s'agit d'une solution technologique douce, respectueuse du sommeil, qui allie confort, santé et innovation.

Dans un premier temps, je reviendrai sur l'apport de la formation d'ingénieur ISMIN en microélectronique et informatique. Je présenterai ensuite les différents composants matériels utilisés, en détaillant leur rôle et leur fonctionnement au sein du système. Dans un troisième temps, j'exposerai l'architecture complète du projet, en décrivant les aspects techniques, les choix réalisés, ainsi que l'organisation générale à travers des schémas synoptiques et organigrammes. Enfin, je conclurai ce rapport en abordant les perspectives d'amélioration du dispositif, les défis rencontrés et les éléments qui restent à développer ou à finaliser.

# 3
# PRÉSENTATION D'ISMIN ET DE SON ENVIRONNEMENT

## 3.1 - Ecole de Mines de SAINT -ETIENNES

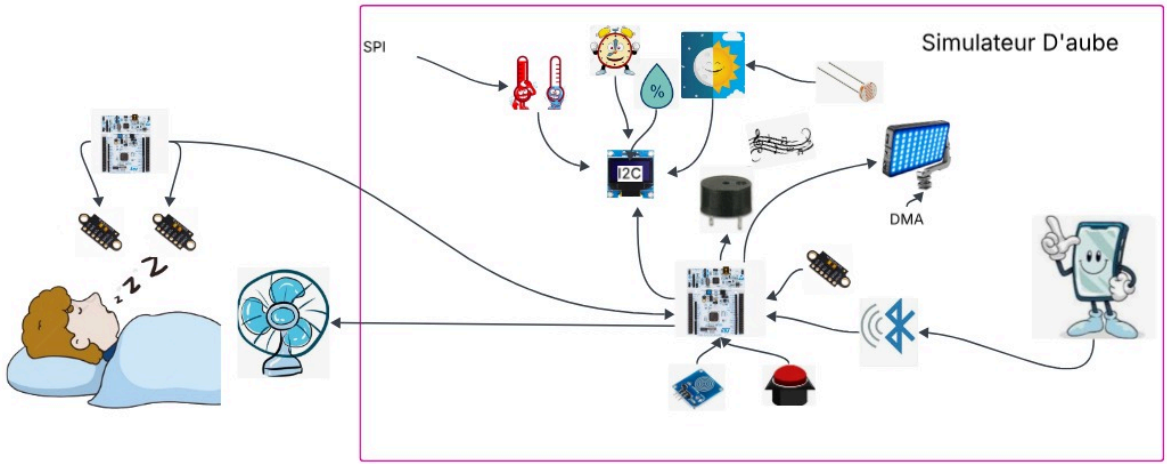
L'École des Mines de Saint-Étienne, fondée en 1816, est l'une des plus anciennes et prestigieuses écoles d'ingénieurs en France. Elle fait partie de l'Institut Mines-Télécom, un regroupement d'écoles d'excellence. Reconnue pour la qualité de sa formation et de sa recherche, l'école s'inscrit dans une tradition d'excellence scientifique tout en s'adaptant aux évolutions technologiques. Le campus Georges Charpak Provence, situé à Gardanne près d'Aix-en-Provence, accueille le cursus ISMIN, dédié aux systèmes intelligents et connectés. Ce campus moderne favorise l'apprentissage par projets, les partenariats industriels et l'ouverture internationale. C'est dans ce cadre stimulant que j'ai eu l'opportunité de développer mes compétences en microélectronique et informatique.

## 3.2 - Programme Ismin

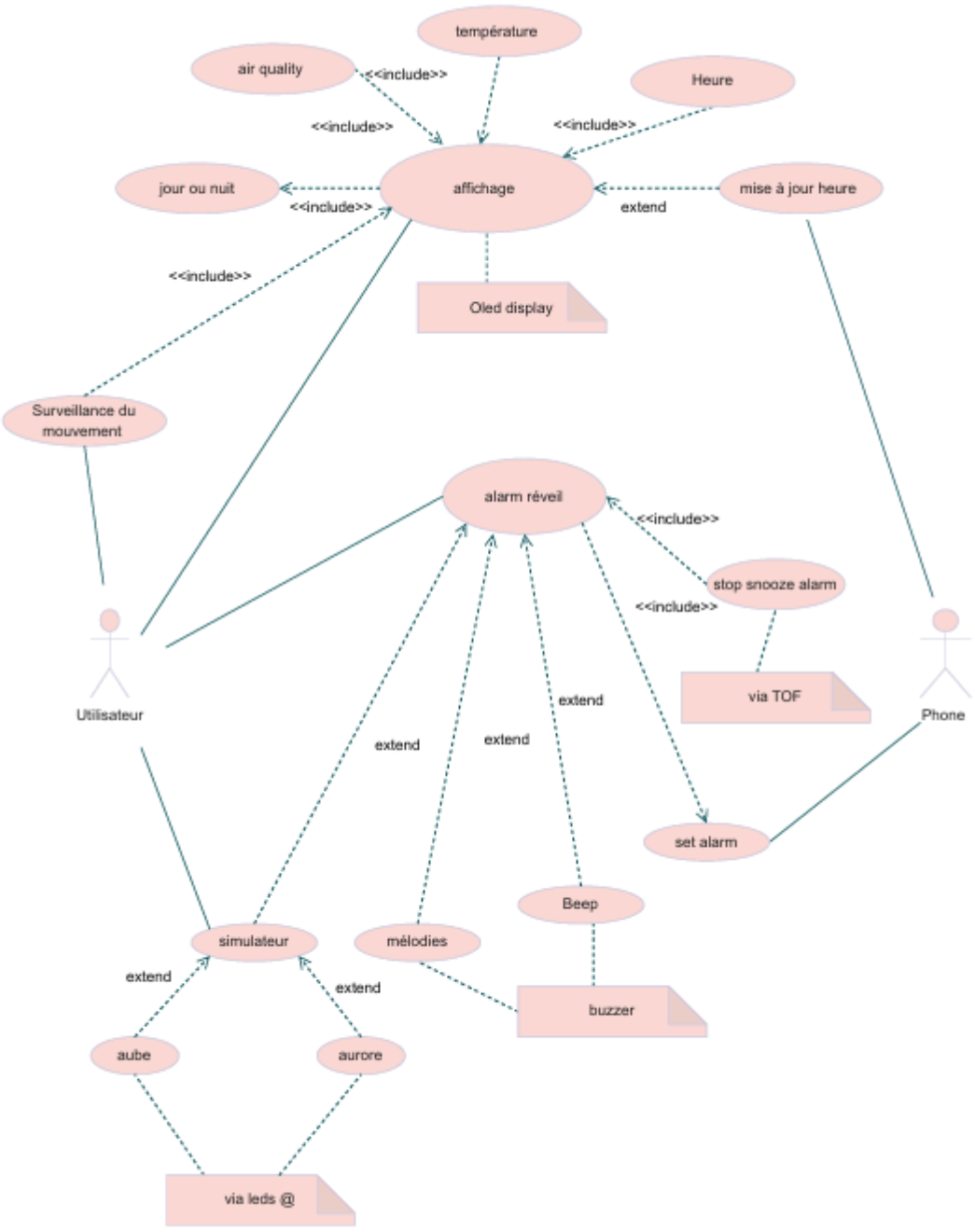
Ce programme met l'accent sur les systèmes embarqués, les architectures numériques, la cybersécurité, les réseaux et l'ingénierie logicielle. Parmi les enseignements clés, les cours de microprocesseurs jouent un rôle central. Il permettent aux étudiants de comprendre en profondeur le fonctionnement interne des processeurs, l'architecture des systèmes à base de microcontrôleurs, ainsi que la programmation bas niveau en langage assembleur et C. Ils sont indispensable pour concevoir et développer des systèmes électroniques intelligents et performants, comme ceux impliqués dans ce projet.

# 4
# DESCRIPTION DU PROJET

## 4.1 - Synoptique




Le synoptique ci-dessus présente l'architecture générale du simulateur d'aube. Il illustre les interactions entre les capteurs (température, qualité de l'air, lumière, distance), les modules de commande (Bluetooth, tactile, horloge) et les actionneurs (buzzer, LEDs RGB, affichage OLED). Une deuxième carte, placée au-dessus du dormeur, mesure la position du corps via deux capteurs ToF et transmet les données à la carte principale pour une analyse du sommeil.




## 4.2 - Microcontrolleur Utilisé

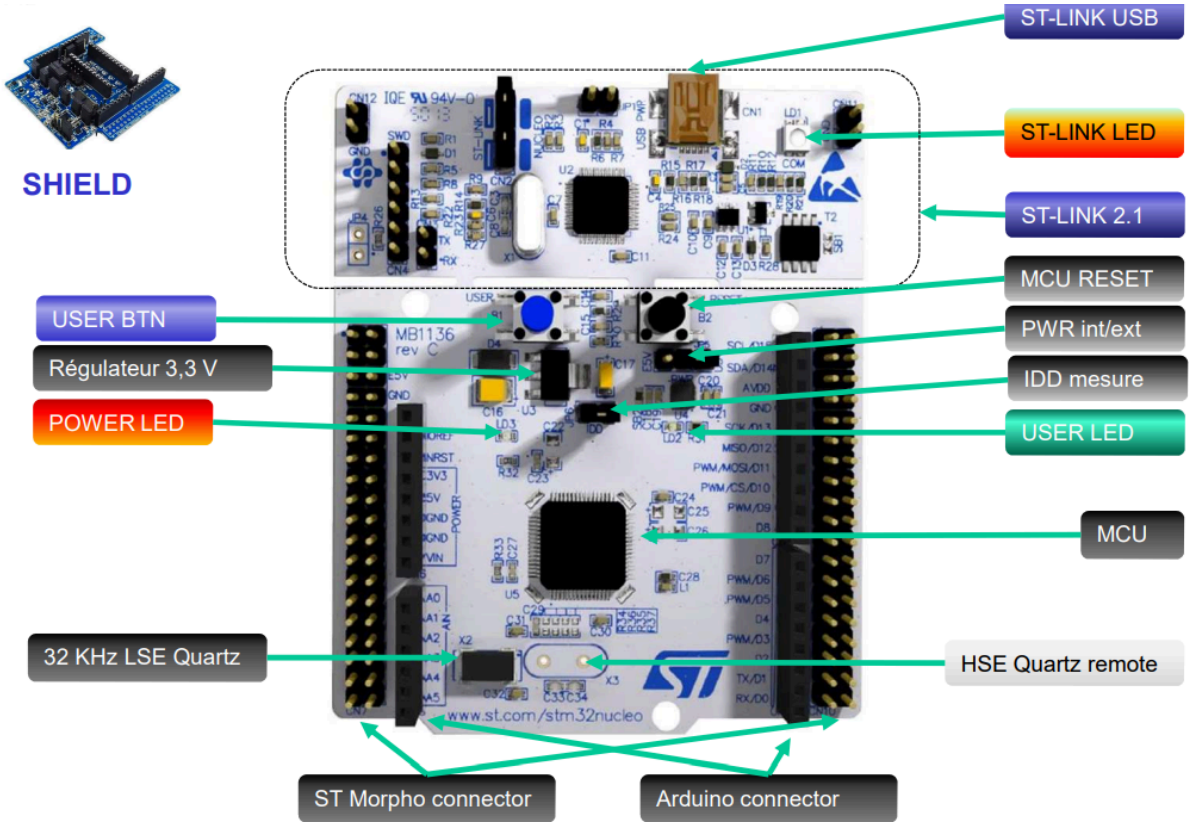
Rapport du projet école

Le choix de STM32L476 a été motivé par plusieurs critères techniques répondant aux demandes du projet. Tout d'abord, le STM32L476 est basé sur un cœur ARM Cortex-M4, reconnu pour sa bonne performance en calcul tout en conservant une faible consommation énergétique. Il s'agit d'un cœur 32 bits cadencé jusqu'à 80 MHz, suffisant pour exécuter des tâches de traitement de signaux, de gestion d'interruptions et de contrôle temps réel.

Ce microcontrôleur embarque 1 Mo de mémoire Flash pour stocker le code programme et les constantes, ainsi que 128 Ko de SRAM pour les données dynamiques, la pile, et le tas. Cette mémoire est organisée de façon intelligente en deux banques de Flash (2 × 512 Ko), permettant la mise à jour d'une banque pendant que l'autre continue de s'exécuter. De même, la mémoire SRAM est divisée en deux segments (SRAM1 et SRAM2), avec une gestion optimisée pour l'exécution du code sans cycle d'attente.

Le STM32L476 offre également une large gamme de périphériques intégrés, ce qui permet d'interfacer facilement différents composants électroniques. Parmi ceux-ci, on trouve des interfaces de communication série (UART, I²C, SPI), des convertisseurs analogique/numérique (ADC), des Timers, des contrôleurs PWM pour piloter des actionneurs, ainsi que des modules de gestion de l'alimentation (watchdog). Cette richesse fonctionnelle limite le besoin de composants externes supplémentaires.

Enfin, Le développement peut être réalisé à l'aide d'outils comme STM32CubeIDE, qui intègre un environnement complet pour l'écriture du code, le débogage, la configuration des périphériques via STM32CubeMX, et l'intégration des bibliothèques HAL.




## 4.3 - Carte Electronique

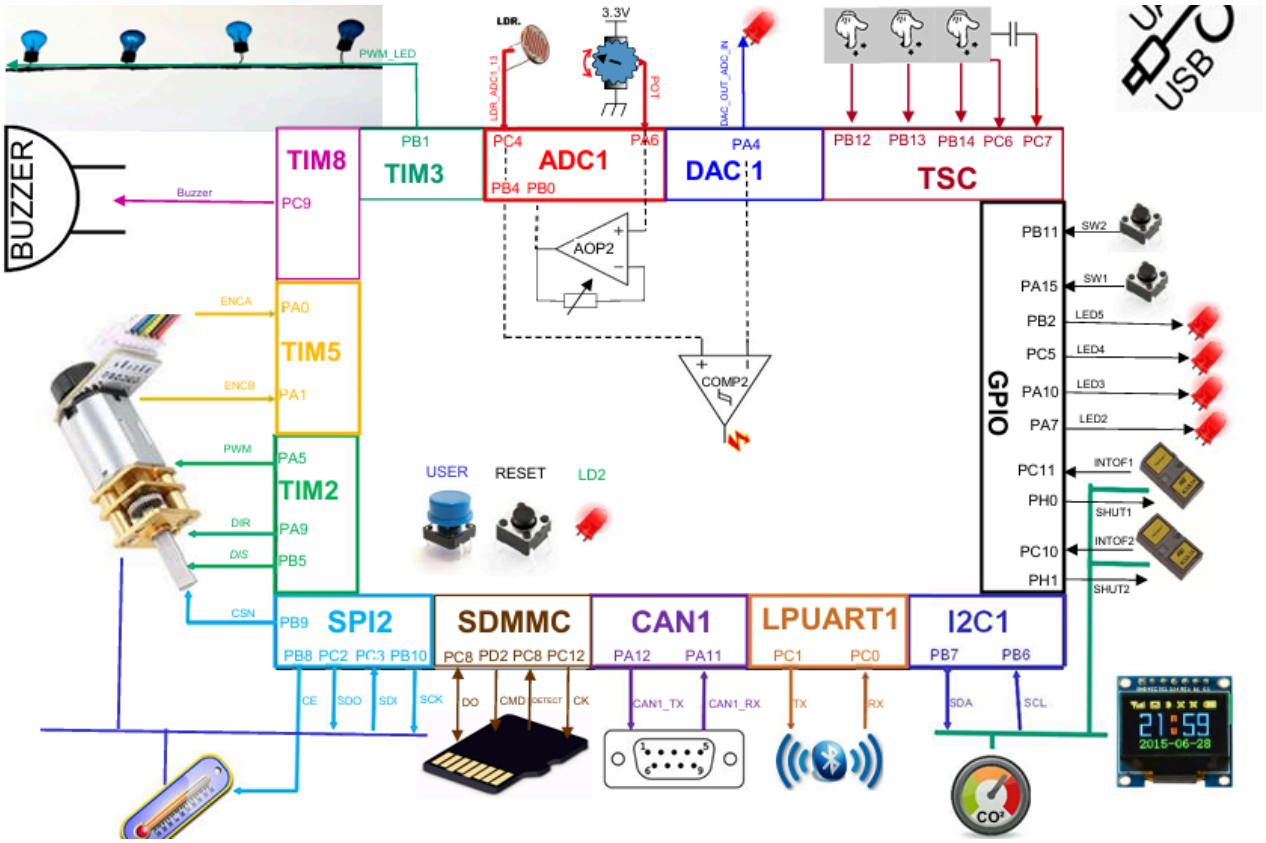
Pour la réalisation de ce projet, j'ai utilisé une carte conçue par mon tuteur à l'école, spécialement pensée pour les travaux pratiques et les projets embarqués. Elle intègre le microcontrôleur STM32L476 ainsi qu'un ensemble complet de périphériques directement disponibles sur une seule carte PCB.

Ce choix m'a permis de gagner un temps précieux, en évitant les étapes longues d'impression de circuits, de soudure, ou de débogage matériel. La carte contient déjà tout ce dont j'avais besoin pour mon application :

- des LEDs pour la simulation de l'aube,
- un module Bluetooth pour recevoir l'heure de réveil,
- des capteurs de température et de qualité de l'air,
- un moteur jouant le rôle de ventilateur si l'environnement devient défavorable,

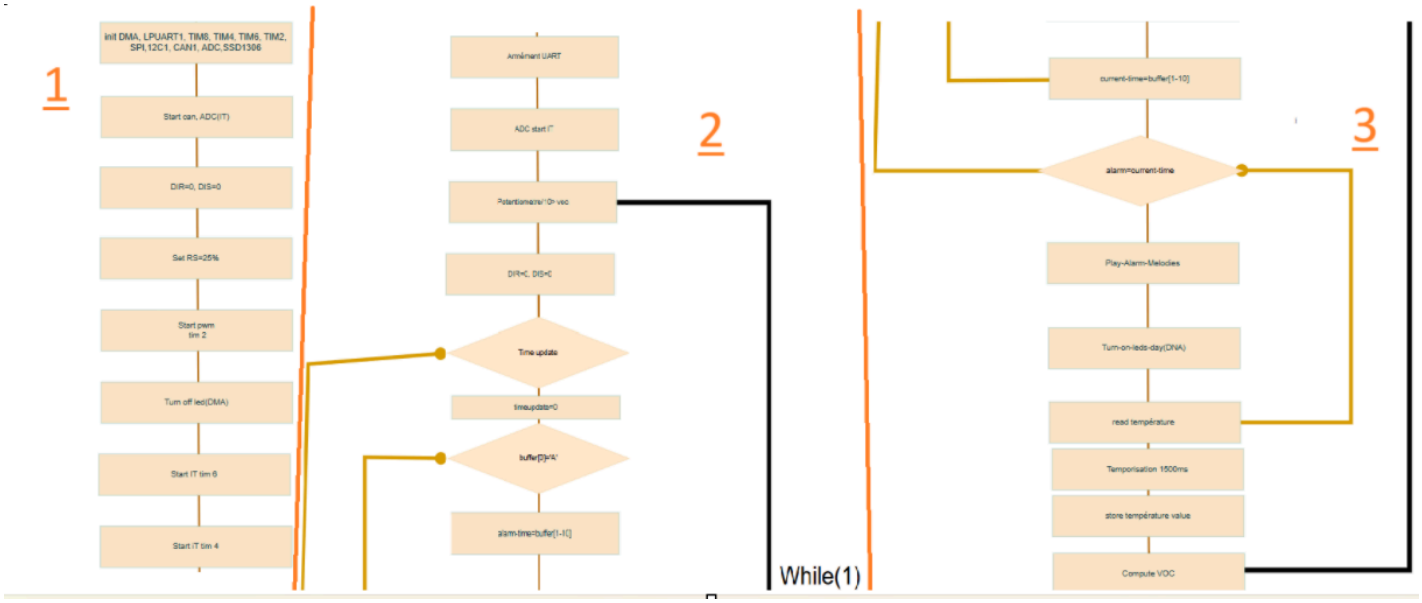
- un buzzer pour sonner l'alarme,
- un écran OLED pour l'affichage,
- des boutons pour le choix de paramètres d'affichage,
- une interface CAN pour la communication avec la deuxième carte de détection de mouvement,
- des capteurs TOF pour détecter la position de l'utilisateur ,
- un photorésistance pour détecter jour ou nuit,
- ainsi qu'un potentiomètre pour contrôler le niveau du ventilateur

Grâce à cette carte conçue pour les étudiants, j'ai pu me concentrer sur le développement logiciel et la logique fonctionnelle du système, tout en disposant d'un support matériel complet.



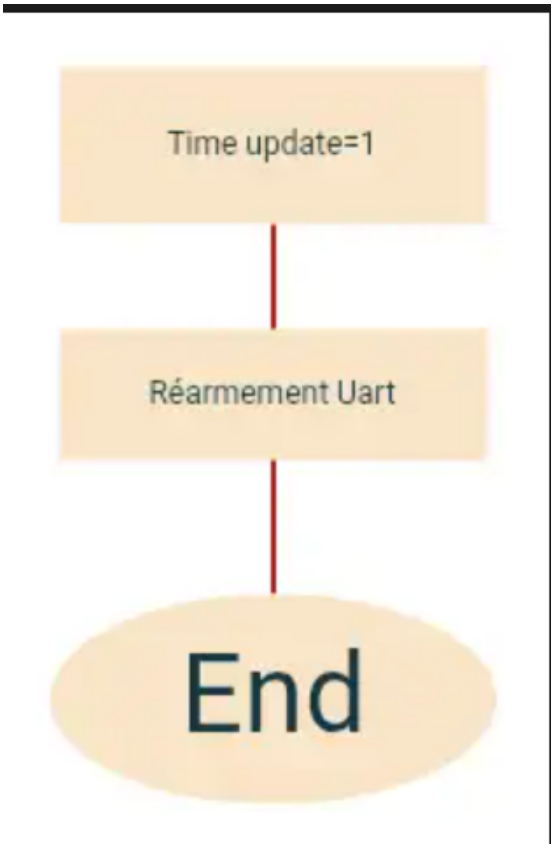

## 4.4 - Organigramme

Rapport du projet école




Pour commencer, j'ai configuré tous les périphériques nécessaires : GPIO, ADC, I2C, SPI, Bus Can, UART, timers, etc. Le convertisseur analogique-numérique (ADC) me permet de lire deux capteurs : un potentiomètre et une photo-résistance (LDR) en mode interruption.

Les LEDs adressables sont pilotées en PWM grâce à TIM2, avec un transfert DMA sur le tableau PwmRC[], ce qui permet un allumage progressif sans bloquer le reste du programme. Les timers jouent un rôle important : TIM6 met à jour l'heure logicielle toutes les secondes via HAL_TIM_PeriodElapsedCallback() et TIM4 me sert pour générer des temporisations utilisées dans les animations et le changement d'affichage pour eviter de bloque le systéme avec HAL_Delay().



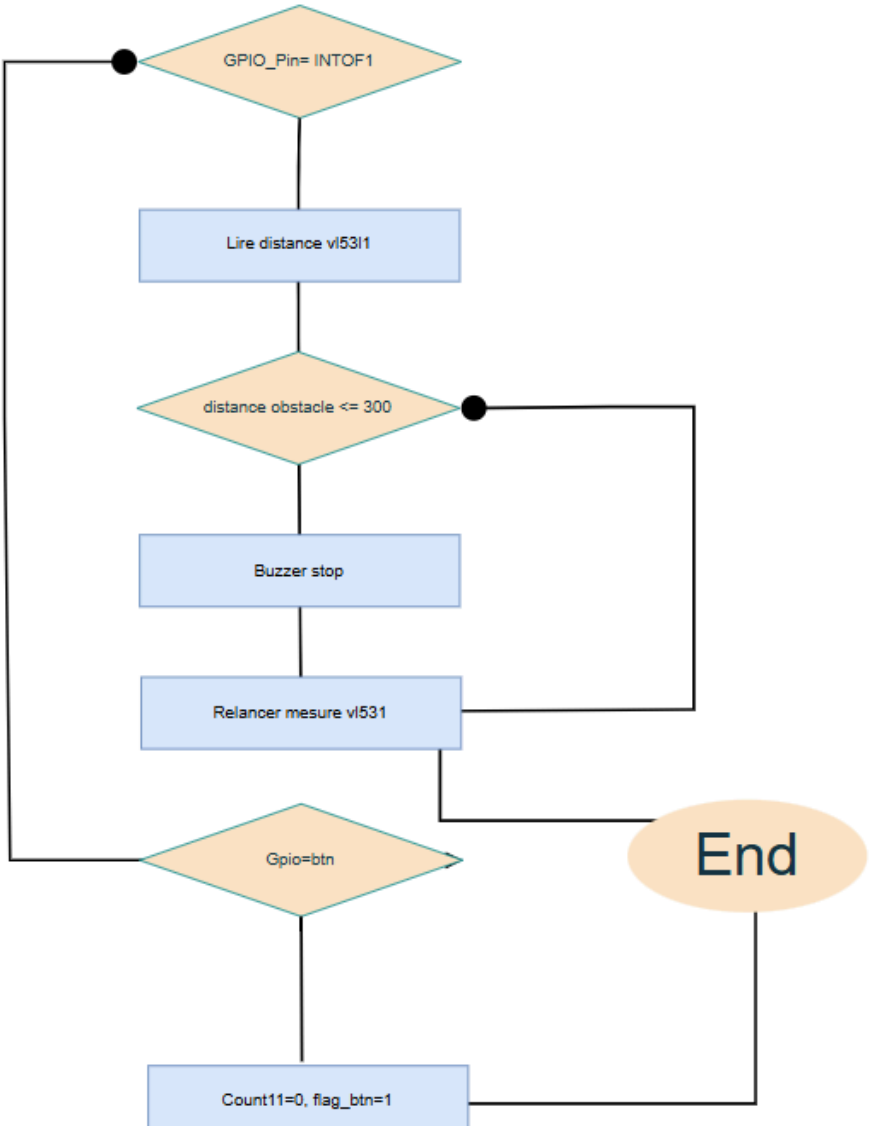

Côté communication, j'utilise LPUART1 pour recevoir des trames Bluetooth envoyées depuis une application mobile. Le message reçu est traité dans HAL_UART_RxCpltCallback(), où je vérifie s'il s'agit d'un réglage de l'heure (T:…) ou d'une alarme (A:…). J'utilise sscanf() pour parser les données et mettre à jour les structures currentTime ou alarmTime.

Quand l'heure atteint l'alarme programmé, une séquence d'éveil se déclenche : le buzzer joue une mélodie via buzzer_ePlaySong() et la fonction progressive_led_on() allume les LEDs une par une, en augmentant la luminosité toutes les 5 secondes en utilisant switch-case qui sert à juste augmenter le compteur pour les LEDs. Pour rendre l'interface plus dynamique, j'ai ajouté une touche tactile capacitive gérée par le module TSC. Elle est lue avec isStableTouch(), qui applique un filtrage logiciel avec le count11 (10 fois periode tim4= 110ms) pour éviter les déclenchements parasites. Chaque appui stable change l'état de l'affichage à travers la variable sw. Quand sw vaut 0, j'affiche la qualité de l'air ; à sw = 1, j'affiche l'heure ; à sw = 2, la température ambiante. Et pour sw = 3 :l'utilisateur indique son heure de coucher, et une séquence

de nuit est lancée. Dans cette séquence, les LEDs s'allument progressivement mais cette fois avec une lumière rouge, pour simuler un coucher de soleil.

La température est mesurée via SPI : j'envoie une commande avec HAL_SPI_Transmit() puis je récupère les données avec HAL_SPI_Receive(). Je reconstitue ensuite la température à partir de msbvalue et lsbvalue our la virgule flottante . Cette température est aussi utilisée pour ajuster la mesure de la qualité de l'air avec le capteur SGP40 (I2C). La fonction SGP40_MeasureVOC() prend en compte la température pour renvoyer une valeur Sraw_voc, que j'affiche ensuite sur l'écran OLED.

Pour l'affichage, j'utilise l'écran OLED via la librairie SSD1306. Je rafraîchis l'écran avec SSD1306_UpdateScreen() après avoir écrit le contenu avec SSD1306_Puts() et SSD1306_GotoXY(). Je change dynamiquement entre les affichages (heure, température, qualité de l'air) selon l'entrée tactile ou les boutons. Ainsi, le contraste d l'écran est changé selon le résultat du photorésistance (jour ou nuit).




Un autre élément important du système est le capteur ToF VL53L1X, que j'ai configuré pour détecterle main de l'utilisateur (< 30 cm) pour arrêter la mélodie. Je l'initialise avec ResetAndInitializeTOF1() où je configure notamment le ROI (zone de détection), le mode de mesure, le timing et les seuils. Lorsqu'une présence est détectée, une interruption GPIO est déclenchée (HAL_GPIO_EXTI_Callback()), ce qui

permet d'arrêter automatiquement le buzzer avec buzzer_vStop() — ainsi l'utilisateur n'a même pas besoin d'appuyer sur un bouton.

Sur une autre carte identique, placée au-dessus de l'utilisateur pendant qu'il dort, j'utilise les deux capteurs ToF afin de mesurer en continu la position et la direction

du corps. En analysant la distance mesurée par chaque capteur, je peux estimer si la personne a beaucoup bougé pendant la nuit. Ces données sont ensuite envoyées via Bus CAN vers la carte principale du simulateur d'aube, qui les traite pour afficher une indication sur l'écran OLED, précisant si la qualité du sommeil a été bonne ou perturbée.

La boucle while(1) tourne en continu, elle gère les lectures de capteurs, les comparaisons d'heure, l'activation des alarmes, les mises à jour d'affichage et la recepion des données (via CAN et BLE).

## 4.5 - Fonctionnement des Périphériques

Dans cette partie, chaque périphérique de la carte utilisé dans le système sera décrit en termes de fonctionnement et de configuration.

### 4.5.1 • Contrôle des LEDs adressables et DMA

Les LEDs nécessitent un signal précis transmis via DMA pour éviter de bloquer le CPU.

#### 4.5.1.1 - Principe de fonctionnement

La couleur d'une LED est codée sur 3 composantes codées en uint8_t : Green, Red et Blue (mélange de 3 couleurs primaires)

Chacune de ces valeurs, entre 0 et 255, détermine l'intensité de la couleur primaire correspondante (0 : couleur éteinte, 255 : couleur au max d'intensité). Considérons une de ces composantes. On écrit sa valeur en donnant une séquence de 8 valeurs de CCR, 1 par bit, avec :

- bit i = 1 encodé par CCR tel que rapport cyclique = 64% (pour i de 0 à 7)
- bit i = 0 encodé par CCR tel que rapport cyclique = 32% (pour i de 0 à 7) DMA .

#### 4.5.1.2 - Initialisation

on configure le DMA memory to périphérique.

```
MX_TIM3_Init();
HAL_TIM_Base_Start_DMA(&htim4, (uint32_t *)PwmRC, k);
```

#### 4.5.1.3 - Génération des données pour les LEDs

La fonction suivante prépare les impulsions à envoyer, en fonction du nombre de LEDs à allumer (num_on) :

```
void set_leds(uint8_t num_on) {
    int k = 0;
    for (int i = 0; i < 4; i++) {
        uint8_t p = (i < num_on) ? 1 : 0;
        for (int j = 0; j < 24; j++) {
            PwmRC[k++] = (p == 0) ? 32 : 64; // 32 = 0 logique, 64 = 1
logique
        }
    }

    for (int i = 0; i < 50; i++) {
        PwmRC[k++] = 0; // pause entre deux cycles
    }

    datasentflag = 0;
    HAL_TIM_PWM_Start_DMA(&htim3, TIM_CHANNEL_4, (uint32_t *)PwmRC, k);
}
```

### 4.5.2 • Sonner une mélodie avec le buzzer

Le morceau joué est codé dans un tableau de type `buzzernote` :

```
const buzzernote nokia[] = {
  {16, 0, 1, {0, 0}}, {14, 0, 1, {0, 0}}, {9, 0, 1, {0, 0}}, ...
  {0, 0, 0, {0, 0}} // marque de fin
};
```

Chaque note est définie par :

note : index dans le tableau des fréquences

sharp : dièse (1 = ♯)

duration : durée relative

{0, 0} : (réservé pour extensions)

#### 4.5.2.1 - Fréquences musicales

Un tableau contient les fréquences des notes :

```
static const float note_freqs[18] = {
  0.0, 261.63, 277.18, 293.66, ..., 659.25
};
```

#### 4.5.2.2 - Démarrage

```
void buzzer_vInit();
```

Lance le timer en mode PWM sur le canal TIM_CHANNEL_4 de TIM8.

#### 4.5.2.3 - Lecture d'une note

```
static void play_note(uint8_t note, uint8_t sharp, uint8_t duration,
uint16_t tempo);
```

Cette fonction calcule la fréquence, configure le timer pour générer un signal carré, attend la durée de la note, puis arrête le son.

#### 4.5.2.4 - Jouer une mélodie

```
buzzer_status_t buzzer_ePlaySong(uint16_t tempo);
```

Joue 5 fois la mélodie nokia[], sauf si buzzer_stop_flag est activé.

#### 4.5.2.5 - Arrêt

```
void buzzer_vStop(void);
Active le flag d'arrêt (buzzer_stop_flag = 1)
```

Couper la PWM immédiatement

#### 4.5.2.6 - Génération du signal

Le signal PWM est généré comme suit :

```
uint32_t period = timer_clk / (freq * prescaler);
uint32_t pulse = period / 2; // signal 50%
```

Puis injecté dans le timer avec :

```
__HAL_TIM_SET_AUTORELOAD(...)
__HAL_TIM_SET_COMPARE(...)
```

### 4.5.3 • TOF

Le capteur ToF est initialisé via la fonction `ResetAndInitializeTOF1()` :

- Réinitialisation matérielle par la broche `SHUT1`.
- Configuration de l'adresse I2C (par défaut `0x52`, changée ici en `0x54` pour le deuxiéme capteur pour ne pas confondre avec le premier capteur).
- Initialisation logicielle via les fonctions :
  - `VL53L1_WaitDeviceBooted`
  - `VL53L1_DataInit`
  - `VL53L1_StaticInit`
- Configuration :
  - Mode de distance : `VL53L1_DISTANCEMODE_LONG`
  - Budget de mesure : `40 ms`
  - Période d'inter-mesure : `50 ms`

#### 4.5.3.1 - Zone de détection (ROI)

La zone de détection est définie comme suit :

TopLeft: (0, 15) BotRight: (15, 0)

#### 4.5.3.2 - Mode de détection d'un objet

Le capteur est configuré pour générer une interruption si la distance mesurée est comprise entre 20 mm et 300 mm.

```
DetectionConfig.Distance.Low = 20;
DetectionConfig.Distance.High = 300;
DetectionConfig.Distance.CrossMode = VL53L1_THRESHOLD_IN_WINDOW;
```

#### 4.5.3.3 - Détection et traitement

Lorsqu'un objet est détecté dans la plage définie, une interruption GPIO est déclenchée :

```
if (GPIO_Pin == INTOF1_Pin) {
  VL53L1_GetRangingMeasurementData(&devLeft, &RangingData);
  Dist_Obst_G = RangingData.RangeMilliMeter;
  if (Dist_Obst_G <= 300) {
    buzzer_vStop();
  }
  VL53L1_ClearInterruptAndStartMeasurement(Dev);
}
```

### 4.5.4 • Oled

L'écran OLED est basé sur le contrôleur **SSD1306**, communiquant via le bus **I2C**. Il permet d'afficher l'heure, la température, la qualité de l'air et des messages de réveil ou de sommeil.

#### 4.5.4.1 - Initialisation

Les fichiers de la librairies doivent être ajoutés dans le répertoire Include.

On configure I2C à fast speed pour bien afficher les données sans erreurs.

L'écran est initialisé en début de programme avec la fonction :

```
SSD1306_Init();
```

#### 4.5.4.2 - Affichage d'un texte

Pour écrire du texte à l'écran, on utilise la séquence suivante :

```
SSD1306_GotoXY(x, y);
SSD1306_Puts("Texte", &Font_16x26, color);
SSD1306_UpdateScreen();
```

**x, y** position du texte (en pixels).

**Font_16x26** taille de la police

**color**

- 1 = texte blanc (mode normal)
- 0 = texte noir sur fond noir → utilisé pour réduire le contraste la nuit

### 4.5.5 • BUs CAN

Le microcontrôleur utilise le bus CAN pour échanger des messages avec l'autre carte. Le filtre est configuré pour accepter les messages ayant l'identifiant `0x0220`. Lorsqu'un message est reçu, il est traité dans l'interruption `HAL_CAN_RxFifo0MsgPendingCallback()`

Le système active également les interruptions de réception (RX_FIFO0) et de transmission (TX_MAILBOX_EMPTY) pour gérer la communication de manière réactive et efficace :

```
HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING);
HAL_CAN_ActivateNotification(&hcan1, CAN_IT_TX_MAILBOX_EMPTY);
```

#### 4.5.5.1 - Envoi de messages CAN

L'envoi de messages sur le bus CAN se fait en utilisant la fonction `HAL_CAN_AddTxMessage()`, qui prend en paramètre l'en-tête (`CAN_TxHeaderTypeDef`) et les données à transmettre (`uint8_t[]`). Voici un exemple typique :

```
CAN_TxHeaderTypeDef TxHeader;
uint8_t TxData[2];
uint32_t TxMailbox;

TxHeader.StdId = 0x0220;
TxHeader.RTR = CAN_RTR_DATA;
TxHeader.IDE = CAN_ID_STD;
TxHeader.DLC = 2;

TxData[0] = (message >> 8) & 0xFF;
TxData[1] = message & 0xFF;

HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox);
Ce code envoie un message de 2 octets contenant une valeur sur
l'identifiant 0x0220. Le TxMailbox permet au HAL de gérer
```

```
automatiquement la file d'attente des messages en attente de
transmission.
```

### 4.5.6 • SPI et mesure de la qualité de l'air et la température

#### 4.5.6.1 - Configuration du SPI

Le capteur de température utilisé dans cette carte communique via le protocole SPI. La configuration du périphérique SPI se fait dans CubeMX en activant le module `SPI2` en mode master.

#### 4.5.6.2 - Lecture de la température via SPI

on envoi une commande one-shot, puis on lit des registres contenant le résultat.

```
uint8_t control_T[2] = {0x80, 0x15};  // Commande one-shot
uint8_t Temp_Address = 0x02;          // Adresse de lecture
uint8_t Temp[3];                      // Données reçues

// Étape 1 : Lancer la mesure
HAL_GPIO_WritePin(CE_GPIO_Port, CE_Pin, GPIO_PIN_SET);
HAL_SPI_Transmit(&hspi2, control_T, sizeof(control_T), HAL_MAX_DELAY);
HAL_GPIO_WritePin(CE_GPIO_Port, CE_Pin, GPIO_PIN_RESET);

// tepmorisation entre la commande et la lecture(faite aprés avec du
timer)

// Étape 2 : Lecture du registre
HAL_GPIO_WritePin(CE_GPIO_Port, CE_Pin, GPIO_PIN_SET);
HAL_SPI_Transmit(&hspi2, &Temp_Address, 1, HAL_MAX_DELAY);
HAL_SPI_Receive(&hspi2, Temp, 3, HAL_MAX_DELAY);
HAL_GPIO_WritePin(CE_GPIO_Port, CE_Pin, GPIO_PIN_RESET);
```

Une fois les données reçues, la température est reconstruite à partir des bits reçus tout en activant la lecture de virgule flottante dans les paramètres du projet:

```
uint8_t msbvalue = Temp[0];
uint8_t lsbvalue = Temp[1];
float fTemperature;

if (lsbvalue & 0x80)
    fTemperature = msbvalue + 0.50;
```

```
else if (lsbvalue & 0x40)
    fTemperature = msbvalue + 0.25;
else
    fTemperature = msbvalue;
```

#### 4.5.6.3 - Utilisation de la librairie SGP40

Le capteur SGP40 communique via I2C et fournit une mesure de la qualité de l'air (VOC). Une librairie dédiée est utilisée (SGP40.h) et ajouté dans le répertoire Inlude . on utilise la fonction suivante de la librairie pour obtenir l'indice VOC :

```
Sraw_voc = SGP40_MeasureVOC(fTemperature, 50);
```

### 4.5.7 • Interfaces tactiles

Le microcontrôleur STM32 dispose d'un module intégré appelé **TSC** qui permet de mesurer la variation de capacité sur une électrode, causée par la présence d'un doigt. Voici les étapes nécessaires à la lecture d'une valeur tactile.

#### 4.5.7.1 - Configuration des broches (via CubeMX)

- Activation du périphérique `TSC`
- Définition des broches :
  - `ChannelIOs` : `TSC_GROUP1_IO1`
  - `SamplingIOs` : `TSC_GROUP1_IO4`

#### 4.5.7.2 - Initialisation du module TSC dans le code

```
htsc.Instance = TSC;
htsc.Init.ChannelIOs = TSC_GROUP1_IO1;
htsc.Init.SamplingIOs = TSC_GROUP1_IO4;
```

#### 4.5.7.3 - Préparation à la mesure

Avant de lancer la mesure, une décharge des électrodes est nécessaire :

```
HAL_TSC_IODischarge(&htsc, ENABLE);
HAL_Delay(1);
HAL_TSC_IODischarge(&htsc, DISABLE);
```

#### 4.5.7.4 - Démarrage de la mesure et lecture

On démarre ensuite le module TSC et on attend que la mesure soit prête :

```
HAL_TSC_Start(&htsc);
while (HAL_TSC_GetState(&htsc) != HAL_TSC_STATE_READY);
value_group1 = HAL_TSC_GroupGetValue(&htsc, TSC_GROUP1_IDX);
HAL_TSC_Stop(&htsc);
```

#### 4.5.7.5 - Détection de contact tactile

La valeur mesurée est comparée à une valeur de référence . Si la différence dépasse un seuil prédéfini (TOUCH_THRESHOLD), un appui est détecté :

```
if (abs(value_group1 - baseline_group1) > TOUCH_THRESHOLD) {
    Touche détectée
}
```

## 4.6 - Choix des protocoles de communication

Le tableau suivant compare les principaux protocoles de communication série utilisés dans les systèmes embarqués : UART, SPI et I2C.

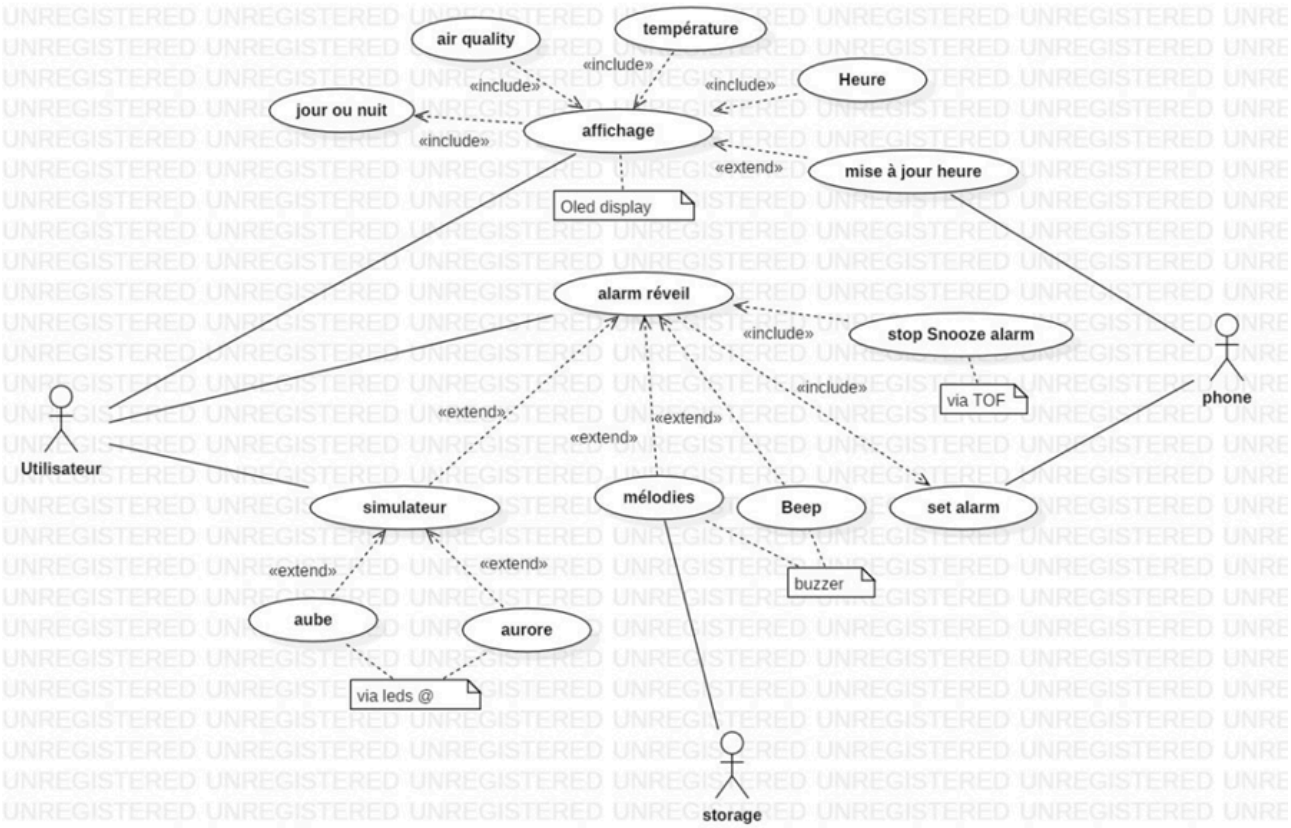
Ce tableau met en évidence les différences en termes de complexité, de vitesse, de distance, de consommation et d'usage. En s'appuyant sur ces caractéristiques, les choix suivants ont été faits dans notre projet :

- Le capteur de température a été connecté via le bus SPI car ce protocole permet un transfert rapide des données et une communication fiable en courte distance (quelques centimètres), ce qui convient parfaitement à la fréquence de lecture nécessaire pour ce capteur.
- L'écran OLED utilise le bus I2C, qui permet de chaîner plusieurs périphériques sur deux fils seulement, tout en consommant peu d'énergie.
- La communication entre deux cartes est assurée par le bus CAN, qui est spécialement conçu pour les systèmes distribués embarqués. Il permet une communication robuste, multi-maître, avec détection d'erreur.

| Protocol | UART | SPI | L2C |
|---|---|---|---|
| Complexity | Simple | Complex as device increases | Easy to chain multiple devices |
| Speed | Slowest(115200 hz) | Fastest(50hz) | Faster than UART(1mz) |
| No. of masters andslaves | Single to Single | 1 master, multiple slaves | Multipleslavesandmasters |
| Distance | 15m | qq cm | qq cm |
| Consommation | Moyenne | Faible | Élevée |
| Utilisation | Communication simple à a longue distance | Transferts rapides de donnée | Communication courte distance avec plusieurs périph |

## 4.7 - Fonctionnalités incomplètes

Le cahier des charges prévoyait également l'ajout d'une carte SD pour stocker les mélodies de réveil. Cette fonctionnalité n'a pas été réalisée, car la gestion de la carte SD n'est pas compatible avec FreeRTOS. Comme le système est destiné à évoluer vers une version temps réel basée sur FreeRTOS, cette partie a été mise de côté en attendant une solution plus adaptée.




# 5
# CONCLUSION

Ce projet de simulateur d'aube m'a permis de développer de compétences pratiques en programmation embarquée. J'ai appris à organiser un projet complexe en structurant le code proprement à travers des fichiers .c et .h, facilitant ainsi la lisibilité, la maintenance et l'évolutivité du programme. J'ai aussi compris l'importance d'éviter l'usage de fonctions bloquantes comme HAL_Delay() en utilisant les timers et les interruptions.

Sur le plan technique, ce projet m'a offert l'opportunité d'utiliser de nouveaux périphériques comme les capteurs ToF pour la détection de distance, les interfaces tactiles via le TSC, ainsi que le bus CAN pour établir une communication entre deux

cartes électroniques. J'ai appris à utiliser plusieurs modules ensemble (I2C, SPI, UART, PWM, ADC, timers, OLED, capteurs), en assurant la bonne gestion du code.

Enfin, pour les perspectives futures, je compte porter ce projet sous FreeRTOS, afin de bénéficier d'un système temps réel. Je prévois également d'introduire un algorithme d'IA légère permettant d'analyser et sauvegarder les mouvements du dormeur au fil du temps, dans le but d'évaluer et améliorer la qualité du sommeil.

# 6 BIBLIOGRAPHIE

[1] Répertoire du travail du cours Systèmes à Microcontrôleurs 2 à l'école de Mines SAINT - ETIENNES pour l'utilisation des librairies.

[2] Support du cours Systèmes à Microcontrôleurs 2 de l'école.

[3] Support du cours Systèmes à Microcontrôleurs 2 de l'école.

[4] Github stm32 pour la configuration des interfaces tactile.