MINISTÈRE
DE L'ÉDUCATION
NATIONALE
Liberté
Égalité
Fraternité

EAE SIN 3

SESSION 2025

AGRÉGATION
CONCOURS EXTERNE

Section : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR

Option : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR ET INGÉNIERIE INFORMATIQUE

CONCEPTION PRÉLIMINAIRE D'UN SYSTÈME, D'UN PROCÉDÉ OU D'UNE ORGANISATION

Durée : 6 heures

Calculatrice autorisée selon les modalités de la circulaire du 17 juin 2021 publiée au BOEN du 29 juillet 2021.

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout autre matériel électronique est rigoureusement interdit.

Il appartient au candidat de vérifier qu'il a reçu un sujet complet et correspondant à l'épreuve à laquelle il se présente.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.
Le fait de rendre une copie blanche est éliminatoire

Tournez la page S.V.P.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours	Section/option	Epreuve	Matière
EAE	1417A	103	1268

Ce sujet de 41 pages est composé :

- d'un dossier questionnement de la page 2 à la page 24 ;
  - Présentation : de la page 2 à la page 2 ;
  - Partie 1 : de la page 3 à la page 12 ;
  - Partie 2 : de la page 13 à la page 16 ;
  - Partie 3 : de la page 17 à la page 21 ;
  - Partie 4 : de la page 22 à la page 24 ;
- des documents techniques à partir de la page 25.

Conseils aux candidats :

- les parties du questionnement sont indépendantes ;
- un parcours attentif de l'ensemble du document est conseillé avant de composer ;
- la présentation des programmes doit respecter les mots clés du langage cible ainsi que l'indentation des structures algorithmiques ;
- les réponses doivent être présentées avec clarté, rigueur et concision.

IMPRIMERIE NATIONALE – 25 0109 – D'après documents fournis

Spectacles de drones

Présentation du système

Contexte du sujet

La pyrotechnie de spectacle a une histoire millénaire, débutant en Chine et se diffusant en Europe via la route de la soie. Elle a toujours mélangé des considérations artistiques avec des savoir faire militaires et techniques. Plus récemment, l'introduction de flottes de drones synchronisés a entraîné des changements significatifs dans ce domaine, permettant la création de tableaux plus variés. Cette évolution apporte des avantages en matière d'esthétique et de sécurité, mais pose également de nouveaux défis, notamment en termes de synchronisation, de fiabilité, de dynamique et de logistique. La société *Étincelle* basée à Rambouillet dans les Yvelines (78), a créé en 2022 une division drones pour proposer aux municipalités et comités d'entreprises, des ballets aériens en lieu et place des traditionnels spectacles pyrotechniques.
Ce sujet s'appuie sur les développements de cette division drones et étudie la conception de plusieurs systèmes participant à la réussite des spectacles. La société *Étincelle* propose plusieurs types de flotte de drones selon les contextes :

- La gamme *Chambord* pour des spectacles lumineux en extérieur (1024 drones).
- La gamme *Versailles* adaptée pour les départs sur plans d'eau (flotte de 256 drones).
- La gamme *Grande arche* adaptée pour les environnements urbains nécessitant une grande précision dans les déplacements pour l'évitement des structures et bâtiments (flotte de 128 drones).

Composition technique d'un système pour un évènement

Un spectacle de drones synchronisés repose sur plusieurs sous-systèmes techniques :

- Un ensemble de drones, identiques ou non. Le nombre de drones dépend des besoins de la direction artistique et influe directement sur les solutions techniques retenues.
- Une base au sol permettant d'accueillir les drones avant la mission, d'assurer leur charge ainsi qu'une zone de retour (deux bases peuvent être nécessaires selon les missions).
- Une station de commande au sol permettant de gérer la préparation, le déroulement et l'exécution de la fin de mission.
- Une infrastructure de communication pour gérer la télémétrie, les situations d'urgences et la synchronisation des drones.

Le sujet traite de la commande, de la détection d'obstacles et des différents aspects de la télémétrie. Il est divisé en quatre parties.

- La première partie est consacrée à l'analyse de la commande des moteurs d'un quadricoptère à partir des consignes de pilotage et des capteurs du contrôleur embarqué.
- La deuxième partie s'intéresse à une gamme de drones équipés de détecteur d'obstacles 360° utilisant la technologie laser afin d'éviter les obstacles dans le cas d'une utilisation de l'essaim dans un environnement contraint.
- La troisième partie s'intéresse aux aspects embarqués du système de télémétrie.
- La quatrième partie traitera des aspects logiciels pour le support aux équipes de suivi et de maintenance.

Première partie

Analyse d'un drone quadricoptère

L'objectif de cette partie est d'analyser la constitution d'un drone et de sa commande.

Sous-partie 1.1 : Modélisation

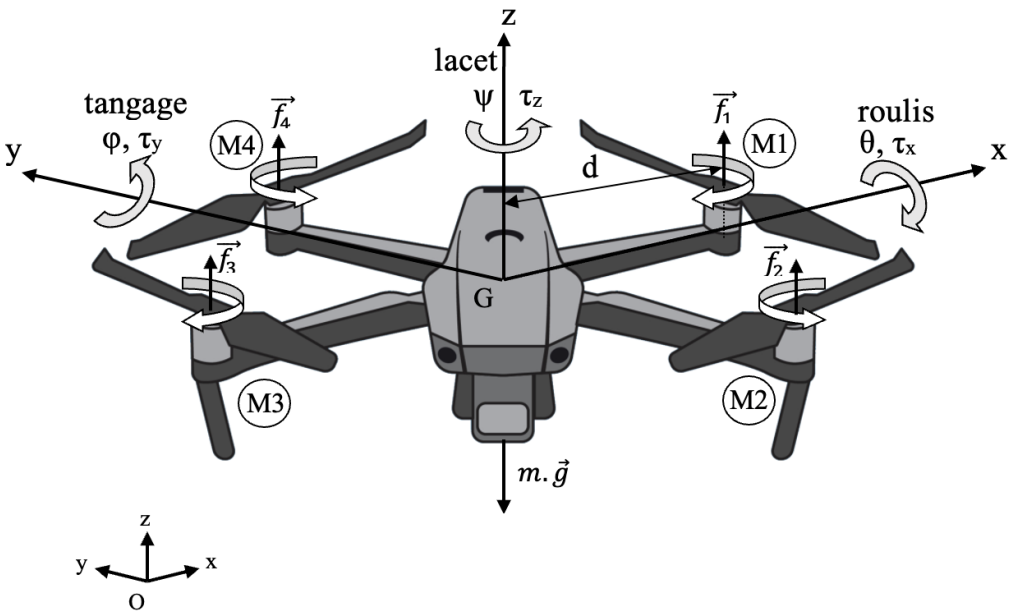


FIGURE 1 – Référentiels et conventions adoptées pour le drone

Les angles d'Euler sont définis comme suit :

- L'angle de roulis noté θ sur la figure 1 correspond à l'angle entre le référentiel fixe de centre O et du référentiel mobile de centre G (celui du drone) autour de l'axe x.
- L'angle de tangage noté φ sur la figure 1 correspond à l'angle entre le référentiel fixe de centre O et du référentiel mobile de centre G (celui du drone) autour de l'axe y.
- L'angle de lacet noté ψ sur la figure 1 correspond à l'angle entre le référentiel fixe de centre O et du référentiel mobile de centre G (celui du drone) autour de l'axe z.

La force de portance est perpendiculaire à l'écoulement de l'air, cette force est dirigée vers le haut afin d'élever le drone. La force de portance produite par la rotation de l'hélice i est donnée par la formule suivante :

$$f_i = b \times \omega_i^2$$

Où f_i est la force de portance produite par la rotation de l'hélice i, b le coefficient de portance en $\mathrm{N s^2}$, ω_i est la vitesse angulaire de l'hélice i en $\mathrm{rad\,s^{-1}}$.

La force de traînage : C'est la résultante des forces qui s'opposent au mouvement du quadrirotor dans l'air, de même direction que le mouvement du quadrirotor mais de sens opposé. Elle représente les forces de frottement visqueux sur l'objet.
Le couple aérodynamique de traînage T_i produite par une hélice s'exprime par la formule suivante :

$$T_i = k \times \omega_i^2$$

Où k est le coefficient de traînage en Nms^2, ω est la vitesse angulaire de l'hélice en rads^{-1}

La distance entre le centre d'inertie G du drone et l'axe des moteurs est notée d.

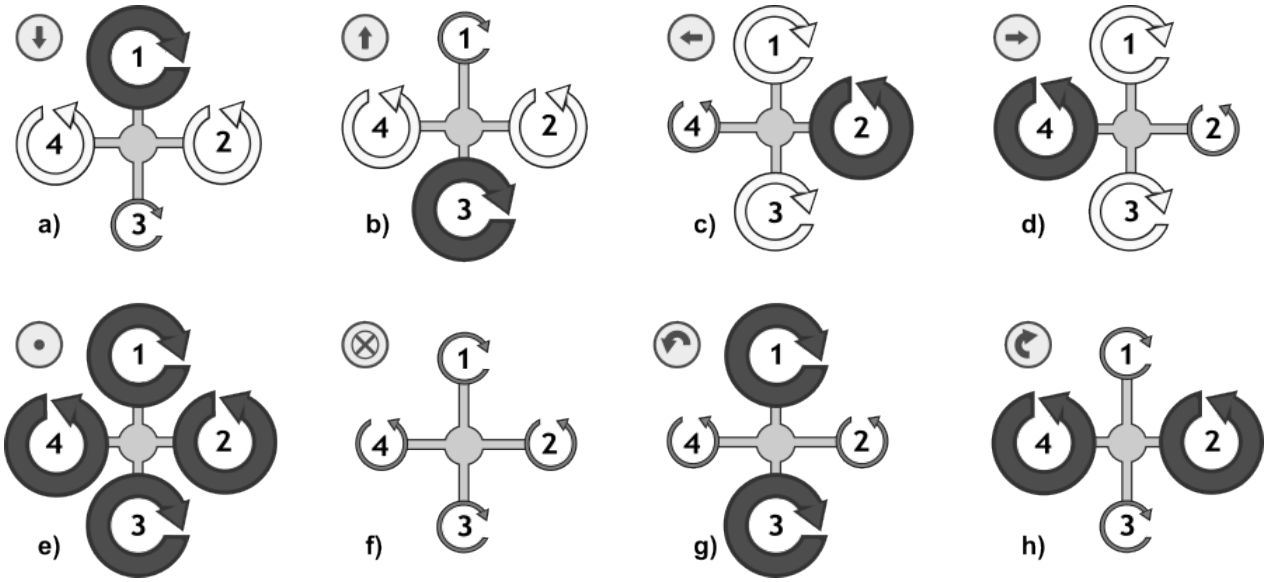


FIGURE 2 – Pilotage des moteurs et actions sur le drone

Q 1 *Déterminer l'expression des moments aérodynamiques notés T_x, T_y, T_z autour des axes x, y, z respectivement roulis, tangage et lacet à partir des informations des figures 1 et 2.*

Lors d'un décollage vertical et après avoir franchi le seuil définissant l'effet du sol, le drone peut rester en vol stationnaire à une hauteur constante par rapport au sol en ayant une vitesse de translation nulle. La force de poussée résultante des 4 moteurs doit alors équilibrer le poids du drone.

Q 2 *Déterminer la force de poussée notée F_p en fonction des vitesses angulaires que le drone doit produire pour compenser la force gravitationnelle dans le cas d'un vol stationnaire.*

Avec $\widetilde{u}_\theta, \widetilde{u}_\varphi, \widetilde{u}_\psi, \widetilde{u}_a$ les variations des entrées de commande du système et $\widetilde{u}_1, \widetilde{u}_2, \widetilde{u}_3, \widetilde{u}_4$ les variations de vitesse des moteurs (on posera $\widetilde{u}_i = \widetilde{\omega_i}^2$) :

$$\begin{cases} T_x = d \times b \times \widetilde{u}_\theta \\ T_y = d \times b \times \widetilde{u}_\varphi \\ T_z = k \times \widetilde{u}_\psi \\ F_p = b \times \widetilde{u}_a \end{cases} \tag{1}$$

Q 3 *Établir la matrice P respectant la relation suivante :*

$$\begin{pmatrix} \widetilde{u}_\theta \\ \widetilde{u}_\varphi \\ \widetilde{u}_\psi \\ \widetilde{u}_a \end{pmatrix} = P \times \begin{pmatrix} \widetilde{u}_1 \\ \widetilde{u}_2 \\ \widetilde{u}_3 \\ \widetilde{u}_4 \end{pmatrix} \tag{2}$$

Tournez la page S.V.P.

Q 4 *Sachant que les variations des entrées de commande sont les erreurs amplifiées des correcteurs (voir figure 3), montrer que la commande des moteurs répond au système d'équations suivant :*

$$\begin{cases} \widetilde{u}_1 = -\dfrac{1}{2} \times \widetilde{u}_\varphi + \dfrac{1}{4} \times \widetilde{u}_\psi + \dfrac{1}{4} \times \widetilde{u}_a \\ \widetilde{u}_2 = -\dfrac{1}{2} \times \widetilde{u}_\theta - \dfrac{1}{4} \times \widetilde{u}_\psi + \dfrac{1}{4} \times \widetilde{u}_a \\ \widetilde{u}_3 = \dfrac{1}{2} \times \widetilde{u}_\varphi + \dfrac{1}{4} \times \widetilde{u}_\psi + \dfrac{1}{4} \times \widetilde{u}_a \\ \widetilde{u}_4 = \dfrac{1}{2} \times \widetilde{u}_\theta - \dfrac{1}{4} \times \widetilde{u}_\psi + \dfrac{1}{4} \times \widetilde{u}_a \end{cases} \tag{3}$$

Sous-partie 1.2 : Détermination des paramètres pour la simulation du drone sous Matlab

Lors d'un essai expérimental, on pose le drone sur une balance puis on mesure sa masse à l'arrêt. Ensuite, on démarre le drone sans le faire décoller et on mesure sa masse et la vitesse de rotation des hélices grâce à un stroboscope (appareil de mesure de vitesse angulaire).
L'essai a donné les résultats suivant :

- Masse initiale du drone mesurée par la balance : $632\,\mathrm{g}$
- Masse du drone mesurée pour une vitesse de rotation des hélices de $1100\,\mathrm{tr/min}$: $612\,\mathrm{g}$

Q 5 *Proposer un protocole de mesurage de la force de poussée F_p ainsi que la détermination du paramètre b.*

Dans un deuxième essai, on démonte 2 hélices opposées puis on suspend le drone à une corde. Pour une vitesse de rotation des hélices de $1100\,\mathrm{tr/min}$, on mesure une force de traînée notée F_t de $29{,}43\,\mathrm{mN}$.

Q 6 *En déduire le coefficient de trainée k*

Sous-partie 1.3 : Fonctionnement du contrôleur de vol

Comme le quadrirotor est un système non linéaire et très instable, son pilotage et sa commande stabilisante nécessitent l'implémentation d'algorithmes qui demandent une grande capacité de calcul dans un temps minimal. Une carte électronique à base de microcontrôleur peut assurer cette contrainte.
La carte doit pouvoir traiter ces informations, exécuter ces lois stabilisantes et générer des signaux MLI (Modulation de largeur d'impulsions) pour commander les vitesses des moteurs. Le processeur reçoit les lectures sur l'orientation et l'altitude des capteurs à travers d'une communication I^2C. Enfin le processeur envoie aux contrôleurs électroniques de vitesse les quatre signaux PWM nécessaires pour contrôler les quatre moteurs qui sert à faire voler et contrôler le quadrirotor.
Le calculateur du drone détermine, à l'aide de ses capteurs de position et d'orientation ainsi que de la consigne de position GPS à atteindre, les consignes de vitesse moteur à appliquer afin de déplacer le drone dans la direction, l'altitude et l'orientation voulues.
On donne en figure 3 le schéma de principe de contrôle de ces grandeurs mis en place dans le calculateur du drone.

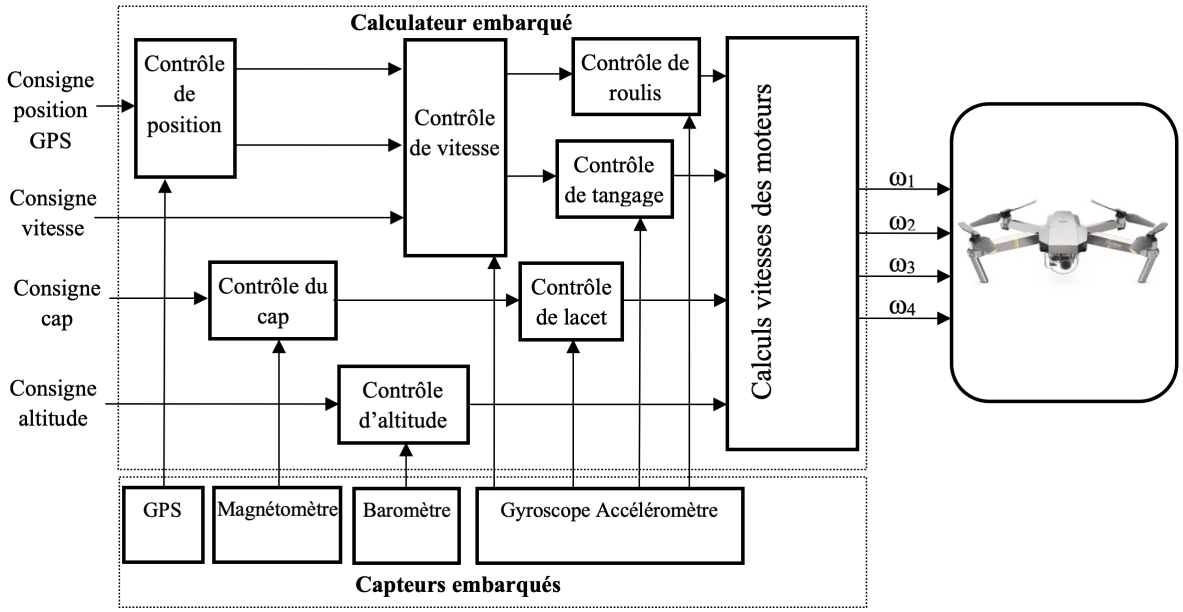


FIGURE 3 – Diagramme fonctionnel calculateur embarqué

Sous-partie 1.4 : Variateurs de vitesse électroniques des moteurs

La variation de vitesse de chaque hélice est réalisée grâce à des contrôleurs électroniques de vitesse (ESC figure 4) insérés dans chaque bras du drone. Ce contrôleur de vitesse pilote un moteur triphasé sans balais (Brushless). La vitesse est imposée par un calculateur en appliquant, en entrée du contrôleur, un signal à rapport cyclique variable (PWM).
Une batterie de 2 à 6 cellules LIPO (Lithium Polymère) fournit l'énergie aux contrôleurs de vitesse.

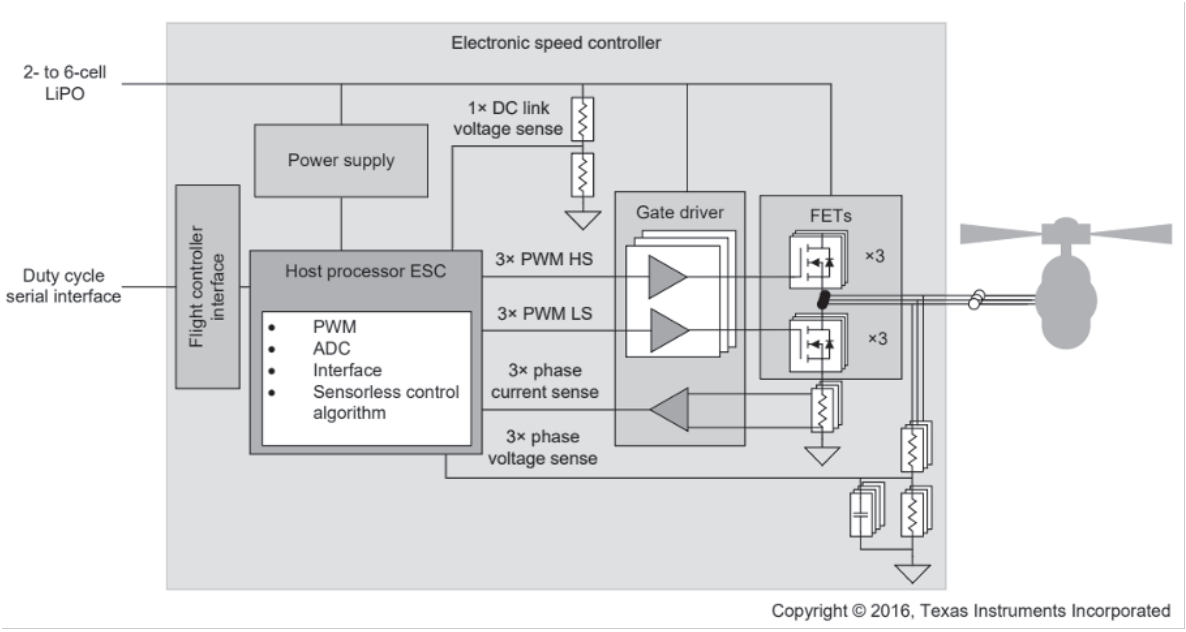


FIGURE 4 – Diagramme fonctionnel d'un Electronic Speed Controller (ESC)

La période de la PWM est constante. La documentation de l'ESC (non fournie) indique les plages de temps d'impulsions pour obtenir les différentes vitesses moteur.

Tournez la page S.V.P.

- Sens trigonométrique 0 à 100% : 1100 µs à 1500 µs
- Bande morte ou vitesse nulle : 1500 µs à 1540 µs
- Sens horaire 0 à 100% : 1540 µs à 1940 µs

Le diagramme de classe complet est donné dans le document technique DT1. Un diagramme de classes partiel concernant les classes `Motor` et `PWM` est donné à la figure 5.

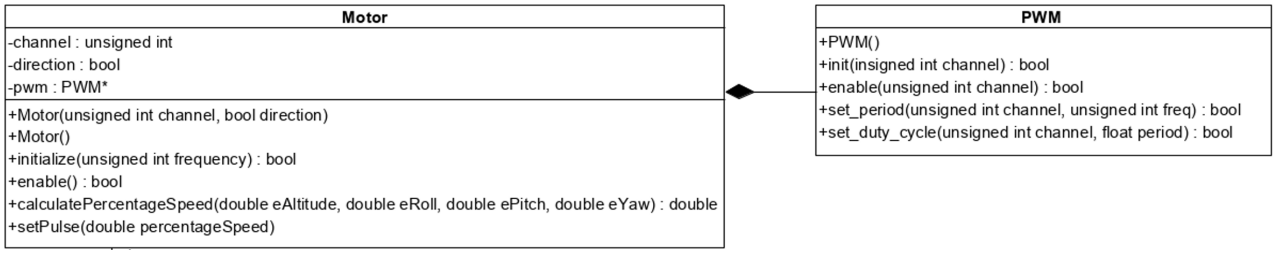


FIGURE 5 – Diagramme de classes partiel

Q 7 *D'après le diagramme de classes fourni à la figure 5, proposer une implémentation en langage C++ de la méthode* `setPulse()` *en tenant compte du sens de rotation du moteur. Remarque : la méthode* `set_duty_cycle()` *de la classe* `PWM` *prend en argument une variable* `period` *dont l'unité est la milliseconde.*

Pour déterminer l'orientation dans le repère géographique, l'altitude et la vitesse du drone, le calculateur récupère ces informations grâce au composant MPU9255 intégrant un accéléromètre 3 axes, un gyromètre 3 axes, un magnétomètre 3 axes. Ces capteurs utilisent un bus I^2C pour échanger les données avec le calculateur.

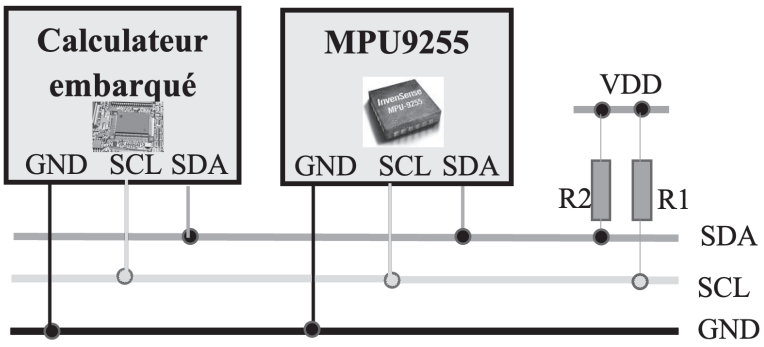


FIGURE 6 – Communication sur le bus I^2C

- Le calculateur embarqué utilise un système d'exploitation linux connecté au bus I^2C
- Le MPU9255 contient un gyromètre 3 axes, un accéléromètre 3 axes et un magnétomètre 3 axes

On se propose d'étudier le bus I^2C puis les échanges de données sur celui-ci.
Le calculateur embarqué émet une trame (figure 7) à destination du MPU9255 :

Q 8 *Citer le composant qui génère le signal d'horloge. Préciser le composant qui prend l'initiative de la communication sur le bus I^2C.*

Q 9 *Citer le deuxième bus de communication qui aurait pu être utilisé par ce capteur en indiquant les spécificités de chacun.*

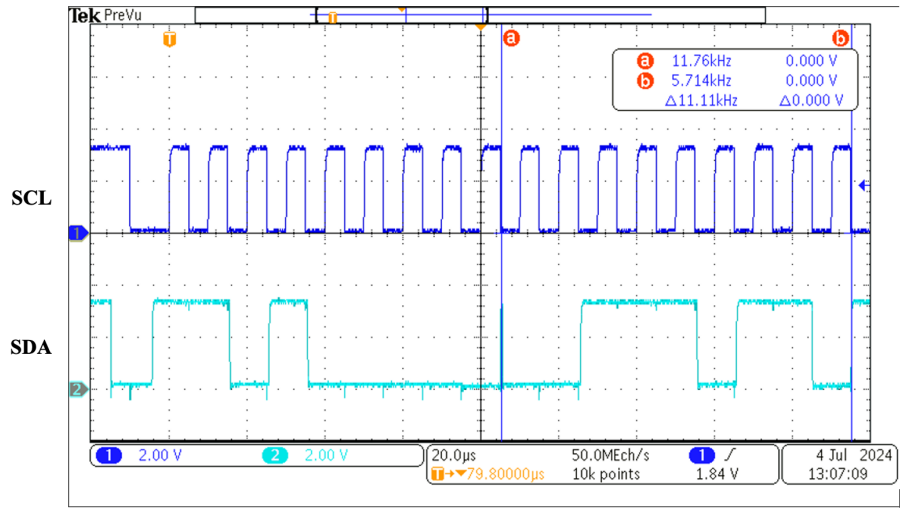


FIGURE 7 – Trame relevée sur le bus I^2C

Q 10 *En ayant au préalable identifié tous les bits* `START` *et* `ACK` *sur le chronogramme de le figure 7, en déduire l'adresse I^2C du MPU9255 et l'adresse du registre de départ concerné par la requête, en hexadécimal. S'agit-il d'une requête de lecture ou d'écriture ? A quoi correspond cette adresse de registre ?*

On capture la suite de 15 octets consécutifs aux 2 octets reçus dans la trame de la figure 7 :
68 FF 31 01 13 08 6F 03 C0 FF E8 00 09 00 02

Q 11 *A partir de la trame fournie (sachant que le bit $R/\overline{W}$ est positionné à 1) et de la documentation technique DT2, indiquer les valeurs d'accélérations et d'orientations sur les 3 axes sachant qu'elles peuvent être positives ou négatives.*

On donne le diagramme partiel de la classe MPU9255 et de la classe I^2C à la figure 8

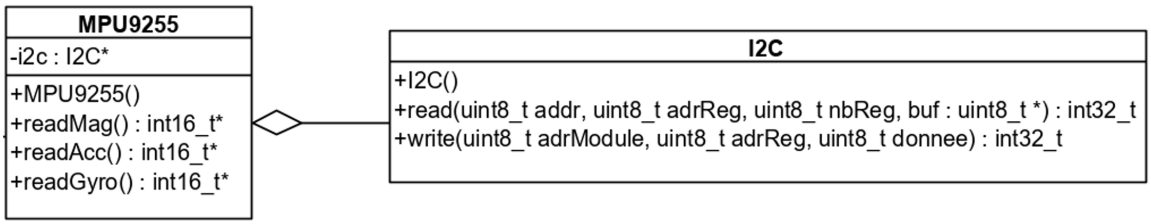


FIGURE 8 – Diagramme de classes partiel du capteur MPU9255

Q 12 *Écrire la méthode* `readGyro()` *de la classe MPU9255 en langage C++ permettant de lire les informations du gyroscope et qui renvoi les valeurs sur les 3 axes x, y et z. L'adresse de l'esclave I^2C MPU9255 se trouve dans la variable définie* `ADR_MPU9255`.

Sous-partie 1.5 : Synthèse du correcteur PID de l'angle de lacet

L'équation du correcteur PID parallèle présenté à la figure 9 est :

Tournez la page S.V.P.

$$\tilde{u}_\psi = K_{p\psi}\varepsilon_\psi(t) + K_{i\psi}\int_0^t \varepsilon_\psi(\tau)d\tau + K_{d\psi}\frac{d\varepsilon_\psi(t)}{dt} \quad (4)$$

Le correcteur PID étant programmé dans le calculateur embarqué, on note T_e la période d'échantillonnage des grandeurs asservies.

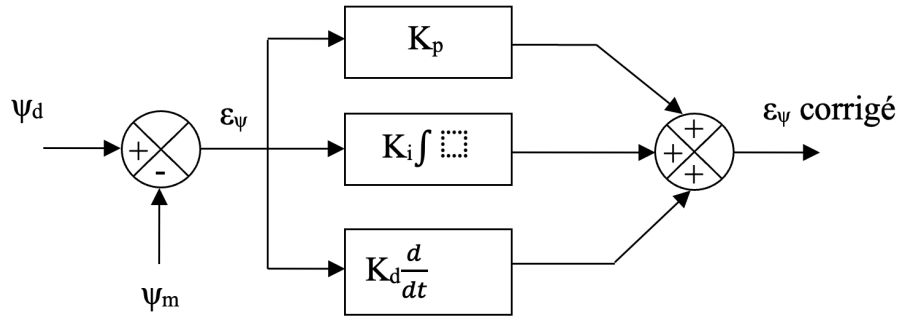


FIGURE 9 – Diagramme du correcteur PID parallèle

Q 13 *A partir de l'équation du correcteur, établir l'équation en temps discret de la variation de l'erreur de l'angle de lacet notée* $\tilde{u}_{\psi_n}$.

Q 14 *Si l'erreur* ε_{ψ_n} *est trop grande, que faut-il prévoir en sortie du correcteur ?*

On désire maintenant programmer le correcteur à l'aide d'une classe PID dont le diagramme de classe est fourni à la figure 10.

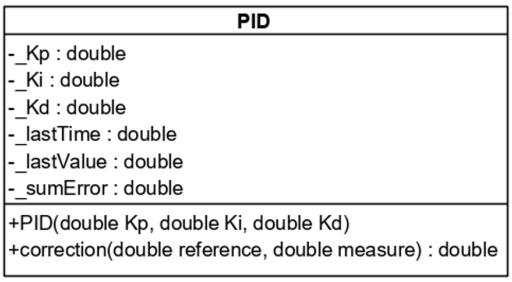


FIGURE 10 – Diagramme de la classe PID

Q 15 *Proposer une implémentation en langage C++ du constructeur pour initialiser les attributs privés de la classe PID. On dispose d'une fonction* `getTimeMs()` *qui renvoie le nombre de millisecondes depuis le 01/01/1970.*

Q 16 *Élaborer le programme de la méthode* `correction()` *de la classe PID à partir des résultats des questions 13 et 14*

Pour déterminer les paramètres des correcteurs PID du drone, on utilise la méthode Ziglers-Nichols. Cette méthode utilise la génération de l'oscillation entretenue en boucle fermée pour caractériser un système asservi.
Méthodologie :

- On annule l'action intégrale et l'action dérivée.
- L'action proportionnelle est augmentée jusqu'à ce que le signal en sortie de la boucle fermée oscille de manière entretenue. On note alors ce gain K_u correspondant au gain maximal (ou gain critique).
- On note T_u la période d'oscillation du signal. Les paramètres K_p, K_i et K_d du régulateur, sont choisis en se référant au tableau de la figure 11.

Type de contrôle	K_p	K_i	K_d
P	$0,5K_u$		
PI	$0,45K_u$	$\frac{0,54K_u}{T_u}$	
PD	$0,8K_u$		$\frac{K_u T_u}{10}$
PID	$0,6K_u$	$\frac{1,2K_u}{T_u}$	$\frac{3K_u T_u}{40}$

(5)

FIGURE 11 – Tableau méthode Ziglers-Nichols

On donne le schéma de simulation du drone avec le logiciel Matlab Simulink à la figure 12 ainsi que le résultat de simulation de l'asservissement d'altitude du drone étudié à la limite d'oscillation pour un gain critique $K_u = 3$ à la figure 13.

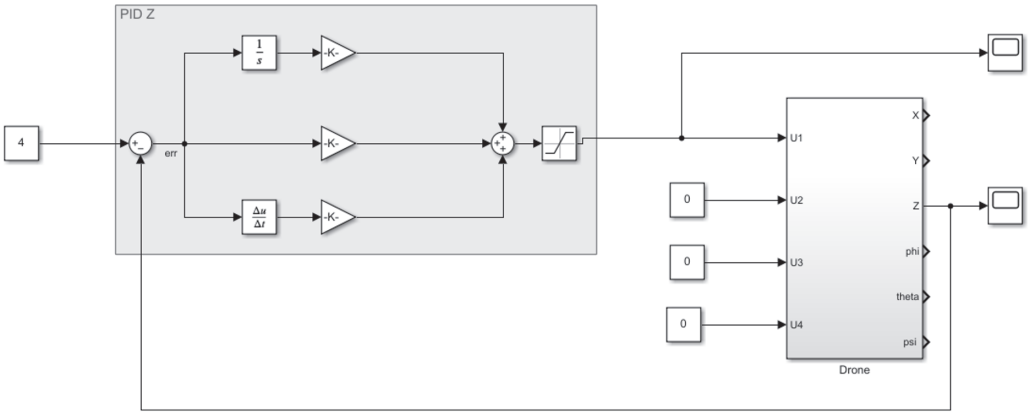


FIGURE 12 – Schéma de simulation de l'asservissement du drone en altitude

Q 17 *Déterminer les paramètres du correcteur PID d'altitude à l'aide la méthode Ziglers-Nichols et du relevé de la figure 13.*

Q 18 *Rappeler l'intérêt des actions de correction proportionnelles, intégrales et dérivés dans un asservissement.*

Le réglage par la méthode Ziglers-Nichols autorise un dépassement élevé comme le montre le relevé de gauche de la figure 14.

Q 19 *Proposer une solution afin de diminuer ce dépassement pour se rapprocher du relevé de droite de la figure 14.*

Tournez la page S.V.P.

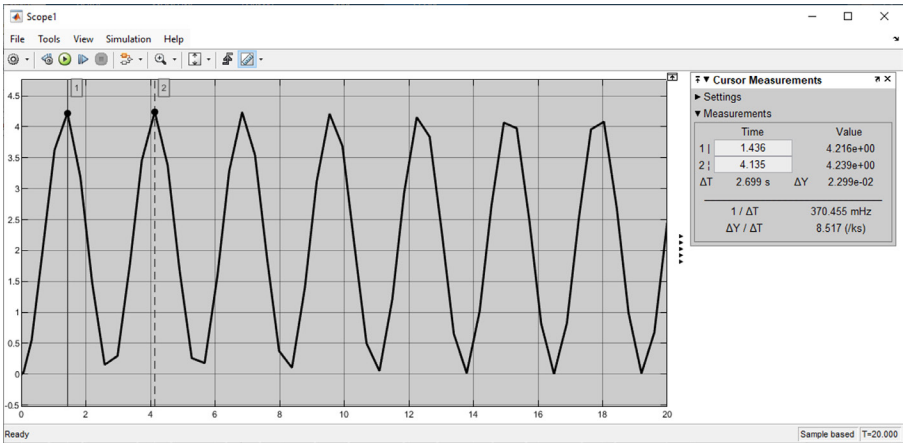


FIGURE 13 – Relevé asservissement d'altitude pour un gain critique $K_u = 3$

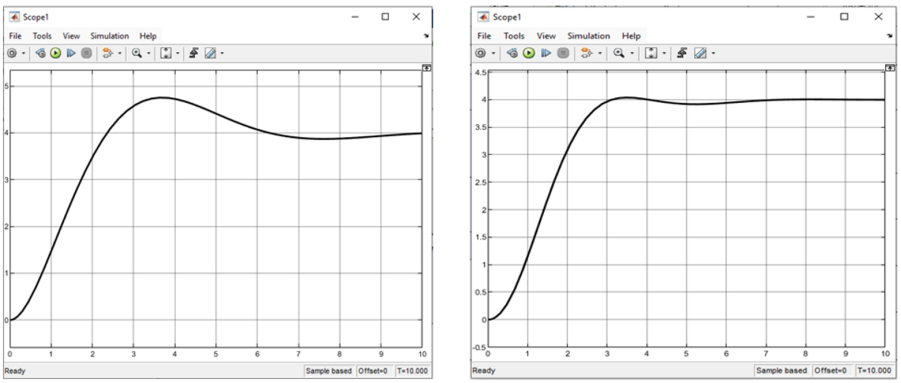


FIGURE 14 – Relevés lors des essais suite un échelon de d'altitude de 4 m

Sous-partie 1.6 : Logique de commande du drone

> **Q 20** *A partir des résultats de la question 4 et du diagramme de classes fourni à la figure 5, compléter (sur votre copie) les zones A et B de programme suivant afin de piloter les moteurs du drone.*

```
void* motors(void* arg)
{
    // instanciation des objets de la classe Motor
    Motor m1 = new Motor(1,true);
    Motor m2 = new Motor(2,false);
    Motor m3 = new Motor(3,true);
    Motor m4 = new Motor(4,false);

    // instanciation des objets de la classe PID à partir
    // des paramètres PID identifié par la méthode Zieglers- Nichols.
    // Roll: Tangage
    // Pitch: Roulis
    // Yaw: Lacet

    PID pidRoll = new PID(kpr, kir, kdr);
    PID pidPitch = new PID(kpp, kip, kdp);
```

```
    PID pidYaw = new PID(kpy, kiy, kdy);
    PID pidAltitude = new PID(kpa, kia, kda);

    // calculs des PID
    outPIDRoll = pidRoll->correction(refRoll, measRoll);
    outPIDPitch = pidPitch->correction (refPitch, measPitch);
    outPIDYaw = pidYaw->correction (refYaw, measYaw);
    outPIDAltitude = pidAltitude->correction (refAltitude, measAltitude);

    //  calculs des vitesses de chaque moteur  (zone A à compléter sur votre copie)
    speedM1 =
    speedM2 =
    speedM3 =
    speedM4 =

    //  commande en vitesse de chaque moteur (zone B à compléter sur votre copie)

}
```

Afin de récupérer rapidement les informations provenant des capteurs, on créé deux processus parallèles afin de traiter la récupération des informations des capteurs et un processus chargé de corriger les erreurs entre les consignes et les mesures fournies par les capteurs afin de piloter les 4 moteurs du drone.

Q 21 *Indiquer les variables qui vont appartenir à la zone mémoire critique.*

Q 22 *Expliquer succinctement le mécanisme à programmer afin de pouvoir écrire et lire ces variables.*

Tournez la page S.V.P.

Deuxième partie

Détection d'obstacles par LIDAR 360 pour la gamme *Grande arche*

Les drones de la gamme *Grande arche* sont conçus pour éviter les structures et bâtiments dans les environnements urbains. Ils sont dotés d'un système de détection d'obstacles basé sur un LIDAR N10.
Le système de mesures réalise un balayage à 360 degrés afin d'obtenir continuellement les informations d'angle grâce à la rotation d'un moteur. Pour chaque mesure, le LIDAR émet un rayon laser infrarouge qui est réfléchi vers le récepteur lorsqu'il détecte un obstacle. On obtient l'instant d'émission du rayon laser et l'instant de réception du rayon réfléchi sur l'obstacle. La distance à l'obstacle peut être calculée grâce à cette différence de temps et la vitesse de la lumière. Une fois les données de distance obtenues, le LIDAR fusionne les valeurs d'angle mesurées par l'unité de mesure d'angle pour former les données en nuage de points, puis envoie les données du nuage de points via une communication série de type USB.

Q 23 *A partir de la documentation du LIDAR et sachant que le temps de réponse à un échelon de consigne de l'asservissement de lacet et de le temps de traitement numérique de la déviation de l'obstacle est inférieur à* $1{,}5\,\mathrm{s}$ *et que le seuil de détection du LIDAR est fixé à* $8\,\mathrm{m}$*, déterminer la vitesse maximum de déplacement du drone.*

Q 24 *Quelle est l'erreur de mesure de distance entre le début et la fin de la rotation du LIDAR pour la vitesse de déplacement maximum du drone trouvée précédemment pour une trajectoire rectiligne ? Quelle serait-elle pour une vitesse de déplacement de 60 km/h dans les mêmes conditions ? Exprimer ces valeurs en pourcentage du seuil de détection du LIDAR. Quel dispositif pourrait-on mettre en place pour augmenter la vitesse maximum de déplacement du drone sur les données récoltées par le LIDAR ?*

On donne la trame suivante envoyé par le LIDAR.
A5 5A 3A 10 43 19 1B 03 FD A8 04 0D AC 04 2C B3 04 3B A7 04 5B C1 04 79 C2 04 98 B6 04 B7 B2 04 D6 B6 05 04 AB 05 14 A2 05 23 AA 05 23 A9 05 23 B5 05 23 B3 05 22 B3 1D CB 28

Q 25 *A partir de la trame ci-dessus et de la documentation du LIDAR, donner les caractéristiques de cette trame ainsi que les valeurs des 3 premières mesures de distances ainsi que l'angle correspondant. Indiquer le nombre de mesures effectué par le LIDAR entre l'angle de départ et l'angle de fin.*

On se propose de programmer la récupération de trames dans un tableau d'octets (attribut `trame`) puis les informations d'angles, de distances et de l'intensité lumineuse dans un buffer circulaire d'entiers de taille 512×3 (attribut `dataLidar`).
On donne le contenu du fichier de déclaration de la classe `Lidar360`.

```
#include <pthread.h>
#include <mutex>
#include <math.h>

class Lidar360
{
    ComSerie *com;
    unsigned char *trame;
    uint32_t ** dataLidar;
    mutex mx;
```

```
public:
    Lidar360(string port);
    void lireTrame();
    void stockerDataLidar();
    bool checksum();
    uint32_t[] rechercheObstacle(int indiceMin, int indiceMax);
    float deviationCap(uint32_t seuil, float largeurDrone);
};
```

On donne un extrait du fichier de déclaration de la classe `ComSerie` :

```
#define NO     0
#define ODD    1
#define EVEN   2

public:
    ComSerie();
    ComSerie(const std::string portCom, unsigned int vitesse, unsigned int parite);
    ~ComSerie();
```

Q 26 *Proposer une implémentation du constructeur de la classe* `Lidar360` *à partir des informations de la classe* `ComSerie`*.*

Q 27 *Proposer une implémentation en langage C++ du destructeur de la classe* `Lidar360`*.*

On désire capter les trames émises par le LIDAR provenant de la liaison série. Dans un premier temps, on doit détecter l'entête des trames. Pour cela, on donne le prototype de la méthode `recevoir()` de la classe `ComSerie`.

```
int recevoir(unsigned char * trame, unsigned char nbOctets);
```

Q 28 *Compléter la méthode* `lireTrame()` *de la classe* `Lidar360` *afin de remplir la variable trame.*

Q 29 *D'après les données récupérées dans la trame à la question 25, combien de valeurs au maximum doit-on stocker dans le tableau* `dataLidar` *?*

Q 30 *Programmer la méthode* `stockerDataLidar()` *de la classe* `Lidar360` *pour remplir le tableau* `dataLidar` *en fonction de l'angle de départ et de fin et des valeurs de distances et d'intensités lumineuses. Les valeurs de distances seront stockées en millimètres et les angles en centièmes de degré.*

L'accès à la variable `dataLidar` exige la mise en place d'une exclusion mutuelle car cette variable est à la fois en écriture et en lecture. L'attribut `mx` de type `Mutex` de la classe `Lidar360` possède deux méthodes :

- une méthode `lock()` qui permet de verrouiller l'accès à la zone mémoire critique,
- une méthode `unlock()` qui permet de déverrouiller l'accès à la zone mémoire critique.

Tournez la page S.V.P.

Q 31 *Préciser dans votre codage précédent où vous devez insérer les instructions pour effectuer l'exclusion mutuelle.*

Suite à la détection d'un obstacle, la stratégie adoptée est de modifier le cap du drone afin de le diriger temporairement vers une zone ou il n'y a pas d'obstacles. Le LIDAR est positionné sur le drone de façon à ce que le 0° du LIDAR correspondent à une marche avant. La recherche d'obstacles se fera pour des angles compris en 0° et 48° (coté droit du drone) puis entre 312° et 360° (côté gauche du drone).

Q 32 *Quelles informations doit-on rechercher avec les données du LIDAR afin de détecter un obstacle ou plusieurs obstacles ?*

Q 33 *Sur le diagramme fonctionnel du calculateur embarqué figure 3, sur quel bloc doit-on agir afin de contourner l'obstacle ?*

On désire rechercher les valeurs minimum de distance et d'angle à l'obstacle pour un indice évoluant de l'indice minimum à l'indice maximum dans le tableau `dataLidar`.

Q 34 *Proposer le codage de la méthode* `rechercheObstacle(int indiceMin, int indiceMax)` *qui renvoie respectivement un tableau avec la distance minimale à l'obstacle ainsi que la valeur de l'angle correspondant entre les deux indices du tableau pris en argument de la méthode. Quelle est la complexité de cet algorithme ?*

Q 35 *Préciser dans votre codage précédent où vous devez insérer les instructions pour effectuer l'exclusion mutuelle.*

Dans le tableau `dataLidar`, la résolution de mesure est de 75 centièmes de degré. On se propose dans cette question d'élaborer un programme simplifié permettant au drone de dévier pour éviter l'obstacle. Pour cela, le programme va prendre en compte tous les cas de figure que le drone peut rencontrer sur un plan à deux dimensions. La déviation verticale ne sera pas étudiée dans cette partie.

Q 36 *Déterminer les indices du tableau pour une recherche de distance pour un angle compris en 0 et 48° et pour un angle compris entre 312° et 360°.*

Q 37 *Dans la suite de la programmation, on désire vérifier si le drone peut passer entre 2 obstacles détectés. Déterminer la relation à intégrer au programme permettant de déterminer l'angle $\delta\theta$ entre 2 obstacles en fonction de la distance d et de la largeur l_d du drone en intégrant une largeur de passage correspondant à 2 fois sa largeur. La figure 15 illustre schématiquement la problématique de la question.*

Pour éviter l'obstacle, la commande du drone doit connaître l'angle de déviation par rapport à la direction de la destination finale.

- Si la valeur de distance à l'obstacle est inférieure à un seuil de distance pris en argument de la méthode en mètres sur l'un des deux secteurs d'angles avant droit ou gauche, la méthode renverra un angle de déviation 40 fois l'inverse de la distance à l'obstacle.
- Si deux obstacles sont détectés sur les secteurs avant droit et gauche :
  - — Si le drone peut passer alors on renvoie l'angle médian entre les 2 obstacles,
  - — Si le drone ne peut pas passer alors on déviera le drone du coté où la distance à l'obstacle est la plus grande en ajoutant l'angle de déviation correspondant à deux fois la largeur du drone.
- S'il n'y a pas d'obstacle, la méthode renverra 0.

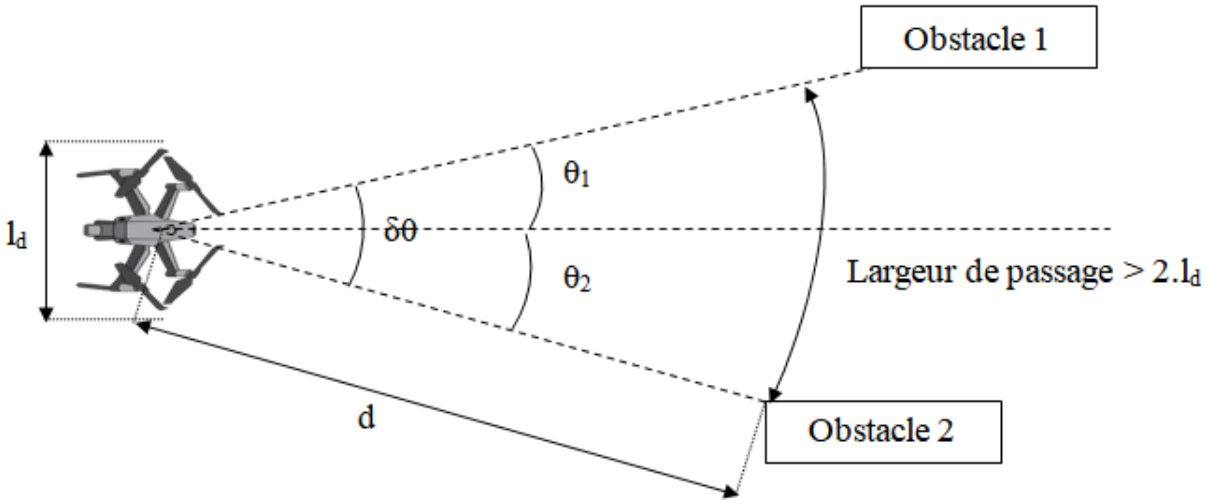


FIGURE 15 – Passage du drone entre 2 obstacles

Q 38 *Proposer le codage de la méthode* `deviationCap()` *qui prend en argument le seuil de distance de détection d'un obstacle et la largeur du drone à l'aide des méthodes disponibles dans le fichier de déclaration de la classe* `Lidar360` *fourni pour la question 26. Cette méthode renverra l'angle de déviation en degré. On utilisera la constante* `M_PI` *de la bibliothèque math.h pour la valeur* π.

Tournez la page S.V.P.

Troisième partie

Télémetrie

Plus le nombre de drones et de spectacles augmente, plus la probabilité d'une défaillance lors d'un spectacle est élevée. Cette partie se concentre sur le système de télémétrie qui a la charge de suivre en temps réel l'état de santé et la position de chaque drone afin d'agir de manière préventive mais aussi de permettre de récupérer les drones défaillants après une prestation et de réaliser une analyse *post-mortem* pour l'amélioration continue des flottes de drones.
On distingue 4 cas qui sont traités de manière différente.

- Le drone présente un fonctionnement nominal et dispose d'une charge suffisante pour la mission.
- Le drone présente un défaut ne lui permettant pas de participer à l'évènement,
- Le drone présente un défaut pendant l'évènement nécessitant une action (extinction des lumières, évacuation du dispositif),
- Le drone présente un défaut ne lui permettant pas de rejoindre sa base au sol à l'issue de l'évènement et nécessitant alors une procédure de localisation et de récupération.

Sous-partie 3.1 : Spécifications

La solution retenue par les concepteurs, est de réaliser un surveillance en temps réel des drones en mettant en place un système de télémétrie. Chaque drone est équipé d'une carte dédiée à la télémétrie pouvant fonctionner indépendamment des circuits de contrôle. Cette carte mesure et récupère des informations des différents capteurs, les analyse et transmet des messages à intervalles réguliers. La transmission des informations à la station sol repose sur un réseau zigbee déployé au moyen de modules XBee. La station sol enregistre et distribue les messages via un courtier de messages MQTT (Message Queuing Telemetry Transport). Le microcontrôleur retenu est un STM32F411.
Les données transmises sont :

- Un identifiant unique pour identifier le drone,
- le niveau de charge de la batterie,
- la position GPS complète,
- un code sur un octet représentant l'état du drone (cf table 1).

Nous considérerons une batterie de type 3S - Tension nominale 11,1 V - Tension à pleine charge 12,5 V - 3000 mAh.

Code hexadécimal	Signification
0x00	Hors charge sur la base
0x01	En charge sur la base
0x02	Mission en cours
0x03	Fin de mission
0x04	Tension de batterie inférieure au seuil medium
0x08	Tension de batterie inférieure au seuil bas
0x10	Lumière défectueuse
0x20	Pas de position GPS mesurée
0x80	Sortie de la zone de sécurité

TABLE 1 – Liste des codes d'état

Sous-partie 3.2 : Mise en forme des données

Coordonnées de localisation

Le drone est équipé d'un module GNSS Quectel L86 permettant d'opérer sur les constellations Galileo, GPS et GLONASS. Un enregistrement des trames NMEA reçues par le drone lors d'une campagne de test est donné en figure 16.

```
1 $GPRMC,173523.000,A,4838.4880,N,00148.8110,E,0.00,307.09,230524,,,A,V*1C
2 $GPGGA,173523.000,4838.4880,N,00148.8110,E,1,7,1.0,199.6,M,0.0,M,,*5A
3 $GNGSA,A,3,19,17,06,09,,,,,,,,,2.16,1.92,0.99,1*0F
4 $GNGSA,A,3,87,71,72,,,,,,,,,,2.16,1.92,0.99,2*01
5 $GNGLL,4838.49,N,00148.81,E,173523.000,A,A*48
```

FIGURE 16 – Extrait d'un relevé GNSS au format NMEA lors d'une campagne de test

Q 39 *En vous aidant du document technique DT4, identifier toutes les trames NMEA de cet extrait contenant la position du module GNSS. Vous indiquerez pour chacune : les numéros de lignes correspondantes, l'identifiant de 5 lettres en début de trame ainsi que la constellation utilisée.*

Q 40 *Indiquer quelle trame permet un positionnement complet du drone. Justifier. Extraire les informations de position de cette trame.*

Q 41 *Combien de satellites sont en vue au moment de la campagne de test ? A quelles constellations appartiennent ces satellites ? Le cas échéant, Préciser en fonction du satellite.*

La position ainsi récupérée est utilisée d'une part pour s'assurer que le drone est bien dans le périmètre prévu de l'évènement et d'autre part pour le retrouver en cas de défaut technique. La précision de la position est alors plus importante que la réactivité de la mesure pour les évènements en champ libre. La valeur récupérée est convertie en `double` (flottant 64bits) pour les calculs puis convertie en `float` (flottant 32bits) pour le stockage et l'envoi à la station au sol.

Q 42 *Décrire le format d'un nombre en virgule flottante (IEEE754).*

Q 43 *Justifier l'utilisation des `double` pour les calculs. Vous préciserez les phénomènes possibles selon les calculs.*

Dans un scénario de perte d'un appareil, la position du drone recherché est considérée fixe et il est envisagé de filtrer les valeurs pour éliminer les positions aberrantes. La fréquence des messages issus du module Quectel sera de 1Hz. On ne se préoccupera pas de la configuration de cette fréquence dans cette étude.

Q 44 *Proposer un algorithme efficace de moyenne mobile sur un horizon de* $10\,\mathrm{s}$ *pour améliorer la précision du positionnement. La minimisation du nombre d'opérations de calcul sera évaluée.*

Dans les spectacles en milieu urbain, le nombre de valeurs aberrantes issues du système de localisation augmente et les concepteurs ont constaté que la moyenne n'était pas suffisante pour compenser l'effet de ces valeurs. La solution retenue est de calculer la médiane sur les 7 dernières valeurs.

Tournez la page S.V.P.

Q 45 *Proposer un algorithme pour déterminer cette médiane indépendamment pour chaque coordonnée.*

Les informations obtenues sont destinées uniquement à la partie télémétrie et ne permettrait pas un contrôle en temps réel du drone.

Q 46 *Justifier cela et indiquer quelles données complémentaires seraient nécessaires pour le contrôle actif de la position. Quel module, usuellement utilisé pour réaliser ce contrôle, peut fournir ces données complémentaires ?*

Niveau de tension de la batterie

Le convertisseur analogique numérique embarqué sur le microcontrôleur STM32 réalise l'acquisition de la tension de la batterie au travers du schéma du document technique DT5. Toutes les résistances sont de précision 1%.

Q 47 *Exprimez le rapport numérique entre la tension batterie `VBATT` et `VMes` la tension en entrée du convertisseur.*

La documentation du STM32 indique que le convertisseur utilise une technologie SAR 12Bits.

Q 48 *Expliquer le principe d'un convertisseur SAR. Quels sont ses avantages et inconvénients ? En quoi cette technologie est elle adaptée à la problématique ?*

La précision souhaitée sur la tension de batterie est de $\pm 10\,\mathrm{mV}$. La tension de référence utilisé par l'ADC est fixée à $V_{Ref} = 3{,}00\,\mathrm{V}$. Une approximation, suffisante dans le cadre de cette étude, du comportement du convertisseur est donnée en DT6.

Q 49 *La résolution du convertisseur est elle compatible avec cette contrainte ? Justifier.*

Le programme du microcontrôleur est conçu en langage C. La donnée récupérée via le convertisseur correspondant à `VMes` doit être mise à l'échelle et affectée à `vbat` pour refléter fidèlement la tension de batterie `VBATT`. Une ébauche de code de la fonction `mesure_tension_batterie()` est donné en figure 17.

```
1  uint16_t mesure_tension_batterie() {
2      uint16_t AD_RES = 0;
3      // Conversion
4      HAL_ADC_Start(&hadc1);
5      // Attente de fin de conversion
6      HAL_ADC_PollForConversion(&hadc1, 1);
7      // Lecture
8      AD_RES = HAL_ADC_GetValue(&hadc1);
9      // Mise a l'echelle
10     uint16_t vbat = a completer sur votre copie
11     return vbat;
12 }
```

FIGURE 17 – Code de la fonction de mesure de la tension batterie

Q 50 *En vous basant sur le document technique DT6 et le code précédent, écrire sur votre copie la ligne de code C complétée correspondant à la ligne 10 pour mettre à l'échelle la tension mesurée.*

Identifiant unique

Afin d'assurer une identification et un suivi des drones sur l'ensemble de leur cycle de vie, toutes les informations de télémétrie comportent un identifiant unique dans le message. Les microcontrôleurs STM32 disposent d'un identifiant unique `UID` de 96 bits consécutifs débutants à l'adresse mémoire `UID_BASE`. Ces 96 bits sont créés à partir des éléments suivants :

```
UID[31:0]: X and Y coordinates on the wafer expressed in BCD format
UID[63:40]: LOT_NUM[23:0] Lot number (ASCII encoded)
UID[39:32]: WAF_NUM[7:0] Wafer number (8-bit unsigned number)
UID[95:64]: LOT_NUM[55:24] Lot number (ASCII encoded)
```

Q 51 *Proposer un code en C pour récupérer l'id et le mettre sous la forme d'un tableau de 3 entiers 32 bits (`uint32_t`). Est-il possible de réduire la taille de l'UID en préservant son caractère unique ? Justifier.*

Code d'état

La table 1 page 17 répertorie les données d'états possibles. On remarquera que cette table a été construite pour permettre de cumuler dans un seul octet plusieurs données d'états différentes.

Q 52 *Justifier cette possibilité en représentant la signification des différentes parties de cet octet. Quelle est la valeur en hexadécimal, représentant l'état d'un drone en cours de mission, avec une tension batterie entre le seuil bas et le seuil medium, sans autre alerte ?*

Structure de données

Dans le code embarqué de la carte télémétrie, les données de localisation, de niveau de batterie, le code d'état ainsi que l'identifiant unique sont regroupés dans une structure de données (struct en C).

Q 53 *Donner le code de définition de cette structure.*

Q 54 *Sans tenir compte des questions d'alignement en mémoire, calculer l'espace mémoire en octets occupé par cette structure.*
Cette structure est elle sujette au remplissage (padding) pour respecter l'alignement en mémoire ? Justifier.

Sous-partie 3.3 : Réseau zigbee

Le standard zigbee publié et maintenu par *Connectivity Standards Alliance (CSA)*, anciennement *zigbee Alliance* propose un protocole de haut niveau basé sur le protocole de communication IEEE 802.15.4 visant particulièrement la communication d'équipements personnels ou domestiques équipés de petits émetteurs radios à faible consommation. Le document technique DT7, définit les éléments qui composent un réseau zigbee. Le document technique DT9 défini la trame `Transmit Request` qui sera utilisée par chaque drone pour transmettre ses données au coordinateur.

Tournez la page S.V.P.

Q 55 *En vous appuyant sur le document technique DT7, justifiez l'utilisation d'un réseau zigbee dans le cadre de notre application et proposer une topologie adaptée en explicitant quels éléments seraient utilisés comme coordinateur, routeurs, terminaux.*

La documentation DT8 donne la structure générale d'une trame zigbee dans le mode API.

Q 56 *Donnez la valeur hexadécimale des trois premiers octets d'une trame transportant 23 octets de données. Justifier.*

Les données collectées par le microcontrôleur STM32 (les données de localisation, de niveau de batterie, le code d'état ainsi que l'identifiant unique), sont assemblées dans une trame de la forme suivante : UID puis Position GPS puis niveau de batterie puis code d'état.

Q 57 *En vous appuyant sur le document technique DT8, calculer N le nombre d'octets qui composent le message complet (API Frame) permettant de transmettre ces données via une requête "Transmit Request Frame" ? Justifier.*

Comme indiqué dans le DT8, le message se termine par un checksum. Dans le code du microcontrôleur, le message à transmettre est construit et stocké dans un tableau d'octets (char) de taille N. (`char Mes[N]`). `Mes` contient l'ensemble du message y compris l'entête « Start Delimiter » et le checksum.

Q 58 *Proposer un algorithme qui calcule le checksum et met* `Mes` *à jour avec sa valeur.*

Q 59 *Proposer un programme qui construit le message complet y compris le checksum. Vous pourrez vous baser sur la structure de données de la question 53.*

Avant l'introduction d'un compteur de trames, les réseaux zigbee étaient réputés sensibles aux attaques par rejeu (aussi nommées attaques par relecture).

Q 60 *Définir une attaque par rejeu.*

Q 61 *Sans tenir compte des mécanismes de protection des réseaux zigbee récents, quelles seraient les conséquences d'une attaque par rejeu sur notre système ? Y-a-t-il un risque de perturber un évènement ?*

Quatrième partie

Suite logicielle

Plusieurs logiciels accompagnent l'équipe en charge de planifier et d'exécuter les évènements ainsi que d'assurer la maintenance des drones. L'équipe est composée de plusieurs personnes disposant toutes d'un suivi d'informations sur un ordinateur portable, une tablette ou un téléphone, soit en exécutant un programme python soit en accédant aux données via une interface web.

Sous-partie 4.1 : Organisation générale

Comme vu dans la partie précédente, les drones communiquent leurs informations au module zigbee coordinateur. Celui-ci connectée par liaison série à une carte raspberry Pi 5 qui assume, entre autres missions, le rôle de concentrateur dans le schéma de la figure 18. Cette carte est reliée à un réseau wifi déployé localement accessible par internet via un routeur 5G. Le courtier MQTT et le réseau sont configurés pour permettre un accès authentifié depuis internet.
Cette carte héberge :

- Un courtier MQTT,
- un programme python qui scrute la liaison série et publie les informations sur le courtier dans les topics : `topic/batterie`, `topic/position` et `topic/etat`.

Les services clients du courtier sont hébergés sur d'autres machines, locales ou distantes. Dans le cadre de cette étude nous nous pencherons sur la création du service *Moniteur* qui utilisera une base de données SQLite et transmet ses alertes via le courtier MQTT.

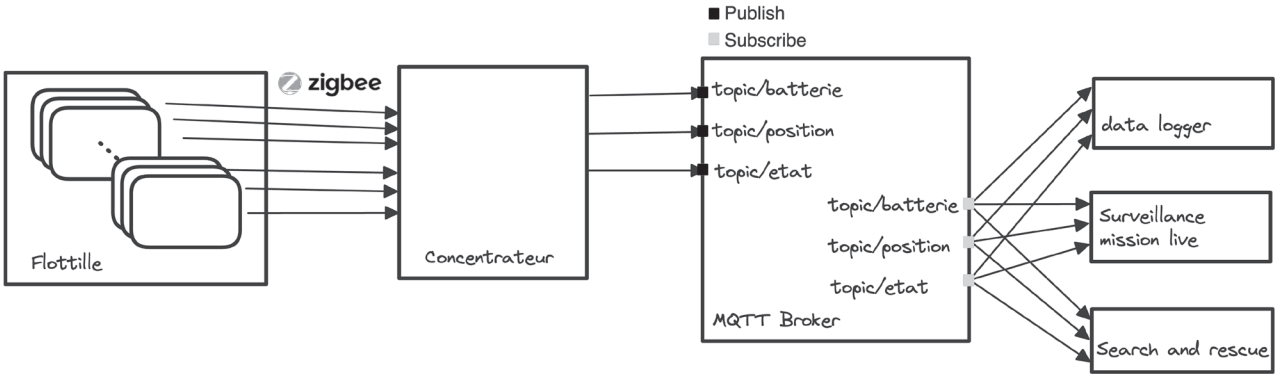


FIGURE 18 – Architecture du système de télémétrie

Courtier MQTT

MQTT, pour Message Queuing Telemetry Transport, est un protocole de messagerie léger et efficace conçu pour les applications de l'Internet des objets (IoT). Il est basé sur un modèle publish/subscribe, où les clients peuvent publier des messages sur des sujets (topics) auxquels d'autres clients peuvent s'abonner. Le courtier MQTT joue le rôle d'intermédiaire, en stockant et en distribuant les messages aux abonnés pertinents.
Les trois clients existants sont :

- `Data logger` : service distant qui enregistre tous les évènements pour l'analyse qualité à posteriori.
- `Surveillance mission live` : service utilisable localement ou à distance pour surveiller l'exécution des missions.

Tournez la page S.V.P.

- `Search and rescue` : service local qui utilise des services de cartographie pour permettre de publier sur une interface web la localisation et la récupération des drones accidentés ou hors zone.

Une des missions du programme `Search and rescue` est de détecter les drones qui ne transmettent plus de messages depuis un temps défini. Cette fonctionnalité est aussi nécessaire dans `Surveillance mission live`. Dans cette partie, nous allons externaliser cette fonctionnalité au service `Moniteur` pour pouvoir récupérer l'information depuis `Search and rescue`, `Surveillance mission live` et d'autres services en transitant par le courtier.

Sous-partie 4.2 : Conception du service `Moniteur`

L'architecture présentée a été pensée pour rendre évolutive la solution globale en permettant d'ajouter rapidement de nouveaux services sans déstabiliser les services existants. Nous allons dans cette sous partie ajouter `Moniteur` un service spécialisé avec des fonctionnalités dédiées à la recherche des drones perdus. Une fois le service crée et validé, ces fonctionnalités pourront être progressivement retirées de `Surveillance mission live` et de `Search and rescue`.

Moniteur de flotte

Après examen des besoins exprimés par les utilisateurs des services `Search and rescue` et `Surveillance mission live` ainsi que par l'équipe en charge du développement de ces services, il a été décidé de charger le moniteur de :

- Surveiller les mises à jour de position de chaque drone et d'ajouter dans une base de données : les nouveaux drones, l'instant du dernier contact et leur position. Le système n'aura pas besoin de connaître la liste des drones avant la mission.
- Mettre à jour dans la base de données, l'horodatage du dernier message à l'arrivée de chaque nouveau message.
- Inspecter la base de données pour trouver l'ensemble des drones dont l'horodatage est plus distant de l'instant présent qu'une valeur définie par avance (on considère alors le contact perdu).
- Publier l'ID du drone, l'instant de la perte de contact et la dernière position connue sur le topic `topic/perdus`.

Le service est déjà partiellement développé et les codes existants sont disponibles dans la documentation technique DT12.

Gestion du courtier mqtt

Q 62 *Analyser la classe* `EcouteMQTT` *et donner son diagramme de classe.*

Q 63 *Donner sur votre copie la ligne 9 de* `mqtt_monitor` *complétée pour se connecter et souscrire au topic* `topic/position` *du courtier accessible à l'adresse « b53z91.etincelle.fx » sur le port 1883.*

Le sous domaine « b53z91.etincelle.fx » a été choisi comme adresse de redirection pour le courtier pour tenter de rendre plus difficile la connexion malicieuse au courtier mqtt par rapport à « mqtt.etincelle.fx ».

Q 64 *Quel service est en charge de la redirection des sous-domaines dans un réseau ?*
Indiquer en quoi cette stratégie n'est pas efficace et proposer un mécanisme plus efficace contre les connexions malicieuses au courtier mqtt.

Gestion de la base de données

Q 65 *Analyser la classe* `SQLiteManager` *et donner son diagramme de classe.*

Q 66 *Expliquer en détail la requête* `create_rq`*. Vous justifierez les mots clefs utilisés, la structure de la table et les types utilisés. Donner le schéma de base de données de la table drones.*

Q 67 *Expliquer le fonctionnement de la méthode* `insert_or_update`*. Sur quel mécanisme/propriété s'appuie-t-elle pour sélectionner l'opération à effectuer.*

Moniteur

Dans le programme moniteur.py, on se connecte à la base SQLite, puis on calcule l'heure avant laquelle les messages sont considérés comme trop ancien, soit 15 s avant l'heure actuelle.

Q 68 *Compléter sur votre copie la requête de la ligne 24 pour que la variable* `hors_delai` *contienne la liste des drones hors délai avec les données requises par le sujet.*

Q 69 *Remplacer la ligne 41 par une commande provoquant l'émission d'un message mqtt sur le topic « topic/alerte » donnant (dans cet ordre et séparés par le caractère « : »), l'UID du drone et ses dernières latitude, longitude et altitude connues.*

Disparition de trames

A plusieurs reprises, les tests (non disponible dans ce sujet) ont fait état de perte de messages. Grâce au cadre de test le problème a été localisé dans l'interaction entre la méthode `on_message` de la classe `EcouteMQTT` et la ligne 31 de `mqtt_monitor.py` qui supprime tous les messages.

Q 70 *Expliquer ce phénomène.*

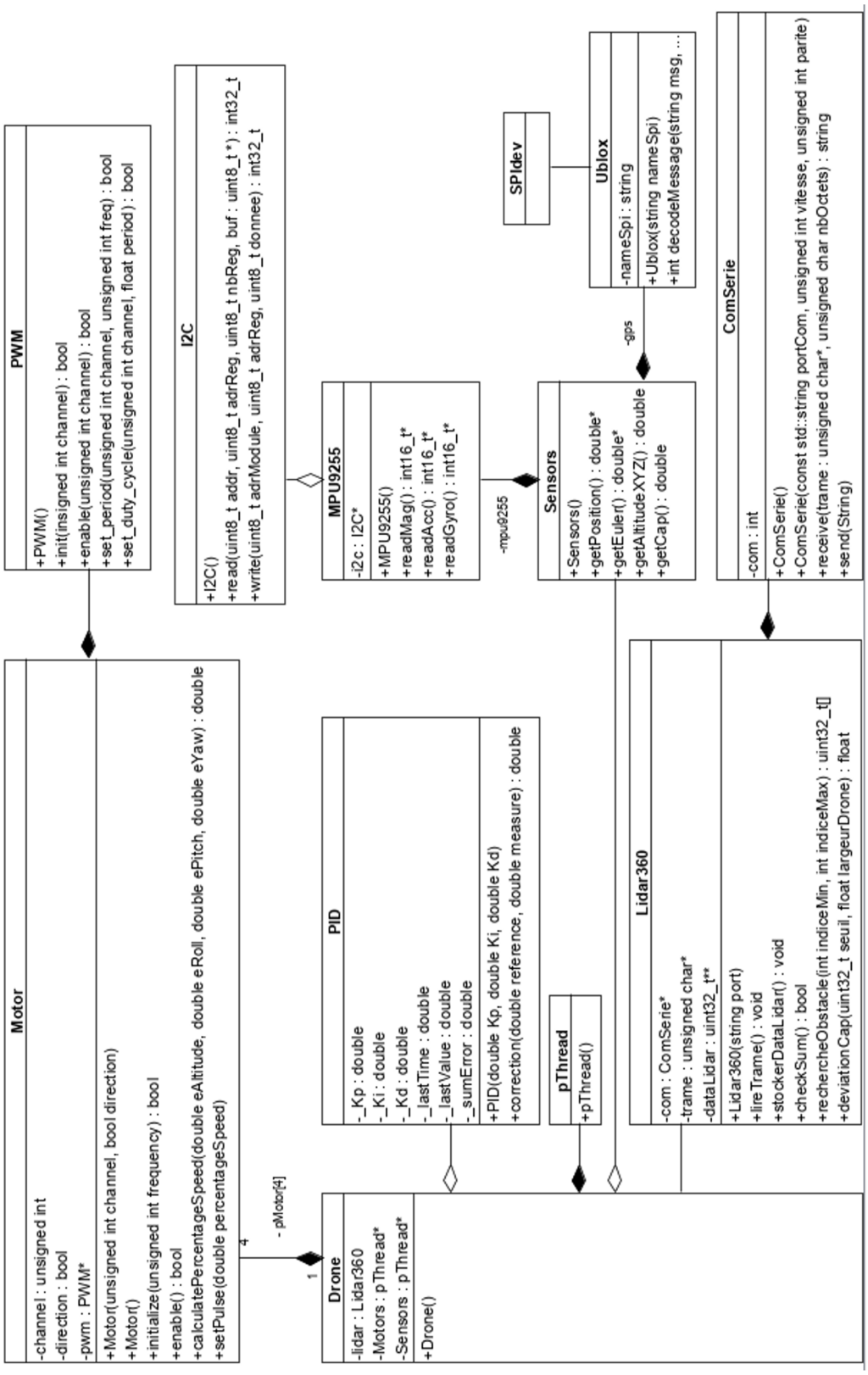
On décide d'implémenter un mécanisme simple de verrou pour résoudre ce problème. Pour commencer, la ligne `self.verrou = False` est ajoutée (décommentée) à l'initialisation de la classe `EcouteMQTT`.

Q 71 *Indiquer comment transformer les différents fichiers pour obtenir un mécanisme efficace basé sur cette variable. Vous donnerez les modifications complètes du code.*

Q 72 *Proposer une méthode plus avancée pour résoudre le phénomène de disparition des trames.*

Document techniques

DT1 Diagramme de classes complet du système



Tournez la page S.V.P.

C

DT2 Extrait du document MPU-9255

InvenSense	InvenSense Inc. 1745 Technology Drive, San Jose, CA 95110 U.S.A. Tel: +1 (408) 988-7339 Fax: +1 (408) 988-8104 Website: www.invensense.com	Document Number: DS-000007 Revision: 1.0 Release Date: 09/14/2014

Extrait du document MPU-9255

1 Product Overview

MPU-9255 is a multi-chip module (MCM) consisting of two dies integrated into a single QFN package. One die houses the 3-Axis gyroscope and the 3-Axis accelerometer. The other die houses the AK8963 3-Axis magnetometer from Asahi Kasei Microdevices Corporation. Hence, the MPU-9255 is a 9-axis MotionTracking device that combines a 3-axis gyroscope, 3 -axis accelerometer, 3-axis magnetometer and a Digital Motion Processor™ (DMP) all in a small 3x3x1mm package available as a pin-compatible upgrade from the MPU-6515. With its dedicated I^2C sensor bus, the MPU- 9255 directly provides complete 9-axis MotionFusion™ output. The MPU-9255 MotionTracking device, with its 9 -axis integration, on-chip MotionFusion™, and run-time calibration firmware, enables manufacturers to eliminate the costly and complex selection, qualification, and system level integration of discrete devices, guaranteeing optimal motion performance for consumers. MPU-9255 is also designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I^2C port.

MPU- 9255 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs, three 16-bit ADCs for digitizing the accelerometer outputs, and three 16-bit ADCs for digitizing the magnetometer outputs.

2 Features

2.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-9255 includes a wide range of features:

Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec and integrated 16-bit ADCs
Digitally-programmable low-pass filter

2.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-9255 includes a wide range of features:
Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g* and integrated 16-bit ADCs
Accelerometer normal operating current: 450µA
Low power accelerometer mode current: 8.4µA at 0.98Hz, 19.8µA at 31.25Hz
User-programmable interrupts

2.3 Magnetometer Features

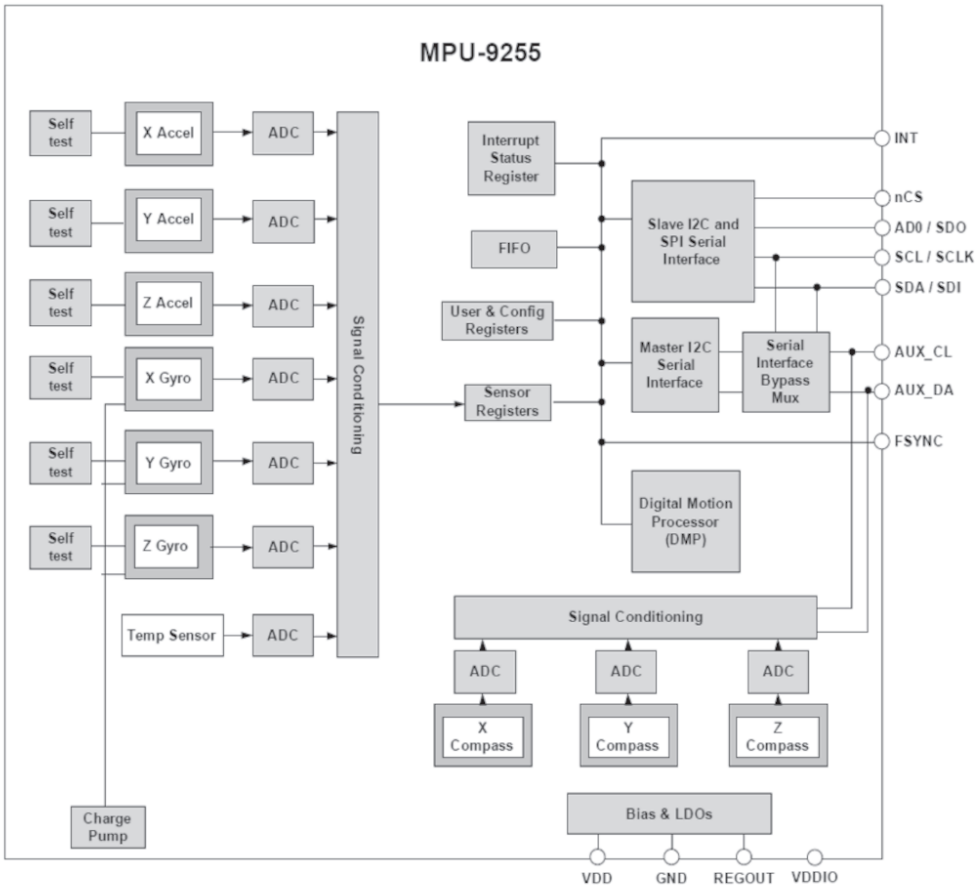
The triple-axis MEMS magnetometer in MPU-9255 includes a wide range of features:

3-axis silicon monolithic Hall-effect magnetic sensor with magnetic concentrator
Wide dynamic measurement range and high resolution with lower current consumption
Output data resolution of 14 bit (0.6µT/LSB) or 16 bit (15µT/LSB)
Full scale measurement range is ±4800µT
Magnetometer normal operating current: 280µA at 8Hz repetition rate
Self-test function with internal magnetic source to confirm magnetic sensor operation on end products

2.4 Additional Features

The MPU-9255 includes the following additional features:Auxiliary master I^2C bus for reading data from external sensors (e.g. pressure sensor) 3.5mA operating current when all 9 motion sensing axes and the DMP are enabled
VDD supply voltage range of 2.4 – 3.6V
VDDIO reference voltage for auxiliary I^2C devices
Smallest and thinnest QFN package for portable devices: 3x3x1mm
Minimal cross-axis sensitivity between the accelerometer, gyroscope and magnetometer axes 512 byte FIFO buffer enables the applications processor to read the data in bursts
Digital-output temperature sensor
User-programmable digital filters for gyroscope, accelerometer, and temp sensor 10,000 *g* shock tolerant
400kHz Fast Mode I^2C for communicating with all registers
1MHz SPI serial interface for communicating with all registers

2.5 Block Diagram



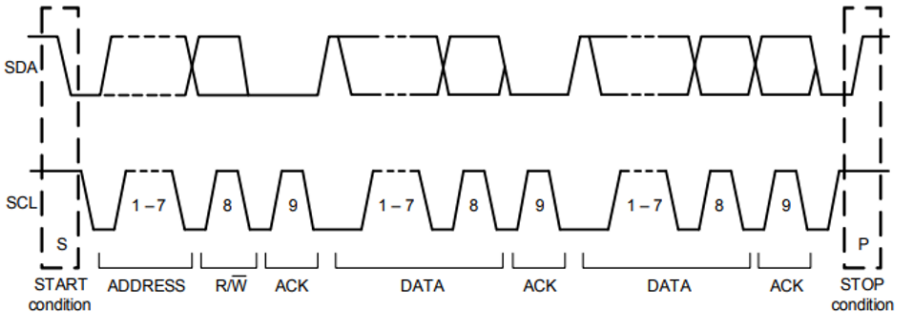
3 Register Map for Gyroscope and Accelerometer

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
35	53	I2C_SLV4_DI	R	I2C_SLV4_DI[7:0]							
36	54	I2C_MST_STATUS	R	PASS_THROUGH	I2C_SLV4_DONE	I2C_LOST_ARB	I2C_SLV4_NACK	I2C_SLV3_NACK	I2C_SLV2_NACK	I2C_SLV1_NACK	I2C_SLV0_NACK
37	55	INT_PIN_CFG	R/W	ACTL	OPEN	LATCH_INT_EN	INT_ANYRD_2CLEAR	ACTL_FSYNC	FSYNC_INT_MODE_EN	BYPASS_EN	-
38	56	INT_ENABLE	R/W	-	WOM_EN	-	FIFO_OFLOW_EN	FSYNC_INT_EN	-	-	RAW_RDY_EN
3A	58	INT_STATUS	R	-	WOM_INT	-	FIFO_OFLOW_INT	FSYNC_INT	-	-	RAW_DATA_RDY_INT
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT_H[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT_L[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT_H[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT_L[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT_H[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT_L[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT_H[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT_L[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT_H[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT_L[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT_H[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT_L[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT_H[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT_L[7:0]							

Tournez la page S.V.P.

4 Communication

After beginning communications with the START condition (S), the master sends a 7-bit slave address followed by an 8th bit, the read/write bit. The read/write bit indicates whether the master is receiving data from or is writing to the slave device. Then, the master releases the SDA line and waits for the acknowledge signal (ACK) from the slave device. Each byte transferred must be followed by an acknowledge bit. To acknowledge, the slave device pulls the SDA line LOW and keeps it LOW for the high period of the SCL line. Data transmission is always terminated by the master with a STOP condition (P), thus freeing the communications line. However, the master can generate a repeated START condition (Sr), and address another slave without first generating a STOP condition (P). A LOW to HIGH transition on the SDA line while SCL is HIGH defines the stop condition. All SDA changes should take place when SCL is low, with the exception of start and stop conditions.



Complete I²C Data Transfer

To read the internal MPU-9255 registers, the master sends a start condition, followed by the I²C address and a write bit, and then the register address that is going to be read. Upon receiving the ACK signal from the MPU-9255, the master transmits a start signal followed by the slave address and read bit. As a result, the MPU-9255 sends an ACK signal and the data. The communication ends with a not acknowledge (NACK) signal and a stop bit from master. The NACK condition is defined such that the SDA line remains high at the 9th clock cycle. The following figures show single and two-byte read sequences.

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

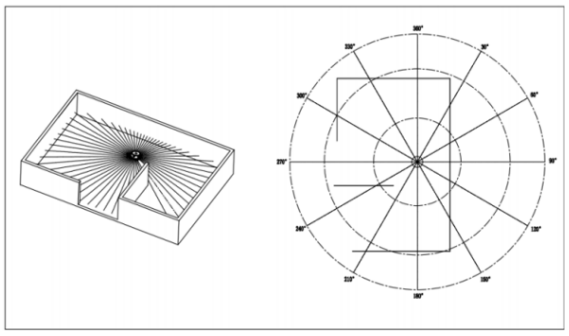
7.4 I2C Terms

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I²C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9th clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9th clock cycle
RA	MPU-9255 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

DT3 Extrait de la documentation du LIDAR N10

1. Product description

The Lidar N10 consists mainly of a laser ranging core, a wireless power transmission unit, a wireless communication unit, an angle measuring unit, a motor drive unit and a mechanical housing. With DTOF technology, Lidar N10 ranging core is able to perform 4,800 measurements per second. For each ranging, the LiDAR emits an infrared laser, which is reflected back to the single photon receiving unit when it encounters the target object. From this, we obtain the time at which the laser is emitted and that at which it is received by the single photon receiving unit. The time difference between them is the time of flight of the light, which can be combined with the speed of light to solve for the distance. Once the distance data have been obtained, Lidar N10 fuses the angle values measured by the angle measuring unit to form the point cloud data and then sends the point cloud data of 58 bytes to an external interface via wireless communication. Meanwhile the external interface supports PWM input to enable the motor drive unit to drive the motor rotation. The external control unit obtains the speed and controls it to the specified speed by means of a PID algorithm in closed-loop control, this allowing the LiDAR to work stably. The diagram of the environmental scan formed by Lidar N10 point cloud data is shown below:



The product is mainly suitable for the navigation and obstacle avoidance of robots (e.g. floor mopping robots and service robots) by performing a 360° scan of the indoor layout and building a map so that a walking path can be planned. It is also suitable for robotics education and research, etc.

The Lidar N10 has a motor driver with stepless speed regulation, which supports internal speed control and external speed control. When the PWM pin is grounded, the default is internal speed regulation, and the default speed is 10±0.1Hz.

2 Serial Port Configuration

Baud rate	Data length	Stop bit	Parity check bit	Flow control
230400	8Bits	1	N/A	N/A

3 Point Cloud Output Protocol

Byte_0	**Byte_1**	**Byte_2**	**Byte_3**	**Byte_4**
A5	5A	Length	Speed_H	Speed_L
Byte_5	**Byte_6**	**Byte_7**	**Byte_8**	**Byte_9**
Start_angle_H	Start_angle_L	Distance_1_H	Distance_1_L	PEAK_1
Byte_10	**Byte_11**	**Byte_12**	**…**	**…**
Distance_2_H ……	Distance_2_L	PEAK_2	…	…
Byte_55	**Byte_56**	**Byte_57**		
Stop_angle_H	Stop_angle_L	Checksum		

Note: L represents low data address; H represents high data address.

1) Byte_0 - Byte_1: frame header, fixed.

2) Byte_2: the length of the data frame, from the frame header to checkbit.

3) Byte_3 - Byte_4: motor speed, in μs, the most significant value in the sequence is stored first, at the lowest storage address while the least significant value is stored at the highest storage address.
For example: Speed_H=0x10, Speed_L=0x46, i.e. 0x1046 => 4166 μs. Time for one revolution of the code disk: 4166 μs*24 = 100 ms, i.e. speed 10 Hz

4) Byte_5- Byte_6: the starting angle of a frame of data, with the high bit in front. It is 100 times of the actual angle. Example: Start_angle_H=0x42, Start_angle_L=0x08, i.e. 0x4208 => 16904 => 169.04°.

5) Byte_7- Byte_54: Point cloud data. Each data includes 2 bytes of distance and 1 byte of intensity information, with the high bit first and the distance in mm For example: Distance_1_H=0x00, Distance_1_L=0X64, PEAK_1=0x64, meaning distance: 0x64=100mm, intensity: 100.

6) Byte_55- Byte_56: End angle. Calculation method is the same as start angle. 7) Byte_57: data and checksum from Byte_0 to Byte_56. CKS = byte0 + byte1 +...+ byte56

Tournez la page S.V.P.

DT4 Structures des trames NMEA

Les trames NMEA sont codées au format ASCII et sont de la forme :

`$<talker ID><Trame type>[,<données>,<données>]*<checksum>`

Champs	Longueur	Signification
$	1	Marqueur de début de trame
Talker ID	2	Équipement ayant généré la trame NMEA
Trame type	3	Code identifiant le contenu de la trame
Données	variable	Charge utile dont le contenu est défini par le «Trame type». Chaque valeur est séparée par le caractère «,»
*	1	Séparateur de checksum
Checksum	2	Somme de contrôle générée par un ou exclusif de tous les caractères situés entre '$' et '*' (exclus)
Fin de ligne	2	Caractères «carriage return» + «line feed» marquant un retour à la ligne (<CR><LF> soit <0x0D><0x0A>)

Exemple : `$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,,0000*0E`

Talker ID

Le type d'équipement à l'origine du signal (talker id) est défini par les deux caractères qui suivent le $. Les principaux préfixes sont :

- BD ou GB - Beidou ;
- GA - Galileo ;
- GP - GPS ;
- GL - GLONASS.

Le préfixe GN est utilisé dans le cas de signaux mixés GPS + GLONASS.

Type de trames

La longueur maximale d'une trame est 82 octets (en incluant les caractères de fin de ligne)
Il existe plus d'une trentaine de trames NMEA différentes. Chaque trame a sa syntaxe propre mais selon le cas, elles peuvent ou doivent se terminer, après le caractère '*', par une somme de contrôle. Ce mécanisme permet de vérifier que la trame n'a pas été altérée lors de sa transmission.
Un récepteur GPS renvoie souvent plusieurs types de trames complémentaires (les GGA et RMC en sont un exemple) car tous les logiciels qui interprètent le NMEA ne connaissent pas toutes les trames. De même de nombreux GPS transmettent des trames non standardisées propres à leur fabricant (d'habitude ces trames propriétaires ne commencent pas par $GP. Par exemple, «GL» réservé aux GLONASS).
Les trames NMEA font toutes référence à l'ellipsoïde WGS84 comme base de son système de coordonnées.

Quelques exemples de types de trames :

La trame GGA (Global Positioning System Fix Data).

```
$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,,0000*0E

$GPGGA       : Type de trame
064036.289   : Trame envoyée à 06 h 40 min 36 s 289 (heure UTC)
4836.5375,N  : Latitude exprimée en ddmm.mmmm: 48° 36.5375' Nord
00740.9373,E : Longitude exprimée en dddmm.mmmm: 7° 40,9373' Est
1            : Type de positionnement (le 1 est un positionnement GPS)
04           : Nombre de satellites utilisés pour calculer les coordonnées
3.2          : Précision horizontale ou HDOP (Horizontal dilution of precision)
```

```
200.2,M      : Altitude 200,2, en mètres
,,,,,0000    : D'autres informations peuvent être inscrites dans ces champs
*            : séparateur de checksum
0E           : Somme de contrôle de parité, un simple XOR sur les caractères entre $ et *4
```

La trame RMC (Recommended Minimum Navigation Information)

```
$GPRMC,053740.000,A,2503.6319,N,12136.0099,E,2.69,79.65,100106,,,A*53

$GPRMC       : type de trame
053740.000   : heure UTC exprimée en hhmmss.sss : 5 h 37 min 40 s
A            : état A=données valides, V=données invalides
2503.6319    : latitude exprimée en ddmm.mmmm: 25° 03.6319'
N            : indicateur de latitude N=nord, S=sud
12136.0099   : longitude exprimée en dddmm.mmmm: 121° 36.0099'
E            : indicateur de longitude E=est, W=ouest
2.69         : vitesse sur le fond en nœuds (2,69 nd = 3,10 mph = 4,98 km/h)
79.65        : route sur le fond en degrés
100106       : date exprimée en jjmmaa : 10 janvier 2006
,            : déclinaison magnétique en degrés (souvent vide pour un GPS)
,            : sens de la déclinaison E=est, W=ouest (souvent vide pour un GPS)
A            : mode de positionnement A=autonome, D=DGPS, E=DR
*            : séparateur de checksum
53           : somme de contrôle de parité au format hexadécimal
```

La trame GLL - Position Géographique - Longitude / Latitude

```
$GPGLL,4916.45,N,12311.12,W,225444,A*31

$GPGLL    : type de trame
4916.45   : latitude exprimée en ddmm.mm :Latitude 49° 16.45'
N         : indicateur de latitude N=nord, S=sud
12311.12  : longitude exprimée en dddmm.mm 123° 11.12'
W         : indicateur de longitude E=est, W=ouest
225444    : acquisition du Fix à 22:54:44 UTC
A         : données valides
*         : séparateur de checksum
31        : somme de contrôle de parité au format hexadécimal
```

La trame GSA - Satellites actifs - DOP dilution de précision

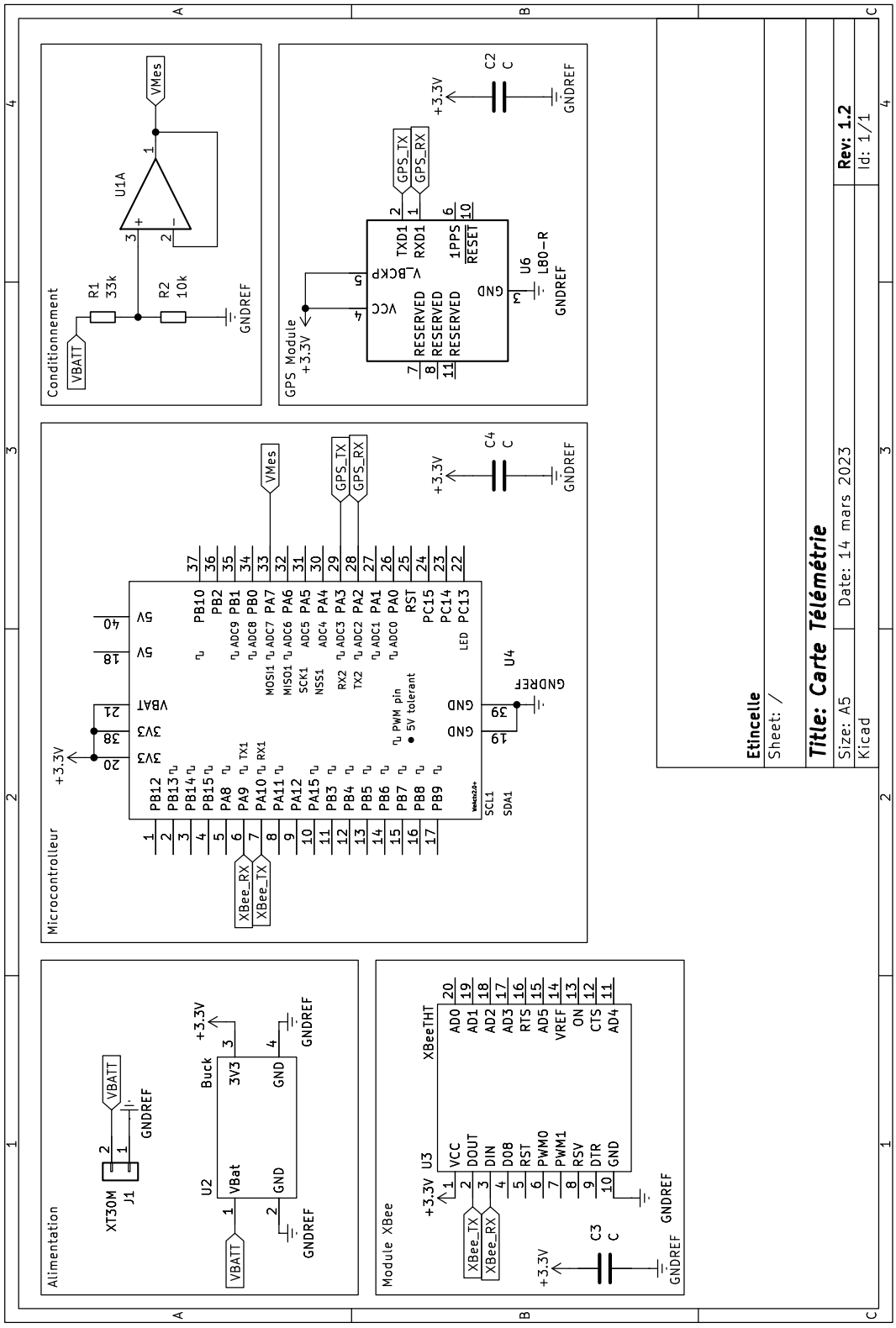
```
$GPGSA,A,3,21,30,31,06,24,29,,,,,,,2.8,1.5,2.4*33

$GPGSA             : type de trame
A                  : Mode 1, switch automatique entre mode 2D et 3D. (M pour manuel)
3                  : Mode 2, Mode de fix (1 : incorrecte, 2 :2D, 3 :3D)
21,30,31,06,24,29, : Numéro de satellite utilisé sur les canaux de 1 à 6
,,,,,,             : Pas de satellites actifs sur les canaux de 7 à 12
2.8                : PDOP (dilution de précision)
1.5                : Dilution de précision horizontale (HDOP)
2.4                : Dilution de précision verticale (VDOP)
*33                : Checksum
```

Dans le cas de plusieurs constellations, il est possible de trouver plusieurs trames GSA consécutives. Les numéros de 01 à 32 sont utilisés pour le GPS, 33 à 64 pour SBAS, 65 à 96 pour GLONASS.

Tournez la page S.V.P.

DT5 Schéma de la carte Télémétrie



DT6 ADC STM32

La formule utilisée pour modéliser la conversion ADC est

$$ADC_{mes} = \left\lfloor \frac{V_{In} \times 4096}{V_{Ref}} \right\rfloor$$

avec $\lfloor x \rfloor$ la partie entière de x, V_{In} la tension en entrée de l'ADC, V_{Ref} la tension de référence de l'ADC.

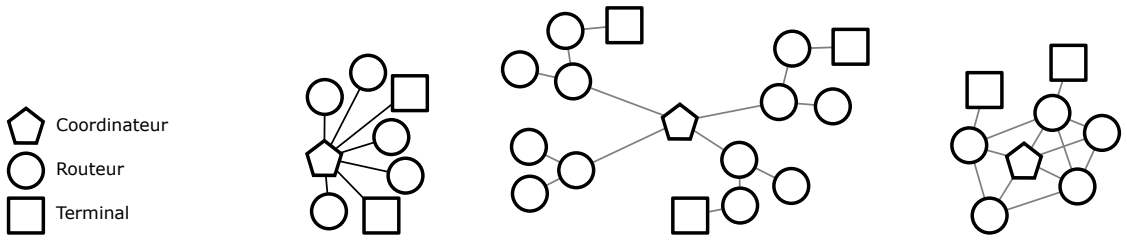
DT7 Structure d'un réseau zigbee

Les topologies d'un réseau zigbee définissent la manière dont les différents nœuds (coordinateur, routeurs, terminaux) sont interconnectés.
Remarque : tous les noeuds peuvent émettre et recevoir des messages.
Types de nœuds :

- Coordinateur : il initialise et contrôle le réseau. Il est unique dans chaque réseau.
- Routeur : il retransmet les messages entre les nœuds et peut également agir comme coordinateur en cas de défaillance du coordinateur principal. Chaque routeur étend la portée du réseau à son voisinage.
- Terminal : il ne peut pas relayer les messages et est généralement alimenté par batterie.

Principales topologies zigbee :

- Étoile : un coordinateur central contrôle tous les autres nœuds du réseau. Les nœuds terminaux communiquent directement avec le coordinateur. Toute défaillance du coordinateur entraîne l'arrêt du réseau. La portée est limitée par la distance entre les nœuds et le coordinateur.
- Arbre : un coordinateur à la racine et des routeurs formant des branches. Les nœuds terminaux sont connectés aux feuilles de l'arbre. La défaillance d'un nœud n'affecte qu'une partie du réseau.
- Maillée : chaque nœud peut communiquer directement avec plusieurs autres nœuds, créant ainsi un réseau interconnecté. La défaillance d'un nœud peut être contournée par d'autres chemins.



(a) Typologie de noeuds (b) Étoile (c) Arbre (d) Maillé

Topologies de réseau zigbee

Module de communication	Digi XBee® 3 zigbee 3.0	Digi XBee® 3 PRO zigbee 3.0
Débit	250Kbps	
Portée en intérieur	60m	90m
Portée en zone urbaine	60m	90m
Portée en champ libre	1200m	3200m
Puissance de transmission	+8dBm	+19dBm
Cryptage	AES 128/256 bits	

Caractéristiques des modules XBee

Tournez la page S.V.P.

DT8 Structure générale d'une trame XBee en mode API

Digi > API Mode > API frame structure

The structured data packets in API mode are called frames. They are sent and received through the serial interface of the device and contain the wireless message itself as well as some extra information such as the destination/source of the data or the signal quality.

When a device is in API mode, all data entering and leaving the module through the serial interface is contained in frames that define operations or events within the device.

An API frame has the following structure:

Start delimiter	Length		Frame data								Checksum
1	2	3	4	5	6	7	8	9	...	n	n+1
0x7E	MSB	LSB	API-specific structure								Single byte

Note MSB represents the most significant byte, and LSB represents the least significant byte.

Any data received through the serial interface prior to the start delimiter is silently discarded by the XBee. If the frame is not received correctly, or if the checksum fails, the data is also discarded and the module indicates the nature of the failure by replying with another frame.

Start delimiter

The start delimiter is the first byte of a frame consisting of a special sequence of bits that indicate the beginning of a data frame. Its value is always 0x7E. This allows for easy detection of a new incoming frame.

Length

The length field specifies the total number of bytes included in the frame data field. Its two-byte value excludes the start delimiter, the length, and the checksum.

Frame data

This field contains the information received or to be transmitted. Frame data is structured based on the purpose of the API frame:

Start delimiter	Length		Frame data								Checksum
			Frame type	Data							
1	2	3	4	5	6	7	8	9	...	n	n+1
0x7E	MSB	LSB	API frame type	Frame-type-specific data							Single byte

Note MSB represents the most significant byte, and LSB represents the least significant byte.

- **Frame type** is the API frame type identifier. It determines the type of API frame and indicates how the information is organized in the Data field.
- **Data** contains the data itself. The information included here and its order depends on the type of frame defined in the Frame type field.

DT9 Transmit Request frame

Frame type - 0x10

Description

This frame type is used to send payload data as an RF packet to a specific destination. This frame type is typically used for transmitting serial data to one or more remote devices.

64-bit addressing

For broadcast transmissions, set the 64-bit destination address to 0x000000000000FFFF
For unicast transmissions, set the 64-bit address field to the address of the desired destination node
If transmitting to a 64-bit destination, set the 16-bit address field to 0xFFFE

Zigbee-specific addressing information

A zigbee coordinator can be addressed in one of two ways :

- Set the 64-bit address to all 0x00s and the 16-bit address to 0xFFFE
- Set the 64-bit address to the coordinator's 64-bit address and the 16-bit address to 0x0000

Format

The following table provides the contents of the frame. For details on frame structure, see API frame format.

Size	Frame Field Description
8-bit	Start Delimiter Indicates the start of an API frame.
16-bit	Length Number of bytes between the length and checksum.
8-bit	Frame type Transmit Request - 0x10
8-bit	Frame ID Identifies the data frame for the host to correlate with a subsequent response frame. If set to 0, the device will not emit a response frame.
64-bit	64-bit destination address Set to the 64-bit IEEE address of the destination device. Broadcast address is 0x000000000000FFFF. zigbee coordinator address is 0x0000000000000000.
16-bit	16-bit destination address Set to the 16-bit network address of the destination device, if known. If transmitting to a 64-bit address, sending a broadcast, or the 16-bit address is unknown, set this field to 0xFFFE.
8-bit	Broadcast radius Sets the maximum number of hops a broadcast transmission can traverse. This parameter is only used for broadcast transmissions.
8-bit	Transmit options See TO (Transmit Options) for available options. If set to 0, the value of TO specifies the transmit options.
variable	Payload data Data to be sent to the destination device.
8-bit	Checksum 0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Tournez la page S.V.P.

DT10 Documentation simplifiée de `paho.mqtt.client`

The class `paho.mqtt.client.Client( ... )` is the main class used to communicate with an MQTT broker. General usage flow:x

- Use `connect()` to connect to a broker
- Use `loop_start()` to set a thread running to call `loop()` for you.
- Or use `loop_forever()` to handle calling `loop()` for you in a blocking function.
- Or call `loop()` frequently to maintain network traffic flow with the broker
- Use `subscribe()` to subscribe to a topic and receive messages
- Use `publish()` to send messages
- Use `disconnect()` to disconnect from the broker

Callbacks

A number of callback functions are available to receive data back from the broker. Typical callback functions are `on_connect`, `on_connect_fail`, `on_disconnect`, `on_message`, `on_publish`, `on_subscribe`, `on_unsubscribe`, `on_log`, `on_socket_open`, `on_socket_close`, `on_socket_register_write`, `on_socket_unregister_write` To use a callback, define a function and then assign it to the client:

```
def on_connect(client, userdata, flags, reason_code, properties):
    print(f"Connected with result code {reason_code}")

client.on_connect = on_connect
```

Methods

```
connect(host: str, port: int = 1883, keepalive: int = 60, bind_address: str = '',
        bind_port: int = 0, clean_start: bool | Literal[3] = 3,
        properties: Properties | None = None) -> MQTTErrorCode
```

Connect to a remote broker. This is a blocking call that establishes the underlying connection and transmits a CONNECT packet. Note that the connection status will not be updated until a CONNACK is received and processed (this requires a running network).
Parameters:

- host (str) – the hostname or IP address of the remote broker.
- port (int) – the network port of the server host to connect to. Defaults to 1883. Note that the default port for MQTT over SSL/TLS is 8883 so if you are using `tls_set()` the port may need providing.
- keepalive (int) – Maximum period in seconds between communications with the broker. If no other messages are being exchanged, this controls the rate at which the client will send ping messages to the broker.
- `clean_start` (bool) – (MQTT v5.0 only) True, False or `MQTT_CLEAN_START_FIRST_ONLY`. Sets the MQTT v5.0 `clean_start` flag always, never or on the first successful connect only, respectively. MQTT session data (such as outstanding messages and subscriptions) is cleared on successful connect when the `clean_start` flag is set. For MQTT v3.1.1, the `clean_session` argument of Client should be used for similar result.
- properties (Properties) – (MQTT v5.0 only) the MQTT v5.0 properties to be sent in the MQTT connect packet.

```
loop_forever(timeout: float = 1.0, retry_first_connection: bool = False) -> MQTTErrorCode
```

This function calls the network loop functions for you in an infinite blocking loop. It is useful for the case where you only want to run the MQTT client loop in your program. `loop_forever()` will handle reconnecting for you if `reconnect_on_failure` is true (this is the default behavior). If you call `disconnect()` in a callback it will return.
Parameters:

- timeout (int) – The time in seconds to wait for incoming/outgoing network traffic before timing out and returning.
- retry_first_connection (bool) – Should the first connection attempt be retried on failure. This is independent of the reconnect_on_failure setting.

```
publish(topic: str, payload: str | bytes | bytearray | int | float | None = None,
        qos: int = 0, retain: bool = False, properties: Properties | None = None)
        -> MQTTMessageInfo
```

Publish a message on a topic. This causes a message to be sent to the broker and subsequently from the broker to any clients subscribing to matching topics.
Parameters:

- topic (str) – The topic that the message should be published on.
- payload – The actual message to send. If not given, or set to None a zero length message will be used. Passing an int or float will result in the payload being converted to a string representing that number. If you wish to send a true int/float, use struct.pack() to create the payload you require.
- qos (int) – The quality of service level to use.
- retain (bool) – If set to true, the message will be set as the "last known good"/retained message for the topic.
- properties (Properties) – (MQTT v5.0 only) the MQTT v5.0 properties to be included.

```
subscribe(topic: str, qos: int = 0) -> tuple[MQTTErrorCode, int | None]
```

Subscribe the client to one topic.

```
subscribe("my/topic", 2)
```

- topic: A string specifying the subscription topic to subscribe to.
- qos: The desired quality of service level for the subscription. Defaults to 0.

Remarks on QoS

Quality of Service (QoS) in MQTT messaging is an agreement between sender and receiver on the guarantee of delivering a message.
When a client (e.g. a remote IoT device) is publishing to a broker, the client determines the QoS level for that message. When the broker sends the message to a subscribing client, the QoS set by the first client for that message is used again. So effectively the original publisher of any message determines the QoS of the message all the way to the final recipients.

- QoS Level 0 - at most once: This is the simplest, lowest-overhead method of sending a message. The client simply publishes the message, and there is no acknowledgement by the broker.
- QoS Level 1 - at least once: This method guarantees that the message will be transferred successfully to the broker.

  The broker sends an acknowledgement back to the sender, but in the event that that the acknowledgement is lost the sender won't realise the message has got through, so will send the message again. The client will re-send until it gets the broker's acknowledgement.

  This means that sending is guaranteed, although the message may reach the broker more than once.
- QoS Level 2 - exactly once: This is the highest level of service, in which there is a sequence of four messages between the sender and the receiver, a kind of handshake to confirm that the main message has been sent and that the acknowledgement has been received.

  When the handshake has been completed, both sender and receiver are sure that the message was sent exactly once.

Tournez la page S.V.P.

DT11 Référence SQLite et usage avec python

INSERT

Tiré de la page Wikipedia correspondante le 14/09/2024
INSERT est une commande SQL qui ajoute un ou plusieurs tuples dans une table d'une base de données relationnelle.

Forme basique

La commande INSERT a la syntaxe suivante :

```
INSERT INTO table (column1 [, column2, column3 ... ]) VALUES (value1 [, value2, value3 ... ])
```

Le nombre de colonnes doit être identique au nombre de valeurs. Si une colonne n'est pas spécifiée, sa valeur par défaut lui sera affectée. Les valeurs insérées doivent respecter toutes les contraintes tel que les clés étrangères, clés primaires, et les colonnes NOT NULL. Si la commande contient une erreur de syntaxe, ou si une contrainte n'est pas respectée, les valeurs ne sont pas insérées et une erreur est rapportée.
Example :

```
INSERT INTO film_cast (firstname, lastname) VALUES ('Raoul', 'Duke')
```

UPDATE

Tiré de la page Wikipedia correspondante le 14/09/2024
UPDATE est une commande SQL qui modifie un ou plusieurs tuples dans une table d'une base de données relationnelle :

- Soit toutes les lignes peuvent être mises à jour,
- soit un sous-ensemble peut être choisi à l'aide d'une condition.

Forme basique

La commande UPDATE a la syntaxe suivante :

```
UPDATE table_name SET column_name = value [, column_name = value ...] [WHERE condition]
```

Pour que la mise à jour soit réussie, l'utilisateur doit disposer des privilèges de manipulation de données (UPDATE privileges) sur la table ou la colonne, et la valeur mise à jour ne doit pas entrer en conflit avec toutes les contraintes applicables (telles que les clés primaires, les index uniques, les contraintes CHECK et les contraintes NOT NULL).

Paramètres nommés avec des dictionnaires dans SQLite3

Dans une requête SQL, un paramètre nommé est un marqueur que l'on remplace par une valeur spécifique au moment de l'exécution de la requête. Cela rend les requêtes plus lisibles, plus sûres (en évitant les injections SQL) et plus faciles à maintenir.
L'exemple ci dessous

```
requete = "SELECT * FROM utilisateurs WHERE age = :age"
cursor.execute(requete, {'age': 24})
```

sera équivalent à la requête SQLite :

```
"SELECT * FROM utilisateurs WHERE age = 24"
```

c'est la méthode `execute` de la bibliothèque sqlite3 qui accepte un dictionnaire de paramètres et remplace les marqueurs dans la requête par les valeurs correspondantes du dictionnaire.

DT12 Service Moniteur - Codes

ecoute_mqtt.py

```python
import paho.mqtt.client as mqtt
import threading

class EcouteMQTT:
    def __init__(self, broker_address, port, topics = None):
        self.client = mqtt.Client()
        self.client.on_connect = self.on_connect
        self.client.on_message = self.on_message
        self.topics = topics
        self.broker_address = broker_address
        self.port = port
        self.t = threading.Thread(target=self.client.loop_forever)
        # self.verrou = False
        self.messages = []

    def on_connect(self, client, userdata, flags, rc):
        print(f"Connecté au broker {self.broker_address} - code de retour {str(rc)}")
        for topic in self.topics :
            self.client.subscribe(f"{topic}")

    def on_message(self, client, userdata, msg):
        self.messages.append(msg.payload.decode("utf-8"))

    def start(self):
        self.client.connect(self.broker_address, self.port)
        self.t.start()

    def stop(self):
        self.client.disconnect()
```

gestion_db.py

```python
import sqlite3

class SQLiteManager:
    def __init__(self, db_file):
        self.conn = sqlite3.connect(db_file)
        self.cursor = self.conn.cursor()
        self.create_rq = f"""CREATE TABLE IF NOT EXISTS drones (
                                id_drone CHAR(12) NOT NULL PRIMARY KEY,
                                date_enregistrement text NOT NULL,
                                lat REAL,
                                long REAL,
                                alt REAL
                        ); """
        self.insert_rq = f" A COMPLÉTER "
        self.update_rq = f" A COMPLÉTER "
        self.cleanup_rq = f"DELETE FROM drones"
        self.execute(self.cleanup_rq)

    def execute(self, query, params=None):
        try:
            if params:
                self.cursor.execute(query, params)
```


Tournez la page S.V.P.

```
            else:
                self.cursor.execute(query)
            self.conn.commit()

            return self.cursor.fetchall()
        except sqlite3.Error as e:
            print(f"Une erreur s'est produite : {e}")
            return None

    def insert_or_update(self, datas):
        try:
            self.cursor.execute(self.insert_rq, datas)
        except sqlite3.IntegrityError:
            self.cursor.execute(self.update_rq, datas)

    def affiche(self):
        requete = f"SELECT * FROM drones"
        self.cursor.execute(requete)
        self.conn.commit()
        messages = (self.cursor.fetchall())
        for message in messages:
            print(message)
        print("\n\n")

    def close(self):
        self.conn.close()

```

mqtt_monitor.py

```
import datetime, time
from ecoute_mqtt import EcouteMQTT
from gestion_db import SQLiteManager

def main():
    """ Service qui récupère les messages MQTT du topic ?????? et
    ajoute les données à la base de donnée sqlite moniteur.db """

    ecoute = EcouteMQTT(A COMPLÉTER)
    ecoute.start()
    database = SQLiteManager("moniteur.db")

    # Création d'une table (exemple)
    database.execute(database.create_rq)

    while True:
        time.sleep(1)
        if len(ecoute.messages) > 0:
            for message in ecoute.messages:
                payload_str = message
                data = payload_str.split(':')
                datas = {'id': data[0],
                         'new_date' : datetime.datetime.now().isoformat("T","seconds"),
                         'lat' : data[1],
                         'long' : data[2],
                         'alt' : data[3],}
```

```
                # Insertion de données
                database.insert_or_update(datas)
                database.affiche()

            ecoute.messages = []
```

moniteur.py

```
import datetime, time
import sqlite3
import paho.mqtt.client as mqtt

############## Spécifique moniteur ##############
def drones_hors_delais(base_de_donnees, delai):
    """ Récupère les IDs des drones dont l'horodatage est
        plus ancien de 'delai' secondes ou plus.
    Args:
        base_de_donnees (str): Le chemin vers la base de données SQLite.
    Returns:
        list: Une liste contenant les IDs des drones perdus.
    """
    try:
        # Connexion à la base de données
        connexion = sqlite3.connect(base_de_donnees)
        curseur = connexion.cursor()

        # Calcul de l'horodatage limite
        horodatage_limite = datetime.datetime.now() - datetime.timedelta(seconds=delai)
        iso_limite = horodatage_limite.isoformat("T","seconds")

        # Requête SQL pour récupérer les IDs
         ## A compléter
        curseur.execute(requete)

        # Récupération des résultats et fermeture de la connexion
        hors_delai = curseur.fetchall()
        connexion.close()
        return hors_delai

    except sqlite3.Error as erreur:
        print(f"Une erreur s'est produite : {erreur}")
        return []

def main():
    database = "moniteur.db"

    while True:
        drones_perdus = drones_hors_delais(database, 15)
        print(drones_perdus) # A remplacer
        time.sleep(5)
```