MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE LA JEUNESSE
ET DES SPORTS
Liberté
Égalité
Fraternité

EAE SIN 3

SESSION 2021

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 électronique de poche - y compris calculatrice programmable, alphanumérique ou à écran graphique – à fonctionnement autonome, non imprimante, autorisée conformément à la circulaire n° 99-186 du 16 novembre 1999.

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

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.

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

IMPRIMERIE NATIONALE – 21 0100 – D'après documents fournis

URBANLOOP

Sommaire

Ce document se décompose en trois parties :

PRESENTATION DU SYSTEME

1. Introduction

Urbanloop est un prototype de transport public, basé sur l'idée qu'un système de transport collectif peut être plus efficace en faisant circuler les voyageurs de manière individuelle dans des véhicules légers et compacts, qu'ensemble dans des véhicules lourds et encombrants.

Ce changement de paradigme devient possible grâce aux récents progrès dans le domaine de la motorisation électrique à faible consommation (scooters, trottinettes, gyropodes ou vélos électriques) mais aussi grâce aux progrès réalisés dans les sciences du numérique qui ont démocratisé les notions de véhicules partagés, véhicules autonomes ou de transport à la demande.

Le système Urbanloop est aujourd'hui développé par la SAS Urbanloop en partenariat avec l'Université de Lorraine dans la région Grand Est.

Il s'agit d'un système de transport public urbain rendant un service individuel en site propre conçu pour répondre aux 3 objectifs sociétaux suivants :

- Concurrencer efficacement l'utilisation de la voiture en ville ;
- Minimiser la consommation d'énergie, l'impact sur l'environnement et l'emprise urbaine ;
- Garantir un haut niveau d'accessibilité et de sécurité.

2. Présentation du système

Pour atteindre ces objectifs sociétaux, les principales évolutions techniques d'Urbanloop par rapport aux systèmes actuels sont :

- Les petites dimensions du matériel roulant ;
- La densité élevée de véhicules roulant simultanément sur la voie ;
- Des stations en dérivation de la voie principale (figure 1) permettant aux voyageurs d'embarquer et de débarquer sans interrompre le flux principal (concept similaire à celui des arrêts d'autoroute) ;

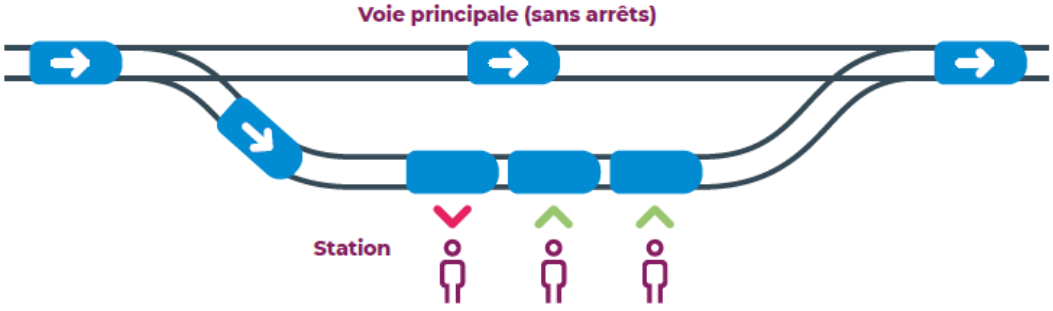


Figure 1 : des stations en dérivations pour la montée et la descente

- Un système d'aiguillage automatisé (figure 2) permettant aux véhicules de passer d'une ligne à une autre sans correspondance ;

Tournez la page S.V.P.

Figure 2 : système d'aiguillage vu du dessus

– Une insertion urbaine (figure 3) qui s'adapte à la configuration modulable (sous le sol ou sur le sol) ;

Figure 3 : exemple d'insertion urbaine sous une piste cyclable, le voyageur est visible par transparence à gauche

– Un système de billetterie sur application mobile ;
– La dématérialisation de l'exploitation et la rationalisation des opérations de maintenance.

La flotte de véhicules roule simultanément sur la voie avec une densité élevée et la position de tous les véhicules est contrôlée avec précision par un système de contrôle commande afin d'assurer des insertions sécurisées sur la voie.

Les véhicules, aussi appelés *capsules,* sont guidés et alimentés par un rail, l'alimentation est 100% électrique, sans batterie**.** Leur propulsion est assurée par

un moteur synchrone à aimant. Le variateur du moteur est alimenté par le rail en 72V continu.

La flotte de véhicules est composée de véhicules identiques avec des châssis et des carrosseries communes mais avec 2 types d'aménagements intérieurs possibles (figure 4) :

- Véhicules équipés de 2 sièges face à face ;
- Véhicules PMR pouvant accueillir une personne en fauteuil roulant et un accompagnateur ou un voyageur et son vélo.

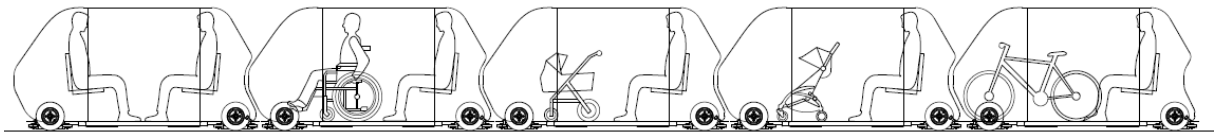


Figure 4 : différentes configurations de capsules

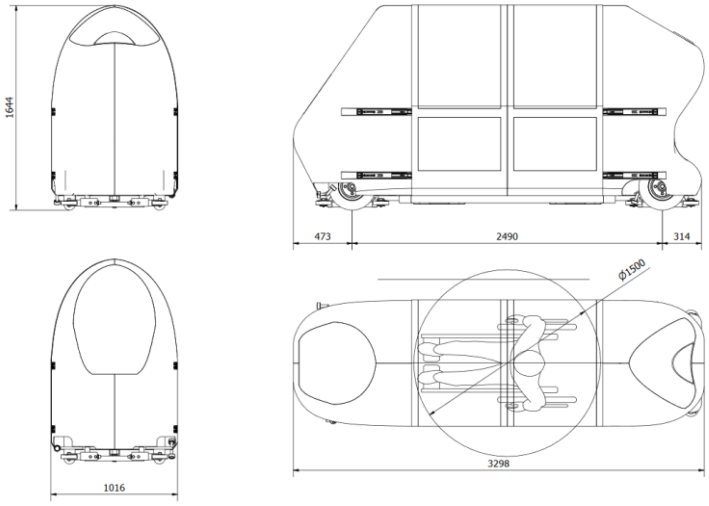


Figure 5 : dimensions de la capsule

Les principales caractéristiques d'interface du matériel s'inscrivent dans les limites suivantes :

DIMENSIONS (figure 5) :

- Longueur du véhicule : 3,3 m
- Largeur du véhicule : 1,01 m
- Hauteur du véhicule : 1,64 m
- Masse à vide : 100 kg
- Masse en charge maxi (2 personnes) : 350 kg

ALIMENTATION ELECTRIQUE : 72 V en courant continu par le rail

INFRASTRUCTURE EMPRUNTABLE :

- Pente maximale franchissable : 150/1000
- Courbe minimale franchissable : rayon de 10 m
- Type de quais : affleurant

Tournez la page S.V.P.

PERFORMANCES :

- Vitesse maximale : 75 km/h
- Décélération maximale en charge : -2,5 $m.s^{-2}$
- Accélération maximale en charge : 1,35 $m.s^{-2}$

3. Diagramme de blocs

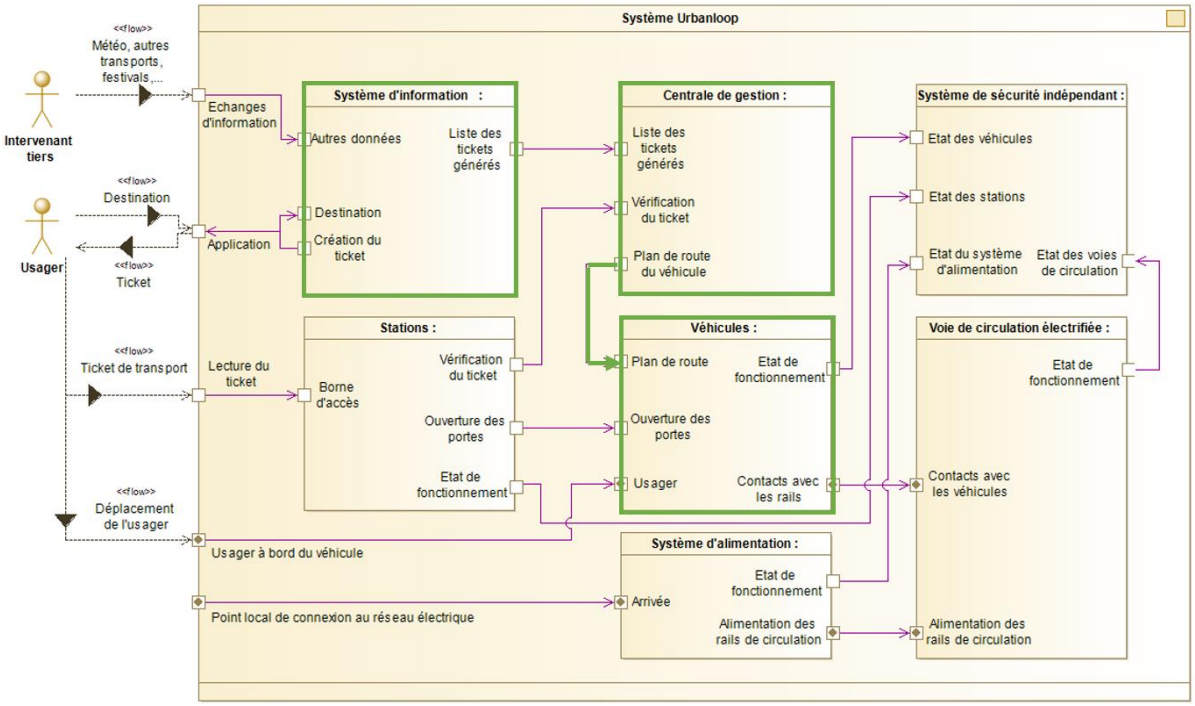


Figure 6: diagramme de blocs internes du système Urbanloop

La figure 6 présente le diagramme de blocs internes du système Urbanloop. Dans le cadre de cette épreuve, seules les parties surlignées en vert seront abordées.

L'épreuve se décompose en 4 parties distinctes et indépendantes :

La partie 1 est relative à la conception du bloc système d'information. Ce bloc traite de la commande des tickets de transport sur smartphone, de la communication entre l'application smartphone et le serveur applicatif ainsi que de l'architecture matérielle du système d'information ;

La partie 2 traite de la communication sans fil entre la centrale de gestion et les véhicules ;

La partie 3 traite de l'algorithme de génération des plans de route transmis aux capsules. Cet algorithme est exécuté dans le bloc de centrale de gestion ;

La partie 4 traite du module de positionnement de la capsule sur le circuit. Ce module est un sous-bloc du bloc véhicule.

PARTIE 1 : CONCEPTION DU SYSTEME D'INFORMATION

1. Application smartphone

Pour rendre le système agréable aux usagers, une application sur terminal mobile est mise en œuvre (figure 7). Ainsi, l'utilisateur s'interface avec le système d'information à l'aide d'une application smartphone. Il se connecte à l'application, commande ses tickets en sélectionnant sa station de destination. L'application génère un QR code qui est validé par une borne à la station de départ.

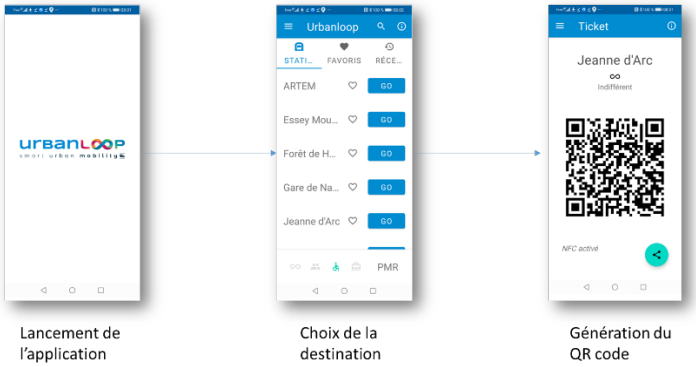


Figure 7 : workflow de l'application pour générer un ticket de transport

Le QR Code contient sous forme binaire chiffré les informations suivantes :

- Informations relatives au service Urbanloop : 8 bits
- Identifiant du client : 32 bits
- Identifiant de la station d'arrivée : 32 bits
- Date de réservation du voyage : 64 bits
- Somme de contrôle : 8 bits

Lors de la lecture du code, la borne est ensuite en capacité d'identifier le trajet et de procéder à l'ouverture d'une capsule de transport.

Q1. En utilisant les spécifications QR Code (document technique DT1), quelle est la taille appropriée pour le QR Code généré par l'application avec une correction d'erreur de minimum 25 % ?

Q2. Pour générer ce QR Code, la librairie Android QR Generator est utilisée. Une classe utilitaire (toutes les méthodes publiques sont statiques) utilise cette librairie afin de générer une image Bitmap qui est affichée dans l'application. En utilisant le document technique DT2, implémenter, en java, dans le document réponse DR1 la méthode statique permettant de générer le QR Code.

Q3. Lors de la première connexion à l'application, une adresse mail est demandée à l'utilisateur. Le programme vérifiant la validité de l'adresse mail utilise une expression régulière dont les règles de syntaxe sont rappelées dans le document technique DT3. Proposer une expression régulière qui valide la syntaxe respectant les contraintes décrites dans l'extrait du cahier des charges de la figure 8.

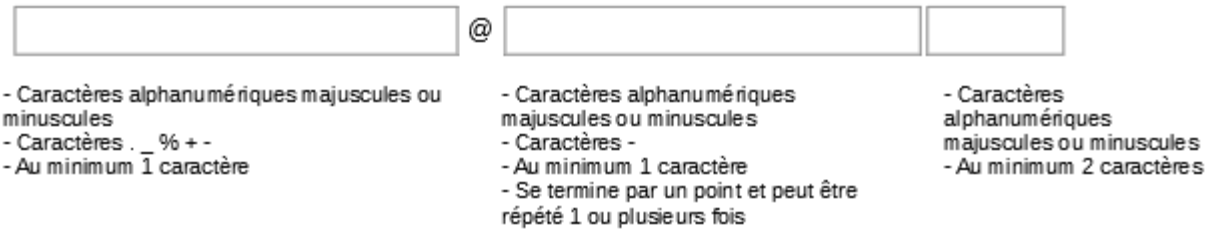


Figure 8 : extrait du cahier des charges précisant les règles syntaxiques acceptées pour une adresse mail

Q4. Afin de vérifier la validité de l'expression régulière, un cas test est réalisé avec JUnit. Compléter chaque assertion du test unitaire présent dans le document réponse DR1 afin de valider le bon fonctionnement de l'expression régulière.

2. Conception du webservice REST

Toutes les opérations effectuées par l'application mobile sont transmises sur un serveur offrant un webservice de type REST. En cas d'absence de connexion internet, il n'est pas possible de consulter les informations en temps réel ou de réserver un voyage. Ce webservice est développé en Python avec le framework Flask. Il est programmé dans l'objectif de minimiser l'empreinte mémoire limitée tout en proposant une simplicité de développement et de maintenance. Le système est couplé à une base de données PostgreSQL pour le stockage des données. La figure 9 présente l'architecture du système d'information.

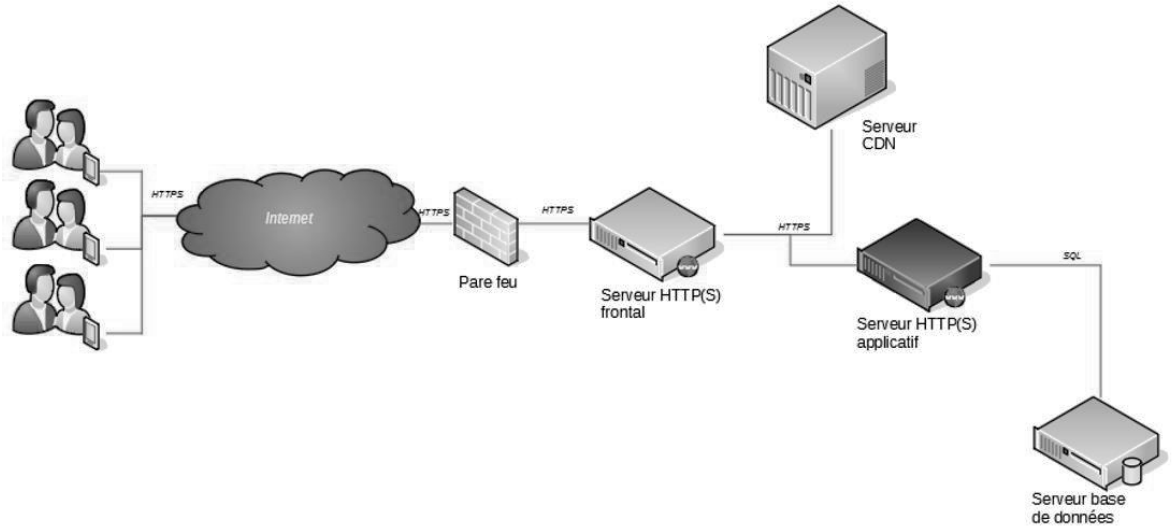


Figure 9 : architecture du système d'information

Q5. Pour empêcher qu'un utilisateur malveillant vole le compte d'un tiers par force brute, un algorithme a été décrit sous forme de logigramme dans le cahier des charges. Après implémentation de l'algorithme par le développeur, il apparaît que les utilisateurs ne peuvent plus se connecter après 5 connexions (validées ou non). Proposer une modification du logigramme du document réponse DR2 pour corriger l'erreur commise.

Q6. Le modèle conceptuel de la base de données du système d'information est donné, de manière partielle, dans le document technique DT4. Ecrire une requête

SQL qui renvoie la liste des 3 stations les plus utilisées pour le client dont l'adresse mail est « dupont.robert@fournisseur.fr »

3. Communication client serveur

Q7. Lors de l'envoi d'une requête d'authentification, le serveur retourne les données ci-dessous. Dans quel format la réponse du serveur est-elle donnée ? Quel est l'intérêt de ce format ?

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 76
Server: Werkzeug/1.0.1 Python/3.8.5
Date: Tue, 15 Sep 2020 14:32:27 GMT
{
  "error": null,
  "token": "f1e6adb0a5ea45fcb8dbfc7e7525f72c9551af92"
}
```

Q8. Selon la documentation du protocole HTTP v1.1 (document technique DT5), quel code erreur devra renvoyer le serveur dans le cas d'un échec d'authentification (mail ou mot de passe incorrect) ?

Q9. Pour une raison de fiabilité, le cahier des charges demande à la fin du projet une redondance matérielle du serveur applicatif. Proposer une nouvelle architecture du système d'information intégrant cette contrainte.

PARTIE 2 : TRANSMISSION DE LA FEUILLE DE ROUTE

L'objectif de cette partie est de mettre en œuvre la communication entre les capsules et la centrale de gestion. Cette communication est primordiale pour la gestion du flux de capsules sur le réseau Urbanloop, elle doit donc être fiable et temps réel.

Lorsqu'un utilisateur achète un trajet, une feuille de route est générée permettant de gérer le déplacement de la capsule. La capsule renvoie alors régulièrement un certificat au serveur pour indiquer le suivi de la feuille de route.

Un réseau de communication sans fil est établi entre les capsules et la centrale de gestion distante de quelques centaines de mètres et jusqu'à 2 km. Ce réseau est basé sur une liaison radio utilisant la technologie Zigbee. Des modules radio Xbee-PRO, communiquant selon le standard Zigbee, permettent la mise en œuvre du réseau de communication, un extrait du manuel d'utilisation de ces modules est donné dans le document technique DT6.

1. Etablissement d'un réseau sécurisé entre les modules Xbee

Un réseau Zigbee est composé d'un coordinateur qui coordonne toutes les opérations sur le réseau, de routeurs et de nœuds terminaux (end-device).

Q10. A partir du document technique DT6, proposer un choix de topologie pour le réseau formé par les capsules et le centre de maintenance hébergeant la centrale de gestion. Justifier ce choix en précisant qui prend le rôle de « coordinateur », de « routeur » et de « end-device ».

Q11. Les modules Xbee sont configurés en mode API Operation. Justifier l'utilisation de ce mode.

Chaque module possède une adresse longue sur 64 bits unique configurée en usine et une adresse courte sur 16 bits appelée adresse réseau qui est transmise aux nœuds qui se connectent au réseau par le coordinateur.

Les nœuds terminaux sont préconfigurés pour rejoindre automatiquement le réseau situé dans leur zone de portée et chaque nœud possède un identifiant (Node Identifier) unique constitué d'une chaîne de caractères contenant la lettre C en majuscule et le numéro de capsule sur 3 octets (codé en ASCII).

Chaque module Xbee-PRO communique avec son hôte (microprocesseur ou microcontrôleur) par une liaison série de type UART. Lors d'une communication entre modules, l'hôte envoie une trame par liaison série à son module qui la transmet par radiofréquence au module destination qui à son tour transmet à son hôte une trame par liaison série. La figure 10 illustre la transmission de données entre modules.

Figure 10 : transmission de données entre modules

A chaque mise en route du réseau Urbanloop, les nœuds terminaux recherchent la présence d'un coordinateur et demandent à rejoindre le réseau. Lorsque l'association entre le coordinateur et le nœud est effective, le nœud transmet en broadcast une trame « Node Identification Indicator » contenant, entre autres, les deux adresses du module et le Node Identifier.

Les deux trames suivantes correspondent à l'émission de « Node Identification Indicator » de deux nœuds terminaux (end-device) lors de leur connexion au réseau.

Trame Node Identification Indicator nœud 1 (notation hexadécimale) : 7E 00 23 95 00 13 A2 00 41 B1 AB 01 93 47 02 93 47 00 13 A2 00 41 B1 AB 01 43 30 30 35 00 00 00 02 02 C1 05 10 1E 3E

Trame Node Identification Indicator nœud 2 (notation hexadécimale) : 7E 00 23 95 00 13 A2 00 41 B6 3D 21 C1 A6 02 C1 A6 00 13 A2 00 41 B6 3D 21 43 30 31 34 00 00 00 02 02 C1 05 10 1E B6

Q12. A partir du manuel d'utilisation des modules Xbee (DT6) et de la description des trames API (DT7), indiquer à partir de quel octet se trouvent les données spécifiques à la trame de Node Identification Indicator et compléter sur le document réponse DR3 les informations relatives à chacun des nœuds (la table des codes ASCII est donnée en document technique DT13).

La fiabilité des transmissions sur le réseau est assurée par l'ajout d'un checksum à chaque trame série, doublé d'un code de CRC (Cyclic Redundancy Check) ajouté à la trame RF. En cas d'erreur à la réception, la trame est rejetée et un code d'erreur est renvoyé. La trame est réémise par la source jusqu'à trois fois.

Q13. Expliquer la différence entre un code CRC et un checksum. Citer un avantage de chaque technique.

Q14. Vérifier la validité de la trame envoyée par le nœud 1 en utilisant les informations du document technique DT6.

2. Synchronisation des échanges entre la centrale de gestion et les capsules

La gestion du trafic sur le réseau Urbanloop implique que chaque capsule soit capable de suivre la feuille de route reçue, et donc de vérifier sa position à chaque instant du trajet. Ce suivi nécessite une synchronisation des horloges du serveur et des différentes capsules. Pour cela, avant chaque transmission de feuille de route, le

serveur transmet l'heure UTC pour synchronisation de l'horloge RTC (Real Time Clock) de la capsule. Le document technique DT8 documente les fonctions de gestion du temps et de mise à l'heure de l'horloge RTC.

La communication avec les modules Xbee est implémentée via la librairie Xbeelib écrite en C++. La classe correspondant au protocole Zigbee est la classe XbeeZB. La réception des données déclenche la transmission d'une trame de type Receive Packet (ID 0x90) sur la liaison UART. La bibliothèque Xbeelib implémente une seule fonction de réception de données. Pour reconnaître la nature des différentes trames reçues par une capsule, les données sont précédées d'un identifiant sur un octet : 0x48 pour la transmission de l'heure, 0x46 pour la transmission de la feuille de route.

L'interrogation de l'horloge du serveur par la fonction `time(time_t* timer);` renvoie la valeur 1 599 208 323 secondes en décimal. L'heure est codée selon la norme UNIX Epoch ou POSIX time, en nombre de secondes depuis le 1er janvier 1970 à 0h00, sur 32 bits.

Q15. Compléter sur le document réponse DR3 la trame Transmit request (décrite en DT7) que le coordinateur envoie à la capsule du nœud 1 pour transmettre l'heure (utiliser l'adressage 16 bits).

Le coordinateur transmet ensuite la feuille de route à la capsule. Chaque trame de Transmit Request peut contenir au maximum 255 octets de données. La feuille de route contient une colonne de temps en secondes et une colonne de position en mètres. La fréquence d'échantillonnage est de 1 Hz et un trajet dure en moyenne 5 minutes.

Q16. Calculer le nombre d'octets nécessaires pour coder le temps de trajet de la capsule.

Q17. La position étant codée sur 2 octets, calculer la taille du fichier binaire contenant la feuille de route et le nombre de trames nécessaires à sa transmission vers la capsule en tenant compte du fait que, pour faciliter la réception des trames, les couples temps-position sont toujours envoyés en entier.

Q18. Compléter le code de la fonction `receive_cb` sur le document réponse DR4 pour extraire les données de la trame API reçue du coordinateur et, soit mettre à l'heure l'horloge RTC du microcontrôleur, soit remplir un tableau à deux colonnes avec les temps et positions reçues. Le document technique DT8 documente les fonctions de gestion du temps du langage C++ ainsi que la fonction `set_time` spécifique à la mise à l'heure de l'horloge RTC du microcontrôleur.

Lors du suivi de la feuille de route, chaque capsule émet vers le coordinateur un certificat d'acquittement de suivi avec la même fréquence d'échantillonnage.

La classe XbeeZB implémente une fonction send_data_to_coordinator qui émet une trame API de Transmit_Request vers le coordinateur :

```
TxStatus XBeeZB::send_data_to_coordinator(const uint8_t *const data,
uint16_t len, bool syncr)
{
    const uint64_t remoteaddr = ADDR64_COORDINATOR;

    TxFrameZB frame = TxFrameZB(remoteaddr, ADDR16_UNKNOWN,
BROADCAST_RADIUS_USE_NH, _tx_options, data, len);
    if (syncr) {
        return send_data(&frame);
    } else {
        frame.set_data(0, 0); /* Set frame id to 0 so there is no
answer */
        send_api_frame(&frame);
        return TxStatusSuccess;
    }
}
```

Q19. Donner la définition de la constante ADDR64_COORDINATOR. Expliquer le rôle du booléen ***syncr*** passé en paramètre de cette fonction.

Q20. Le paramètre ***syncr*** a la valeur TRUE par défaut, préciser à quel endroit est faite cette initialisation. Discuter l'intérêt de conserver ce paramétrage pour la fiabilisation des échanges entre le serveur et les capsules.

3. Synthèse

Q21. Le débit Radio Fréquence (RF) d'un réseau Xbee est de 250 kbits.s^{-1} et la liaison UART entre le module et son hôte est garantie jusqu'à 115 200 bauds. En utilisant la description des trames de Transmit-Request (DT7) et les questions précédentes, calculer le temps de transmission d'une feuille de route correspondant à un trajet de 5 min en UART et en RF.

Q22. La norme Zigbee définie une latence minimum de 1 octet et une latence maximum de 32 octets dans les réseaux comprenant une centaine de nœuds.

Définir le concept de latence et calculer le délais minimum et maximum de latence pour la transmission RF.

Q23. L'activation des réponses des modules à chaque réception de trame permet la fiabilisation des transmissions ; discuter cette solution en termes de transmission temps réel en considérant que le nombre de capsules est de l'ordre de la centaine.

Q24. Dans la cadre des différentes transmissions étudiées précédemment et au vu des objectifs de cette partie, indiquer dans quelle mesure le choix du réseau Zigbee est adapté. Proposer une évolution technologique permettant l'amélioration du réseau.

PARTIE 3 : GENERATION DES FEUILLES DE ROUTES

Cette partie étudie la génération des trajectoires qui est basée sur une simulation numérique des véhicules dans le réseau. L'objectif de cet algorithme est de calculer une vitesse de la capsule en tout point du circuit qui garantisse la sécurité. Une fois toutes les vitesses calculées, elles sont envoyées aux capsules sous la forme d'une feuille de route.

Dans la réalité le réseau Urbanloop est constitué de tronçons de rails formant des boucles interconnectées avec des stations en dérivation du flux principal. D'un point de vue informatique, le réseau de rails est modélisé, en python, uniquement avec une liste de nœuds (figure 11) et une liste de segments.

Les nœuds sont caractérisés par un identifiant, une coordonnée X (en pixel), une coordonnée Y (en pixel) et un type. Le type décrit la particularité du nœud et il existe 4 types : les nœuds standards (codés par le type 1), les nœuds en intersection ouvrante (codés par le type 2), les nœuds en intersection fermante (codés par le type 3) et les nœuds correspondant à une place en station (codés par le type 4).

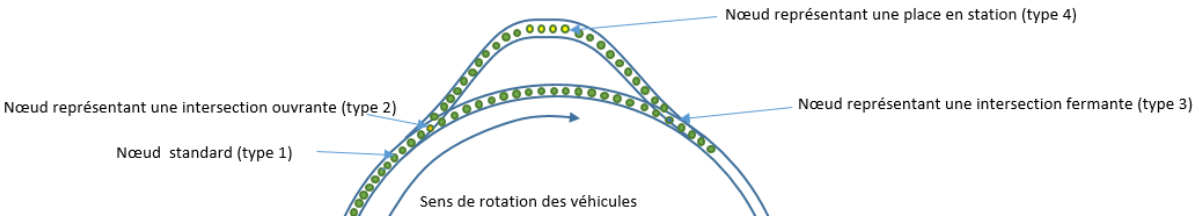


Figure 11 : représentation du circuit par une liste de nœuds

La liste des nœuds n'est pas suffisante pour décrire le réseau. Une liste d'arcs décrit les différents tronçons de rails existants (figure 12). Chaque arc est caractérisé par un identifiant, un nœud de départ, un nœud d'arrivée ainsi qu'une limitation de vitesse.

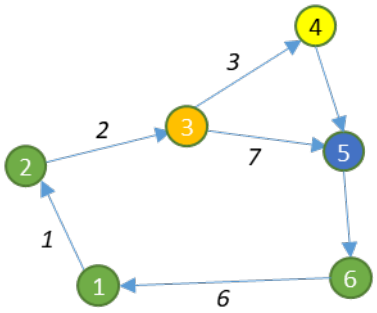


Figure 12 : graphe simplifié composé de 3 nœuds standards (en vert), d'un nœud de type 2 (en orange), d'un nœud de type 4 (en jaune) et d'un nœud de type 3 (en bleu)

Les graphes sont sauvegardés dans un fichier texte dont la structure respecte les 2 règles décrites ci-dessous :

1. Pour chaque nœud, une ligne est écrite au format suivant :

```
N <id> <Coordonnee_X> <Coordonnee Y> <Type>
```

2. Pour chaque segment, une ligne est écrite au format suivant :

```
S <id> <id_Noeud_Origine> <id_Noeud_Destination> <limitation de vitesse>
```

Ainsi le graphe de la figure 12 est sauvegardé dans le fichier texte suivant :

```
N 1 195 238 1
N 2 118 144 1
N 3 255 110 1
N 4 337 15 4
N 5 372 105 3
N 6 357 249 1
S 1 1 2 22.2
S 2 2 3 22.2
S 3 3 4 22.2
S 4 4 5 22.2
S 5 5 6 22.2
S 6 6 1 22.2
S 7…….
```

Q25. Ecrire la ligne correspondant au segment 7 en considérant que la limitation de vitesse sur ce segment est de $22{,}2\ m.s^{-1}$.

Q26. Un éditeur graphique de trajectoires permet à l'utilisateur de dessiner et sauvegarder ses tracés. Pour être réaliste, un tracé nécessite environ 200 nœuds par kilomètre de ligne, stations incluses. Il peut être considéré que le nombre de nœuds de types 2, 3 et 4 est négligeable par rapport aux nœuds de type 1. Quelle va être, en octets, la taille approximative d'un fichier texte correspondant à 5 km de ligne ? Justifier la réponse en précisant les hypothèses prises pour l'estimation.

Q27. Pour simuler le comportement des véhicules sur le réseau, le logiciel lit le fichier texte et crée en mémoire différents objets pour encapsuler ces données. A partir de la structure du fichier texte et à partir du diagramme de classes du programme donné en document technique DT9, compléter le programme dans le document réponse DR5 permettant d'initialiser un graphe.

La simulation du déplacement des véhicules sur les voies est réalisée en considérant que les véhicules se déplacent en ligne droite sur les différents segments du graphe. Lorsqu'un véhicule se déplace sur un segment, il commence par accélérer de manière constante à $\gamma = +1{,}35\ m.s^{-2}$ jusqu'à atteindre la vitesse maximale autorisée sur le segment. Une fois sa vitesse atteinte il la maintient.

Q28. Compléter la fonction proposée dans le document réponse DR6 permettant avec Euler de tenir à jour un vecteur d'état composé de la distance parcourue et de la vitesse courante à chaque pas de temps.

Un trajet peut être composé de virages dans lesquels les véhicules devront ralentir. Les virages sont modélisés par une succession de segments. L'objectif des

questions suivantes est de programmer les fonctions qui permettront de définir, pour chaque segment, la vitesse maximale autorisée pour que l'accélération latérale générée due aux changements de directions ne dépasse pas la valeur de $\gamma_{max} = 2\ m.s^{-2}$.

Q29. En ramenant le problème à 2 segments, et en se ramenant à la figure 13, quelle serait la valeur de l'accélération théorique au moment précis où un véhicule roulant à vitesse constante changerait de direction au nœud B pour prendre celle du nœud C ?

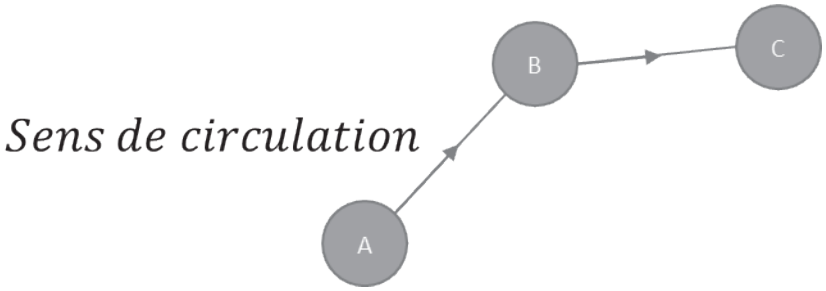


Figure 13 : virage élémentaire

Q30. En simulation numérique, quel serait le problème si cette accélération était calculée à partir d'une différence de vitesse entre 2 pas de temps ?

Pour pallier ce problème et pour dimensionner la vitesse admissible à l'entrée de chaque segment, la trajectoire ABC est considérée être un arc de cercle (cercle passant par les 3 nœuds ABC), dont le rayon est appelé R. La cinématique du véhicule est considérée comme circulaire uniforme. Il est aussi considéré que la vitesse admissible sur un segment est la vitesse maximale admissible au point du début segment.

Q31. Exprimer la vitesse maximale V_{max} admissible sur le segment BC en fonction du rayon R et de l'accélération maximale admissible γ_{max}.

Q32. A partir d'une liste de 3 tuples où chaque tuple contient les coordonnées en mètres des 3 nœuds d'une trajectoire [(x_a, y_a), (x_b, y_b), (x_c, y_c)], écrire une fonction prenant en paramètre cette liste et qui retourne le rayon R d'un cercle passant par ces 3 nœuds.

L'objectif des questions suivantes est de réaliser le programme qui permet de calculer les endroits où le véhicule doit accélérer ou décélérer pour optimiser le temps de trajet tout en respectant les contraintes de vitesse sur chacun des segments.

Par exemple, sur la figure 14, le véhicule roule sur 3 segments. Il démarre avec une vitesse nulle et accélère de manière constante à $+1{,}35\ m.s^{-2}$ pour atteindre la vitesse limite du segment 1 qui est de $20\ m.s^{-1}$. Puis, à un instant donné, le véhicule doit freiner avec une accélération négative constante de $-1{,}35\ m.s^{-2}$ pour entrer sur le segment 2 à la vitesse maximale autorisée de $10\ m.s^{-1}$. Puis il maintient cette vitesse de $10\ m.s^{-1}$ pendant toute la traversée du segment 2 avant d'accélérer dès l'entrée sur le segment 3.

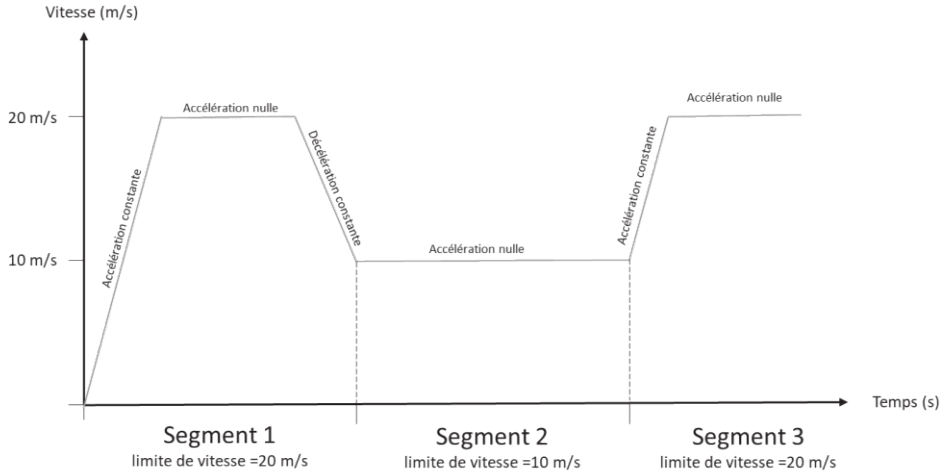


Figure 14 : évolution de la vitesse au cours du temps

Q33. Lorsqu'un véhicule roule à une vitesse V et souhaite atteindre une vitesse cible V_{cible} avec une accélération constante γ, exprimer la durée d_{cible} nécessaire pour atteindre une vitesse cible en fonction de V, V_{cible} et de l'accélération γ.

Q34. Exprimer la distance d pour atteindre cette vitesse cible en fonction de V, V_{cible} et de l'accélération γ. Donner une application numérique lors d'un freinage avec $V = 20\ m.s^{-1}, V_{cible} = 10\ m.s^{-1}\ et\ \gamma = -1{,}35\ m.s^{-2}$

Dans un trajet composé de plusieurs segments, il est parfois nécessaire d'anticiper, plusieurs segments en amont, un freinage. Les informations caractérisant les segments sont stockées dans une liste de tuples. Chaque tuple contient 2 informations : la longueur du segment en mètres et la vitesse maximale admissible sur ce segment en $m.s^{-1}$.

Q35. Dans le document réponse DR6, proposer une fonction qui renvoie la distance à parcourir avant le prochain freinage et qui prend en paramètres la vitesse d'entrée dans le premier segment, la liste de segments et la décélération maximale admissible.

Q36. Ecrire en pseudocode la condition qui permet de déterminer à chaque pas de temps si un véhicule doit accélérer, décélérer, ou maintenir sa vitesse.

Q37. Pour construire un trajet, le document technique DT9 présente une fonction de la classe Graphe utilisée pour déterminer le chemin à emprunter entre 2 stations. Expliquer en quelques lignes son fonctionnement et expliquer pourquoi un bloc `try : except` est nécessaire pour entourer l'instruction :
`temps_precedent=temps_minimal_detecte[point].`
(Pour retrouver facilement, cette instruction dans le code, celle-ci est précédée par une flèche verte.)

Q38. L'algorithme étudié dans cette partie permet de calculer la trajectoire des capsules sans se soucier de la disponibilité de la voie (présence d'autres capsules). Rédiger, en quelques phrases, une stratégie permettant à l'algorithme de gérer simultanément toutes les trajectoires de toutes les capsules tout en garantissant qu'elles n'entrent pas en collision.

PARTIE 4 : SYSTEME DE POSITIONNEMENT

Lorsqu'un utilisateur réserve un trajet, une feuille de route est générée par la centrale de gestion avant d'être transmise à la capsule. Il s'agit d'un fichier texte indiquant pour différents instants du trajet une consigne de position exacte pour la capsule sur le circuit. La capsule suit en permanence sa position sur la boucle, gère sa vitesse pour correspondre aux consignes de la feuille de route et transmet régulièrement un point de suivi du trajet au serveur. La gestion du flux est ainsi totalement automatisée.

Pour que les capsules puissent respecter en temps réel ces feuilles de route sur le réseau Urbanloop, il est nécessaire de connaître la position de chaque capsule sur les boucles de transport

L'ordre de grandeur de l'échantillonnage de la feuille de route est la seconde.

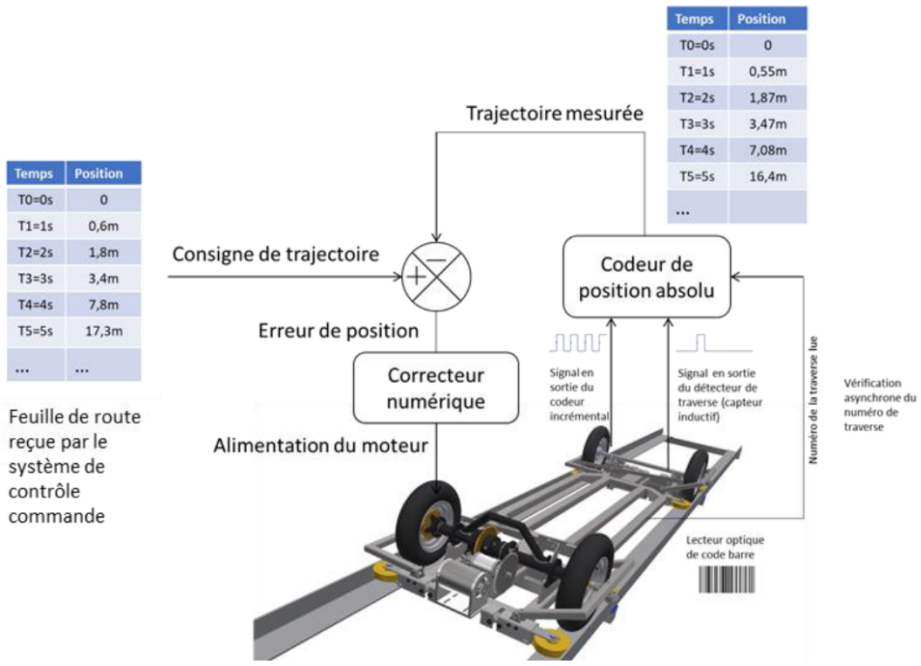


Figure 15 : principe du système de suivi de la feuille de route

La vitesse moyenne d'une capsule est de l'ordre de 60 km/h, temps d'embarquement et débarquement pris en compte. La vitesse peut varier d'environ 25 km/h lors de l'insertion de la capsule dans la boucle ou dans les virages à une vitesse de pointe pouvant atteindre jusqu'à 75 km/h. Un trajet type est de l'ordre de 5 km, la longueur moyenne des boucles du réseau est de 300 m.

L'objectif de cette partie est d'étudier le système de suivi de la vitesse et de la position absolue de chaque capsule sur le circuit.

1. Etude de la mesure de la vitesse de la capsule

Compte tenu de la configuration de l'essieu et des roues de la capsule, le choix s'est porté sur l'utilisation d'un capteur à effet Hall et d'un aimant positionné sur le système d'entrainement des roues pour compter les tours de roues (figure 15). Chaque

passage de l'aimant devant le capteur correspond à un tour de roue. Le diamètre des roues de la capsule est de 40 cm.

La carte d'acquisition et de traitement des signaux a été développée autour d'un microcontrôleur MSP430F6989 16 bits de Texas Instrument.

Q39. Calculer les fréquences du signal de sortie du capteur à effet Hall pour les vitesses minimale, moyenne et maximale de la capsule.

La documentation du capteur à effet Hall indique une sortie de type tout ou rien qui passe au niveau bas en présence d'un champ magnétique, le niveau de repos étant le niveau haut.

La mesure de vitesse est réalisée par un Timer configuré en mode capture d'événements qui permet d'obtenir le temps écoulé entre deux passages de l'aimant devant le capteur.

Q40. En utilisant le document technique DT10 donnant un extrait de la documentation du Timer A du MSP340FR6989, expliquer le principe de la mesure de vitesse de la capsule.

Le système d'horloge du MSP430 propose différentes voies d'horloges pour le Timer, une horloge externe ACLK ou l'horloge interne du microcontrôleur SMCLK. L'horloge externe est un quartz de fréquence 32,768 kHz. La fréquence de l'horloge interne est configurable entre 1 MHz et 24 MHz.

Q41. Compte-tenu de la fréquence des événements à capturer, proposer un choix d'horloge pour la capture du changement d'état du capteur à effet Hall.

Le capteur à effet Hall est connecté sur une broche externe du microcontrôleur capable de déclencher un événement de capture dans le registre CCIA0 du Timer_A0. Le document technique DT11 donne des extraits de la librairie Driverlib utilisée pour la configuration des MSP430 en langage C.

Q42. Compléter sur le document réponse DR7 la fonction d'initialisation du Timer en mode de comptage continu et capture d'événement synchrone.

Le Timer 16 bits est configuré en mode de comptage continu. La capture d'un événement dans le registre CCIA0 peut intervenir à n'importe quel moment de la séquence de comptage du Timer avec une périodicité qui change en fonction de la vitesse de la capsule.

Q43. D'après le document technique DT12, donner les lignes de code qui permettent de calculer le temps entre deux passages de l'aimant devant le capteur à effet Hall.

Q44. Expliquer et justifier le fait que cette formule de calcul soit applicable dans tous les cas y compris si le timer atteint la valeur maximum du compteur et repart à zéro entre deux captures.

2. Suivi de la position absolue de la capsule sur la boucle

La gestion automatisée du flux de circulation des capsules sur les boucles de transport nécessite le suivi précis de la position absolue de chaque capsule. Lorsque

la capsule circule à haute vitesse, des glissements se produisent sur les rails de guidage ce qui rend la seule mesure de la vitesse de rotation des roues insuffisante pour un suivi précis de la position absolue.

Une mesure de position faite par des éléments extérieurs aux capsules étant exclue, le choix s'est porté sur le comptage des traverses sur lesquelles se déplace la capsule. Chaque traverse est équipée d'un code-barres, qui lui confère une identification unique. Le système de suivi de position allie la mesure de la vitesse de rotation des roues et le comptage des traverses qui permet de resynchroniser périodiquement la position effective de la capsule. La lecture des codes-barres des traverses assure qu'il n'y a pas d'erreur de comptage de celles-ci.

Le code-barres équipant chaque traverse utilise le code 128 qui permet d'encoder jusqu'à 128 caractères au format ASCII. Le choix s'est porté sur une suite de 7 caractères, 3 lettres et 4 chiffres permettant un nombre de codes différents suffisamment élevé.

Chaque capsule est munie d'un lecteur de codes-barres qui transmet le code à la carte d'acquisition au moyen d'une liaison RS232 configurée à 9 600 bauds, sur l'horloge SMCLK du microcontrôleur, cadencée à 8 MHz.

Q45. Le document technique DT12 donne la fonction d'initialisation de la liaison UART (RS232) et le document technique DT10 un extrait de la documentation du microcontrôleur MSP430FR6989. Montrer que la vitesse de transmission est établie à 9 600 bauds.

La configuration de la liaison série est 1 bit de start, 8 bits de données LSB en premier, pas de parité, 1 bit de stop. Le lecteur impose un IDLE de repos de 4,5 ms entre chaque caractère transmis et termine la transmission par le caractère 0x0D.

Q46. Calculer le temps de transmission d'un code-barres à la carte d'acquisition.

Les trois informations, mesure de la vitesse/position, comptage des traverses, lecture du code-barres des traverses sont obtenues sur le microcontrôleur par capture d'événements extérieurs. Chaque capture génère une interruption indépendante.

Q47. Rappeler le principe général d'une interruption sur un microcontrôleur. Donner le nom des 3 vecteurs et des 3 routines d'interruption utilisés dans le cadre du suivi de la position sur le microcontrôleur MSP430FR6989 et préciser quel évènement déclenche l'interruption (le programme est donné document technique DT12).

La gestion des interruptions nécessite la définition d'une table des vecteurs d'interruption définissant un ordre de priorité en cas de déclenchement simultané. Cette table est donnée en DT13. Le microcontrôleur est cadencé par l'horloge MCLK configurée à 8 MHz. L'exécution d'une instruction du microcontrôleur utilise au maximum 6 cycles d'horloges.

Q48. Evaluer le risque de déclenchement simultané de plusieurs interruptions et montrer que l'acquisition de l'ensemble des données de suivi de la position sera au maximum de l'ordre de la milliseconde.

3. Synthèse, validation du suivi de la position

Les positions des traverses données en centimètres sont stockées dans le tableau d'entiers nommé ***synchro*** et les codes-barres associés sont stockés dans le tableau de caractères ***traverse_bc***. Ces deux tableaux sont dans le même ordre et leur indice correspond au numéro de la traverse.

Q49. Proposer le code de la fonction *unsigned int synchro_barcode(static char *barcode)* qui permet de resynchroniser le numéro de traverse avec le dernier code-barres lu.

Q50. La détection du code-barres de chaque traverse est l'opération la plus longue. Justifier la stratégie utilisée dans la fonction `main` et la gestion des interruptions pour minimiser le temps de mise à jour de la position de la capsule (voir DT12).

Q51. A partir des questions précédentes, vérifier que le temps d'acquisition d'une position est compatible avec la fréquence de mise à jour des données des capteurs. On supposera que le temps maximum de parcours du tableau de code-barres ne dépasse pas la milliseconde.

Q52. Conclure quant au suivi en temps réel la position absolue de la capsule et à la possibilité de transmettre un certificat de suivi avec une fréquence de 2 Hz à la centrale de gestion.

Document technique DT1. INFORMATION CAPACITY AND VERSIONS OF THE QR CODE

Error Correction Feature

For advanced users

Back to top page
"What is a QR Code?"

QR Code has error correction capability to restore data if the code is dirty or damaged. Four error correction levels are available for users to choose according to the operating environment. Raising this level improves error correction capability but also increases the amount of data QR Code size. To select error correction level, various factors such as the operating environment and QR Code size need to be considered. Level Q or H may be selected for factory environment where QR Code get dirty, whereas Level L may be selected for clean environment with the large amount of data. Typically, Level M (15%) is most frequently selected.

QR Code Error Correction Capability*	
Level L	Approx 7%
Level M	Approx 15%
Level Q	Approx 25%
Level H	Approx 30%

*Data restoration rate for total codewords (codeword is a unit that constructs the data area. One codeword of QR Code is equal to 8 bits.)

Error Correction Feature

The QR Code error correction feature is implemented by adding a Reed-Solomon Code*to the original data.
The error correction capability depends on the amount of data to be corrected. For example, if there are 100 codewords of QR Code to be encoded,50 of which need to be corrected, 100 codewords of Reed-Solomon Code are required, as Reed-Solomon Code requires twice the amount of codewords to be corrected. In this case, the total codewords are 200, 50 of which can be corrected. Thus, the error correction rate for the total codewords is 25%. This corresponds to QR Code error correction Level Q.

In the example above, the error correction rate for QR Code codewords can be considered as 50%. However, it is not always the case that codewords of not Reed-Solomon Code but only QR Code are susceptible to dirt and damage.QR Code therefore represents its error correction rate as a ratio of the total codewords.

*Reed-Solomon Code is a mathematical error correction method used for music CDs etc. The technology was originally developed as a measure against communication noise for artificial satellites and planetary probes. It is capable of making a correction at the byte level, and is suitable for concentrated burst errors.

Version	Modules	ECC Level	Data bits (mixed)	Numeric	Alphanumeric	Binary	Kanji
1	21x21	L	152	41	25	17	10
		M	128	34	20	14	8
		Q	104	27	16	11	7
		H	72	17	10	7	4
2	25x25	L	272	77	47	32	20
		M	224	63	38	26	16
		Q	176	48	29	20	12
		H	128	34	20	14	8
3	29x29	L	440	127	77	53	32
		M	352	101	61	42	26
		Q	272	77	47	32	20
		H	208	58	35	24	15
4	33x33	L	640	187	114	78	48
		M	512	149	90	62	38
		Q	384	111	67	46	28
		H	288	82	50	34	21
5	37x37	L	864	255	154	106	65
		M	688	202	122	84	52
		Q	496	144	87	60	37
		H	368	106	64	44	27
6	41x41	L	1,088	322	195	134	82
		M	864	255	154	106	65
		Q	608	178	108	74	45
		H	480	139	84	58	36
7	45x45	L	1,248	370	224	154	95
		M	992	293	178	122	75
		Q	704	207	125	86	53
		H	528	154	93	64	39
8	49x49	L	1,552	461	279	192	118
		M	1,232	365	221	152	93
		Q	880	259	157	108	66
		H	688	202	122	84	52

Tournez la page S.V.P.

Document technique DT2. LIBRAIRIE QRGENERATOR POUR ANDROID

Whats New in 1.0.4:

1. QR code color can be changed dynamically
2. Android X support is included
3. Minimum support from version 14 is included

Permission:

Add This Permission for saving your generated code

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

How to use this Library:

After importing this library, use the following lines to use this library. The following lines are used to generated the QR Code

```
// Initializing the QR Encoder with your value to be encoded, type you required and Dimension
QRGEncoder qrgEncoder = new QRGEncoder(inputValue, null, QRGContents.Type.TEXT, smallerDimension);
qrgEncoder.setColorBlack(Color.RED);
qrgEncoder.setColorWhite(Color.BLUE);
try {
  // Getting QR-Code as Bitmap
  bitmap = qrgEncoder.getBitmap();
  // Setting Bitmap to ImageView
  qrImage.setImageBitmap(bitmap);
} catch (WriterException e) {
  Log.v(TAG, e.toString());
}
```

Save QR Code as Image

```
// Save with location, value, bitmap returned and type of Image(JPG/PNG).
QRGSaver qrgSaver = new QRGSaver();
qrgSaver.save(savePath, edtValue.getText().toString().trim(), bitmap, QRGContents.ImageType.IMAGE_JPEG);
```

Document technique DT3. SYNTAXE DES EXPRESSIONS REGULIERES

Les métacaractères / ancres

- ^ accent circonflexe
  marque le début d'une chaîne(voir classe aussi)
- $ dollar
  marque la fin d'une chaîne

Memo:
^chat$ reconnaît chat seul
^$ reconnaît chaîne vide
^ début de chaîne
$ fin de chaîne

L'alternative

- | barre verticale
  marque l'alternative

Memo:
L[y|i]s reconnaît Lys ou Lis
^(De|Sujet|Date):@ reconnaît tout ce qui commence par De:@ ou Sujet:@ ou Date:@

Les quantificateurs

- ? point d'interrogation
  Facultatif
  zéro ou une occurrence
- * étoile
  Facultatif
  zéro, une ou plusieurs occurrences
- + signe plus
  Obligatoire
  une ou plusieurs occurrences
- {x}accolade + nombre
  Obligatoire restrictif
  doit apparaître exactement x fois
- {x,}accolade + nombre
  Obligatoire non restrictif
  doit apparaître au moins x fois
- {x,y}accolade + nombre
  Obligatoire restrictif
  doit apparaître exactement x fois et maximum y fois

Memo:
a? reconnaît 0 ou 1 a
a* reconnaît 0 ou plusieurs a
a+ reconnaît 1 ou plusieurs a

Les classes de caractères

- [ ] les crochets
  indique une classe
- - le tiret
  indique l'intervalle dans une classe

Memo:
[a-z] reconnaît les lettres de a à z
[Yy]ves un mot avec ou sans majuscule
<h[1-6]> une balise de titre par exemple

La classe complémentée

- [^...] au lieu de [...]
  indique une classe complémentée
  reconnaît tout caractère qui n'est pas énuméré

Memo:
[^0-9] reconnaît tout ce qui n'est pas des chiffres
[^1-6] reconnaît tout sauf les chiffres de 1 à 6
Rappel : l'accent circonflexe est un métacaratère à l'intérieur de la classe. A l'extérieur, c'est une ancre qui signifie le début de...

Le tiret

- - indique un intervalle dans une classe [0-9]

Rappel: Le tiret est un métacaractère à l'intérieur d'une classe à condition qui exprime bien un intervalle.
Pour utiliser le tiret en tant que littéral à l'intérieur d'une classe, soit le placer au début, soit en fin de classe [-0-9] ou [0-9-]
A l'extérieur d'une classe le tiret est un caractère normal.

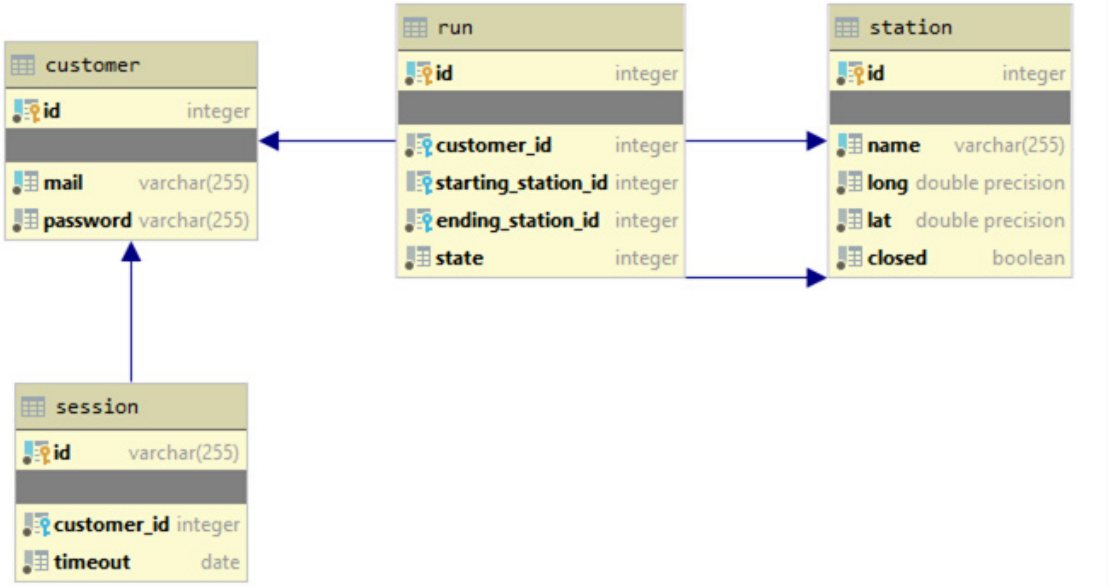
Les parenthèses

- ( ) encadrer les instructions
  capture de sous-chaînes

Rappel:les parenthèses en dehors de leur fonction d'encadrer les instructions, permettent la capture de la partie de sous-chaîne définie et satisfaisant le motif.

Tournez la page S.V.P.

Document technique DT4. SCHEMA PARTIEL DE LA BASE DE DONNEES



Document technique DT5. LISTE DES CODES ERREUR DANS LE PROTOCOLE HTTP V1.1

Informational Status Codes

100 – Continue [The server is ready to receive the rest of the request.]

101 – Switching Protocols [Client specifies that the server should use a certain protocol and the server will give this response when it is ready to switch.]

Client Request Successful

200 – OK [Success! This is what you want.]

201 – Created [Successfully created the URI specified by the client.]

202 – Accepted [Accepted for processing but the server has not finished processing it.]

203 – Non-Authoritative Information [Information in the response header did not originate from this server. Copied from another server.]

204 – No Content [Request is complete without any information being sent back in the response.]

205 – Reset Content [Client should reset the current document. Ie. A form with existing values.]

206 – Partial Content [Server has fulfilled the partial GET request for the resource. In response to a Range request from the client. Or if someone hits stop.]

Request Redirected

300 – Multiple Choices [Requested resource corresponds to a set of documents. Server sends information about each one and a URL to request them from so that the client can choose.]

301 – Moved Permanently [Requested resource does not exist on the server. A Location header is sent to the client to redirect it to the new URL. Client continues to use the new URL in future requests.]

302 – Moved Temporarily [Requested resource has temporarily moved. A Location header is sent to the client to redirect it to the new URL. Client continues to use the old URL in future requests.]

303 – See Other [The requested resource can be found in a different location indicated by the Location header, and the client should use the GET method to retrieve it.]

304 – Not Modified [Used to respond to the If-Modified-Since request header. Indicates that the requested document has not been modified since the the specified date, and the client should use a cached copy.]

305 – Use Proxy [The client should use a proxy, specified by the Location header, to retrieve the URL.]

307 – Temporary Redirect [The requested resource has been temporarily redirected to a different location. A Location header is sent to redirect the client to the new URL. The client continues to use the old URL in future requests.]

Client Request Incomplete

400 – Bad Request [The server detected a syntax error in the client's request.]

401 – Unauthorized [The request requires user authentication. The server sends the WWW-Authenticate header to indicate the authentication type and realm for the requested resource.]

402 – Payment Required [reserved for future.]

403 – Forbidden [Access to the requested resource is forbidden. The request should not be repeated by the client.]

404 – Not Found [The requested document does not exist on the server.]

405 – Method Not Allowed [The request method used by the client is unacceptable. The server sends the Allow header stating what methods are acceptable to access the requested resource.]

406 – Not Acceptable [The requested resource is not available in a format that the client can accept, based on the accept headers received by the server. If the request was not a HEAD request, the server can send Content-Language, Content-Encoding and Content-Type headers to indicate which formats are available.]

407 – Proxy Authentication Required [Unauthorized access request to a proxy server. The client must first authenticate itself with the proxy. The server sends the Proxy-Authenticate header indicating the authentication scheme and realm for the requested resource.]

408 – Request Time-Out [The client has failed to complete its request within the request timeout period used by the server. However, the client can re-request.]

409 – Conflict [The client request conflicts with another request. The server can add information about the type of conflict along with the status code.]

410 – Gone [The requested resource is permanently gone from the server.]

411 – Length Required [The client must supply a Content-Length header in its request.]

412 – Precondition Failed [When a client sends a request with one or more If... headers, the server uses this code to indicate that one or more of the conditions specified in these headers is FALSE.]

413 – Request Entity Too Large [The server refuses to process the request because its message body is too large. The server can close connection to stop the client from continuing the request.]

414 – Request-URI Too Long [The server refuses to process the request, because the specified URI is too long.]

415 – Unsupported Media Type [The server refuses to process the request, because it does not support the message body's format.]

417 – Expectation Failed [The server failed to meet the requirements of the Expect request-header.]

Server Errors

500 – Internal Server Error [A server configuration setting or an external program has caused an error.]

501 – Not Implemented [The server does not support the functionality required to fulfill the request.]

502 – Bad Gateway [The server encountered an invalid response from an upstream server or proxy.]

503 – Service Unavailable [The service is temporarily unavailable. The server can send a Retry-After header to indicate when the service may become available again.]

504 – Gateway Time-Out [The gateway or proxy has timed out.]

505 – HTTP Version Not Supported [The version of HTTP used by the client is not supported.]

Unused status codes

306- Switch Proxy

416- Requested range not satisfiable

506- Redirection failed

HTTP protocol version 1.1 Server Response Codes

Tournez la page S.V.P.

Document technique DT6. EXTRAITS DU MANUEL XBEE

Operation

Serial Interface

The XBee/XBee-PRO Zigbee RF Module interface to a host device through a serial port. The device can communicate with any logic and voltage compatible UART, through a level translator to any serial device (for example, through a RS-232 or USB interface board).

Serial interface protocols

The XBee/XBee-PRO Zigbee RF Module supports both Transparent and Application Programming Interface (API) serial interfaces.

The following tables compare the advantages of Transparent and API operating modes:

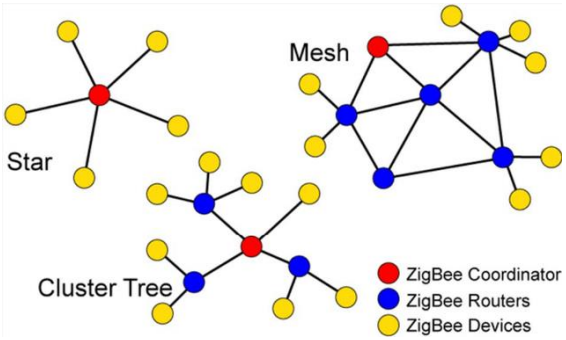
Transparent operation	
Simple interface	The device transmits all received data unless it is in Command mode.
Easy to support	It is easier for an application to support transparent operation and Command mode.
API operation	
Easy to manage data transmissions to multiple destinations	Transmitting data to multiple remotes only requires changing the address in the API frame. This process is much faster than a transparent operation. Each API transmission can return a transmit status frame the success or reason of failure.
Received data frames indicates the sender's adress	All received RF data API frames indicate source address.
Advanced networking diagnostics	API frames can provide indication of I/O samples from remote devices, and node identification messages

Zigbee networks

Zigbee networking concepts

Depending on the application requirements, an IEEE 802.15.4 LR-WPAN operates in either of two topologies: the star topology or the peer-to-peer topology. In the star topology, the communication is established between devices and a single central controller, called the PAN coordinator. The PAN coordinator is the primary controller of the PAN. All devices operating on a network of either topology have unique addresses, referred to as extended addresses.

The peer-to-peer topology also has a PAN coordinator; however, it differs from the star topology in that any device is able to communicate with any other device as long as they are in range of one another. Peer-to-peer topology allows more complex network formations to be implemented, such as Mesh networking topology or Cluster Tree.



Zigbee defines three different device types: coordinator, router, and end device. Coordinator is the IEEE 802.15.4 PAN coordinator. Router can join existing networks and send, receive, and route information, can allow other routers and devices to join the network. End device can join existing networks and send and receive information but cannot route information and cannot allow other devices to join the network.

API Operation

The following table shows the API frame structure:

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

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

Frame data: this field contains the information that a device receives or transmits. The cmdID frame (API identifier) indicates which API messages contains the cmdData frame (Identifier-specific data). The device sends multi-byte values **big endian format.**

The structure of frame data depends on the purpose of the API frame. (cf. DT7)

Calculate and verify checksums

– To calculate the checksum of an API frame:

1. Add all bytes of the packet, except the start delimiter 0x7E and the length (the second and third bytes).

2. Keep only the lowest 8 bits from the result.

3. Subtract this quantity from 0xFF.

– To verify the checksum of an API frame:

1. Add all bytes including the checksum; do not include the delimiter and length.

2. If the checksum is correct, the last two digits on the far right of the sum equal 0xFF.

Tournez la page S.V.P.

Node Identification Indicator – 0x95

This frame type is emitted when a node identification broadcast is received or when a node is joining a network. The node identification indicator contains information about the identifying device, such as address, identifier string (NI), and other relevant data.

Format of the specific data frame (common API frame bytes are described in DT6)

API identifier (cmdID) – 0x95
Identifier specific data (cmdData):

- The sender's 64-bit address (8bytes)
- The sender's 16-bit network address (2bytes)
- Options: 0x02 = broadcast, 0x04 = broadcast over all PAN, other = Digimesh reserved (1byte)
- The 16-bit network address of the device that sent the Node Identification (same as sender for a network joining) (2bytes)
- The 64-bit address of the device that sent the Node Identification (same as sender for a network joining) (8bytes)
- Node identification string (NI), ASCII string (2bytes minimum), the string is terminated with a NULL byte (0x00)
- Indicates the 16-bit address of the remote's parent or 0xFFFE if the remote has no parent (2bytes)
- Type of network device (1byte): 0 = coordinator, 1 = router, 2 = end device
- The event that caused the node identification broadcast to be sent (1byte)
- Profile ID: 0xC105 for Digi modules
- Manufacturer ID: 0x101E for Digi modules

Transmit Request Frame – 0x10

This frame type is used to send payload data as an RF packet to a specific destination.

Format of the specific data frame (common API frame bytes are described in DT6)

API identifier (cmdID) – 0x10
Identifier specific data (cmdData):

- Frame ID (1byte): 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.
- The 64-bit address of the destination device (8bytes)
- The 16-bit network address of the destination device (2bytes)
- Broadcast radius (1byte): sets the maximum number of hops for a broadcast transmission. 0 = uses the preconfigured broadcast radius (recommended).
- Transmit options (1bytet): 0x01 = disable Ack, 0x04 = indirect transmission, 0x08 = multicast transmission.
- Payload data: Data to be sent to the destination device. Maximum 0xFFbytes in big-endian format

Addressing information :

64-bit addressing: For broadcast transmissions, set the 64-bit destination address to 0x000000000000FFFF. If transmitting to a 64-bit destination, set the 16-bit address field to 0xFFFE

16-bit addressing: to use 16-bit addressing, set the 64-bit address field to 0xFFFFFFFFFFFFFFFF

Addressing a Zigbee coordinator: set the 64-bit address to all 0x00s and the 16-bit address to 0xFFFE

Receive Packet Frame – 0x90

This frame type is emitted when a device configured with standard API output receives an RF data packet. Typically, this frame is emitted as a result of a device on the network sending serial data using the Transmit Request - 0x10.

Format of the specific data frame (common API frame bytes are described in DT6)

API identifier (cmdID) – 0x90
Identifier specific data (cmdData):

- The sender's 64-bit address. (8bytes)
- The sender's 16-bit network address (2bytes)
- Receive options : (1byte) :
- Transmit options (1byte): 0x01 = packet was Ack, 0x02 = Packet was sent as a broadcast, 0x10 =packet was sent across a secure session, 0x40 Zigbee only, packet was sent from an End Device.
- Received data: the RF payload data that the device receives.

Extended Transmit Status – 0x8B

This frame type is emitted when a network transmission request completes (Transmit Request 0x10). The status field of this frame indicates whether the request succeeded or failed and the reason.

Format of the specific data frame (common API frame bytes are described in DT6)

API identifier (cmdID) – 0x8B
Identifier specific data (cmdData):

- Frame ID: Identifies the data frame for the host to correlate with a prior request (1byte)
- 16-bit destination address (2bytes): The 16-bit network address where the packet was delivered (if successful). If not successful, this address is 0xFFFD (destination address unknown).
- Transmit retry count (1byte): The number of application transmission retries that occur.
- Delivery status (1byte): 0x00 = success, 0x01 = MAC ACK failure, 0x15 = invalid destination endpoint, 0x24 Address not found, 0x25 = route not found, 0x74 = data payload too large….
- Discovery status: 0x00 = No discovery overhead, 0x01 = Zigbee address discovery, 0x02 = route discovery, 0x03 = Zigbee address and route discovery, 0x40 = Zigbee end device extended timeout.

Tournez la page S.V.P.

Bibliothèque ctime

```
#include time.h
```

```
time_t time (time_t* timer);     //get current time
```

Get the current calendar time as a value of type time_t.

```
char* ctime (const time_t * timer); //Convert time_t value to string
```

Interprets the value pointed by timer as a calendar time and converts it to a C-string containing a human-readable version of the corresponding time and date, in terms of local time.

```
double difftime (time_t end, time_t beginning); //Return difference between two times
```

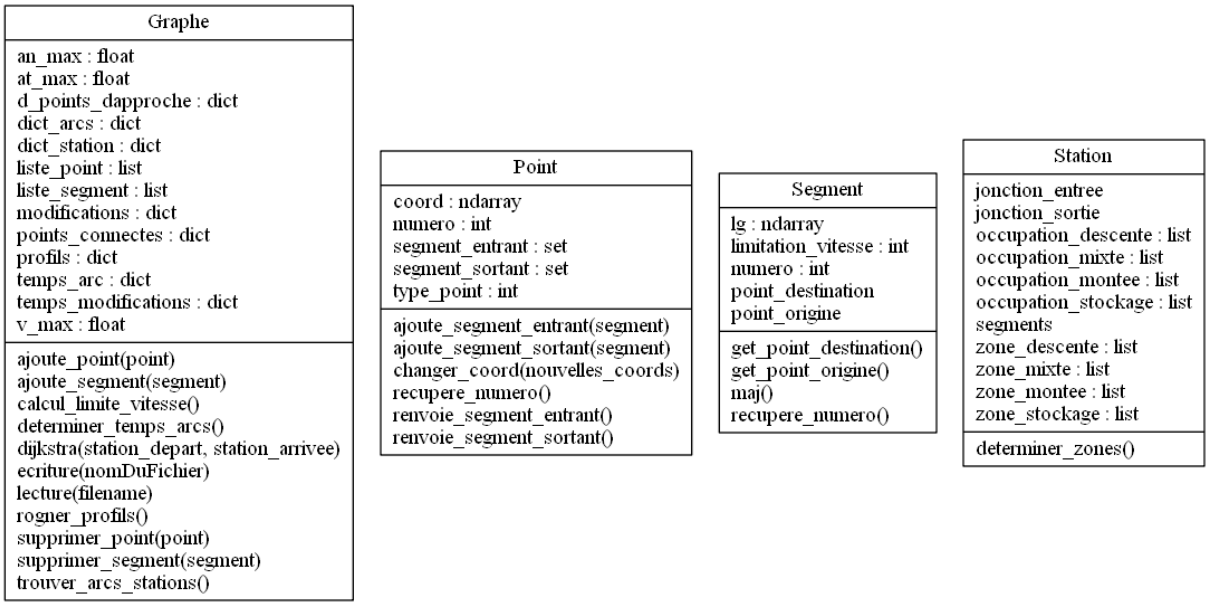
Calculates the difference in seconds between beginning and end

Fonction de mise à l'heure de l'horloge RTC

```
set_time (time_t timer);         //set the RTC clock
```

Set the RTC time to the timer value (number of seconds since January 1, 1970, the UNIX timestamp)

Document technique DT9. DIAGRAMME DE CLASSES ET FONCTIONS UTILES




```
class Segment:

    def __init__(self, point_origine, point_destination, *, numero=0,
limitation_vitesse=-1):
        assert isinstance(point_origine, Point)
        assert isinstance(point_destination, Point)
        self.numero = numero
        self.point_origine = point_origine
        self.point_destination = point_destination
        self.lg = np.linalg.norm(self.point_origine() -
self.point_destination())
        point_origine.ajoute_segment_sortant(self)
        point_destination.ajoute_segment_entrant(self)

        self.limitation_vitesse = limitation_vitesse

    def __repr__(self):
        return '''S {} {} {} {}'''.format(self.numero,
self.point_origine.recupere_numero(),

self.point_destination.recupere_numero(), self.limitation_vitesse)

    def get_point_origine(self):
        return self.point_origine

    def get_point_destination(self):
        return self.point_destination

    def recupere_numero(self):
        return self.numero

    def maj(self):
        coord_origine = self.point_origine()
        coord_destination = self.point_destination()
        self.lg = np.linalg.norm(coord_origine - coord_destination)
```


Tournez la page S.V.P.

```
def dijkstra(self,station_depart,station_arrivee):
    point_depart=station_depart.jonction_sortie
    point_destination=station_arrivee.jonction_entree
    temps_minimal_detecte={point_depart:0}
    point_suivant=self.points_connectes[point_depart][0]

temps_minimal_detecte[point_suivant]=self.temps_arc[(point_depart,point_sui
vant)]

trajets_a_continuer=[([point_depart,point_suivant],temps_minimal_detecte[po
int_suivant])]
    while trajets_a_continuer[0][0][-1]!=point_destination:
        trajet_etudie=trajets_a_continuer[0][0]
        temps_trajet_etudie=trajets_a_continuer[0][1]
        for point in self.points_connectes[trajet_etudie[0][-1]]:
            PlusCourt=False
            temps=temps_trajet_etudie+self.temps_arc[(trajet_etudie[-
1],point)]+self.temps_modifications[((trajet_etudie[-2],trajet_etudie[-
1]),(trajet_etudie[-1],point))]
            try:
                temps_precedent=temps_minimal_detecte[point]
            except KeyError:
                PlusCourt=True
            else:
                if temps<temps_precedent:
                    PlusCourt=True
            if PlusCourt:
                temps_minimal_detecte[point]=temps
                for i in range(len(trajets_a_continuer)):
                    if trajets_a_continuer[i][1] > temps:
                        trajets_a_continuer = trajets_a_continuer[:i] +
[(trajet_etudie + [point], temps)] + trajets_a_continuer[i:]
                        break
        trajets_a_continuer=trajets_a_continuer[1:]
    return trajets_a_continuer[0]
```

Timer A

Timer_A is a 16-bit timer/counter with up to seven capture/compare registers. Timer_A can support multiple capture/compares, PWM outputs, and interval timing. Timer_A also has extensive interrupt capabilities. Interrupts may be generated from the counter on overflow conditions and from each of the capture/compare registers.

Operation

The 16-bit timer/counter register TAxR increments or decrements (depending on mode of operation) with each rising edge of the clock signal. TAxR can be read or written with software. Additionally, the timer can generate an interrupt when it overflows. TAR may be cleared by setting the TACLR bit.

Clock Source Select and Divider

The timer clock can be sourced from ACLK (32 768 Hz), SMCLK, or externally from TAxCLK or INCLK. The clock source is selected with the TASSEL bits. The selected clock source may be passed directly to the timer or divided by 2, 4 or 8, using ID bits.

Timer Mode Control

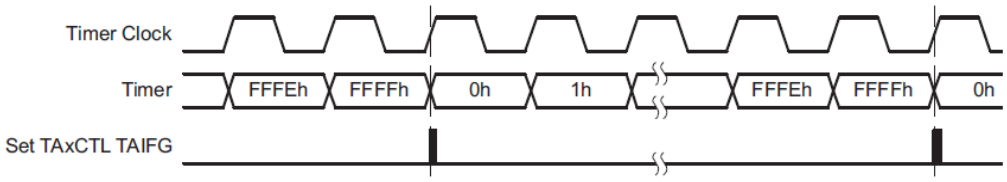
The timer has four modes of operation: stop, up, continuous, and up/down. The operating mode is selected with the MC bits.

MC	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of TAxCCR0
10	Continuous	The timer repeatedly counts from zero to 0FFFFh.
11	Up/down	The timer repeatedly counts from zero up to the value of TAxCCR0 and back down to zero.

Continuous Mode

In the continuous mode, the timer repeatedly counts up to 0FFFFh and restarts from zero. The capture/compare register TAxCCR0 works the same way as the other capture/compare registers.

The TAIFG interrupt flag is set when the timer counts from 0FFFFh to zero.



Continuous Mode Flag Setting

Capture/Compare Blocks

Up to seven identical capture/compare blocks, TAxCCRn (where n = 0 to 6), are present in Timer_A. Any of the blocks may be used to capture the timer data or to generate time intervals.

Tournez la page S.V.P.

Capture Mode
The capture mode is selected when CAP = 1. Capture mode is used to record time events. It can be used for speed computations or time measurements. The capture inputs CCIxA and CCIxB are connected to external pins or internal signals and are selected with the CCIS bits. The CM bits select the capture edge of the input signal as rising, falling or both. A capture occurs on the selected edge of the input signal. If a capture occurs:

- The timer value is copied into the TAxCCRn register.
- The interrupt flag CCIFG is set.

The input signal level can be read at any time from the CCI bit.

The capture signal can be asynchronous to the timer clock and cause a race condition. Setting the SCS bit synchronizes the capture with the next timer clock. Setting the SCS bit to synchronize the capture signal with the timer clock is recommended.

Enhanced Universal Serial Communication Interface – UART Mode

In UART mode, the eUSCI_A transmits and receives characters at a bit rate asynchronous to another device. Timing for each character is based on the selected baud rate of the eUSCI_A. The transmit and receive functions use the same baud-rate frequency.

Character format
The UART character format consists of a start bit, seven or eight data bits, an even/odd/no parity bit and one or two stop bits. The UCMSB bit controls the direction of the transfer and selects LSB or MSB first. LSB first is typically required for UART communication.

UART Baud-Rate Generation
The eUSCI_A baud-rate generator is capable of producing standard baud rates from nonstandard source frequencies. It provides two modes of operation selected by the UCOS16 bit.
For a given BRCLK clock source, the baud rate used determines the required division factor N: N = f_{BRCLK}/baud rate
The division factor N is often a noninteger value, thus, at least one divider and one modulator stage is used to meet the factor as closely as possible.
If N is equal or greater than 16, TI recommends using the oversampling baud-rate generation mode by setting UCOS16.

To calculate the correct settings for the baud-rate generation, perform these steps:

1. Calculate N = f_{BRCLK}/baud rate [if N > 16 continue with step 3, otherwise with step 2]

2. OS16 = 0, clockPrescalar = INT(N) [continue with step 4]

3. OS16 = 1, clockPrescalar = INT(N/16), firstModReg = INT([(N/16) – INT(N/16)] × 16)

4. secondModReg (UCBRSx) depends on fractional part of N, f(N) = N - INT(N). f(N)=0,3000 UCBRSx=0x25, f(N)=0,3335, UCBRSx=0x49, f(N)=0,3575, UCBRSx=0x4A

Document technique DT11. EXTRAITS DE LA LIBRAIRIE DRIVERLIB

Timer_A_initContinuousMode()

```
void Timer_A_initContinuousMode (uint16_t baseAddress,
                Timer_A_initContinuousModeParam * param)
```

Configures Timer_A in continuous mode.

Parameters

baseAddress	is the base address of the TIMER_A module.
param	is the pointer to struct for continuous mode initialization.

Timer_A_initCaptureMode()

```
void Timer_A_initCaptureMode (uint16_t baseAddress,
                Timer_A_initCaptureModeParam * param)
```

Initializes Capture Mode.

Parameters

baseAddress	is the base address of the TIMER_A module
param	is the pointer to struct for capture mode initialization

Timer_A_initContinuousModeParam Struct Reference

Used in the `Timer_A_initContinuousMode()` function as the param parameter.

Data Fields

uint16_t clockSource
uint16_t clockSourceDivider
uint16_t timerInterruptEnable_TAIE
uint16_t timerClear
bool startTimer w*hether to start the timer immediately.*

clockSource

```
uint16_t Timer_A_initContinuousModeParam::clockSource
```

Selects Clock source.

Valid values are:
TIMER_A_CLOCKSOURCE_EXTERNAL_TXCLK [Default]
TIMER_A_CLOCKSOURCE_ACLK
TIMER_A_CLOCKSOURCE_SMCLK
TIMER_A_CLOCKSOURCE_INVERTED_EXTERNAL_TXCLK

Tournez la page S.V.P.

Timer_A_initCaptureModeParam Struct Reference

Used in the `Timer_A_initCaptureMode()` function as the param parameter.

captureInputSelect

```
uint16_t Timer_A_initCaptureModeParam::captureInputSelect
```

Decides the Input Select

Valid values are:
TIMER_A_CAPTURE_INPUTSELECT_CCIxA
TIMER_A_CAPTURE_INPUTSELECT_CCIxB
TIMER_A_CAPTURE_INPUTSELECT_GND
TIMER_A_CAPTURE_INPUTSELECT_Vcc

captureMode

```
uint16_t Timer_A_initCaptureModeParam::captureMode
```

Is the capture mode selected.

Valid values are:
TIMER_A_CAPTUREMODE_NO_CAPTURE [Default]
TIMER_A_CAPTUREMODE_RISING_EDGE
TIMER_A_CAPTUREMODE_FALLING_EDGE
TIMER_A_CAPTUREMODE_RISING_AND_FALLING_EDGE

captureRegister

```
uint16_t Timer_A_initCaptureModeParam::captureRegister
```

Selects the Capture register being used. Refer to datasheet to ensure the device has the capture compare register being used.

Valid values are:
TIMER_A_CAPTURECOMPARE_REGISTER_0
TIMER_A_CAPTURECOMPARE_REGISTER_1
TIMER_A_CAPTURECOMPARE_REGISTER_2
TIMER_A_CAPTURECOMPARE_REGISTER_3
TIMER_A_CAPTURECOMPARE_REGISTER_4
TIMER_A_CAPTURECOMPARE_REGISTER_5
TIMER_A_CAPTURECOMPARE_REGISTER_6

synchronizeCaptureSource

```
uint16_t Timer_A_initCaptureModeParam::synchronizeCaptureSource
```

Decides if capture source should be synchronized with timer clock

Valid values are:
TIMER_A_CAPTURE_ASYNCHRONOUS [Default]
TIMER_A_CAPTURE_SYNCHRONOUS

Document technique DT12. PROGRAMME DE SUIVI DE LA POSITION DE LA CAPSULE

Variables globales

```
1     #define perimetre 126
2     #define timerfreq 32768
3     unsigned int tempscapture=0,occurrence_hall=0,
4        traverse=0,endcode=0,transmission=0;
5     unsigned long position=0;
6     unsigned char barcode[];
```

Extraits de la fonction main

```
7     int main(void) {
8         WDT_A_hold(WDT_A_BASE);           // stop Watchdog

9      //Peripherals Init
10        initGPIO();
11        initUART0();
12        initTimerA();

13    //calcul de la vitesse et de la position
14    float vitesse=0;
15    volatile unsigned int i;
16      while(1){
17         if(endcode==1){
18            if(strcmp(barcode,traverse_bc[traverse])!=0){
19               traverse=barcode_synchro(barcode);
20               position=synchro[traverse];
21            }
22            endcode=0;
23         }
24         else{
25            if(occurrence_hall!=0){
26               vitesse = perimetre*
27                 ((float)timerfreq / (float)tempscapture);
28               position += perimetre;
29            }
30         }
31         if(transmission==1){
32            transmission=0;
33            EUSCI_A_UART_transmitData(EUSCI_A1_BASE, position);
34            EUSCI_A_UART_transmitData(EUSCI_A1_BASE, position>>8);
35         }
36         //low power mode 0 et global interrupt enable
37         __bis_SR_register(LPM0_bits | GIE);
38      }
39      return (0);
40    }
```

Interruption capture du Timer

```
41    #pragma vector = TIMER0_A0_VECTOR
42    __interrupt void Timer0_A0_ISR(void)
43    {
44       static unsigned int timervalue=0;
```


Tournez la page S.V.P.

```
45        tempscapture=Timer_A_getCaptureCompareCount(TIMER_A0_BASE,
46            TIMER_A_CAPTURECOMPARE_REGISTER_0)-timervalue;
47        timervalue=Timer_A_getCaptureCompareCount(TIMER_A0_BASE,
48           TIMER_A_CAPTURECOMPARE_REGISTER_0);
49        occurence_hall++;
50        __bic_SR_register_on_exit(LPM0_bits);    // Exit LPM0(Wakeup)
51    }
```

Interruption UART

```
52    #pragma vector=USCI_A0_VECTOR
53    __interrupt void USCI_A0_ISR(void)
54    {
55       unsigned char temp;
56       static int i=0;
57       temp=EUSCI_A_UART_receiveData(EUSCI_A0_BASE);
58       barcode[i]=temp;
59       i++;
60       if(temp==0x0D){
61          barcode[i-1]='\0';
62          i=0;
63          endcode=1;
64          __bic_SR_register_on_exit(LPM0_bits);      // Exit LPM0
65       }
66    }
```

Interruption GPIO capteur inductif

```
67    #pragma vector=PORT1_VECTOR
68    __interrupt void Port_1_ISR(void)
69    {
70       static unsigned int traverse=0;
71       traverse++;
72       position=synchro[traverse];
73       GPIO_clearInterrupt(GPIO_PORT_P1,GPIO_PIN1);
74    }
```

Initialisation UART

```
75    void initUART0 (void){
76     EUSCI_A_UART_initParam Uart0param = {0};
77     Uart0param.selectClockSource = EUSCI_A_UART_CLOCKSOURCE_SMCLK;
78     Uart0param.clockPrescalar = 52;
79     Uart0param.firstModReg = 1;
80     Uart0param.secondModReg = 0x49;
81     Uart0param.overSampling =
82          EUSCI_A_UART_OVERSAMPLING_BAUDRATE_GENERATION;

83     if (STATUS_FAIL == EUSCI_A_UART_init(EUSCI_A0_BASE,
84          &Uart0param)) {
85            return;
86     }

87     EUSCI_A_UART_enable(EUSCI_A0_BASE);
88     EUSCI_A_UART_clearInterrupt(EUSCI_A0_BASE,
89        EUSCI_A_UART_RECEIVE_INTERRUPT);
90     EUSCI_A_UART_enableInterrupt(EUSCI_A0_BASE,
91        EUSCI_A_UART_RECEIVE_INTERRUPT);
92    }
```

Document technique DT13. TABLES

TABLE DES VECTEURS ET FLAG D'INTERRUPTION PAR ORDRE DE PRIORITE SUR LE MSP430FR6989

INTERRUPT SOURCE	INTERRUPT FLAG	SYSTEM INTERRUPT	PRIORITY
System Reset	SVSHIFG, WDTIFG, WDTPW, PMMBORIFG	Reset	Highest
System NMI	VMAIFG, JMBINIFG …	Non maskable	
User NMI	NMIIFG, OFIFG	Non maskable	
Watchdog Timer	WDTIFG	Maskable	
eUSCI_A0 receive or transmit	UCRXIFG, UCTXIFG, UCSTTIFG, UCTXCPTIFG	Maskable	
Timer_A TA0	TA0CCR0.CCIFG	Maskable	
Timer_A TA0	TA0CCR1.CCIFG, TA0CTL.TAIFG	Maskable	
eUSCI_A1 receive or transmit	UCRXIFG, UCTXIFG, UCSTTIFG, UCTXCPTIFG	Maskable	
I/O Port P1	P1IFG.0 to P1IFG.7	Maskable	Lowest

TABLE DES CARATERES ASCII

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

asciichars.com

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR1 à DR3

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR1. CODES DE L'APPLICATION SMARTPHONE (Q2 et Q4)

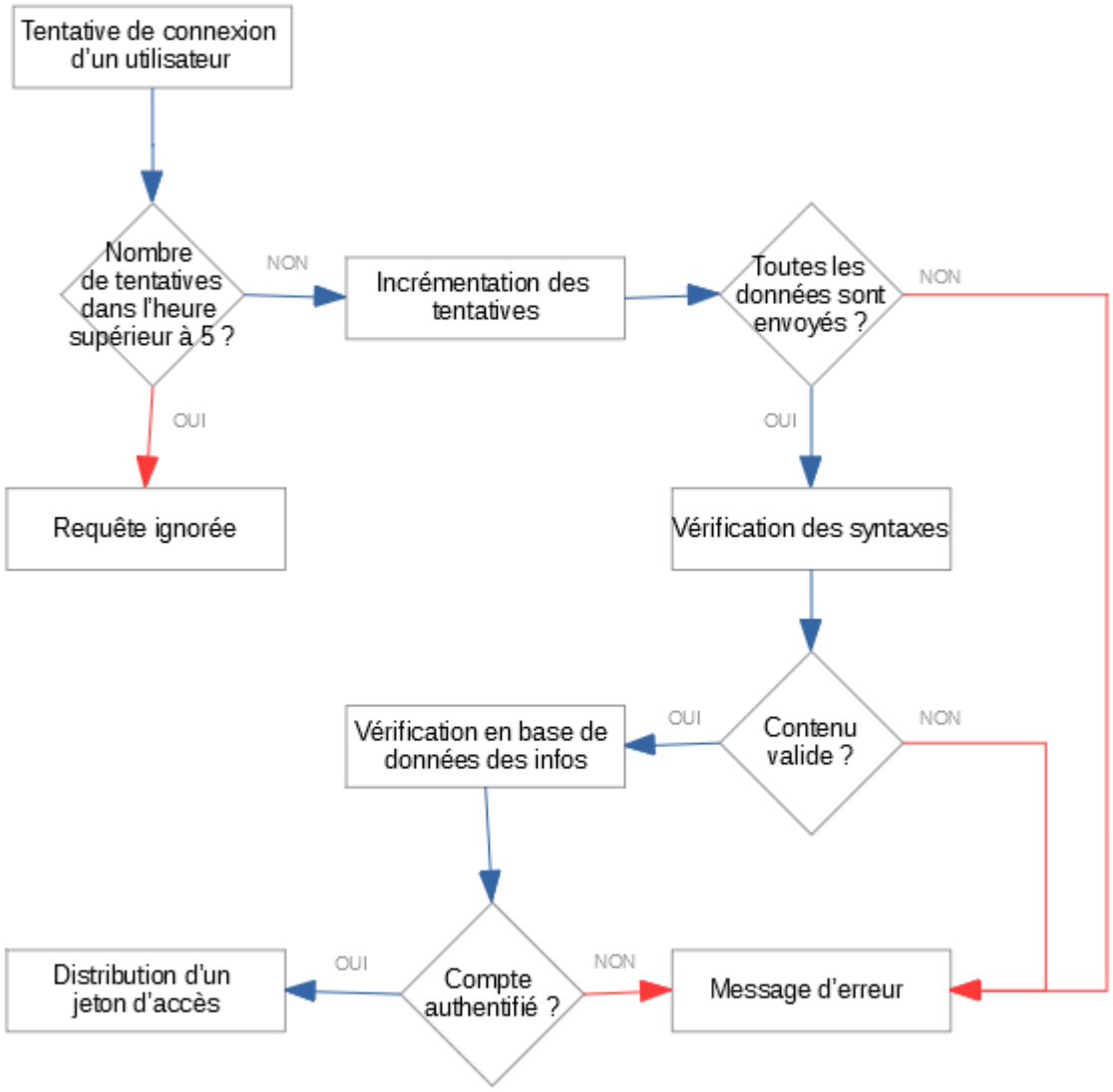
Question Q2. Implémenter la méthode statique permettant de générer le QR Code.

```
public class QrCodeUtils {
   @Nullable
   public static Bitmap generateQrCode(String data, int imageSize)
throws WriterException {









   }
}
```

Question Q4. Compléter chaque assertion du test unitaire afin de valider le bon fonctionnement de l'expression régulière.

```
public class MailUtilsTest {
    @Test
    public void isValid() {
        assertEquals(       ,
MailUtils.isValid("dupont.robert@urbanloop.fr"));
        assertEquals(_________, MailUtils.isValid("urbanloop@univ-
lorraine.fr"));
        assertEquals(_________, MailUtils.isValid("ne-pas-
répondre@ubanloop.fr"));
        assertEquals(_________,
MailUtils.isValid("Alain.Genieur@prod.urbanloop.com"));
        assertEquals(_________,
MailUtils.isValid("capsule45217854@54000.urbanloop"));
        assertEquals(_________,
MailUtils.isValid("jdoe@urbanloop..fr"));
    }
}
```

Document réponse DR2. ALGORITHME CONTRE LES ATTAQUES DE FORCE BRUTE (Q5)



Tournez la page S.V.P.

Document réponse DR3. TRANSMISSION SUR LE RESEAU ZIGBEE (Q12 ET Q15)

Q12. Compléter les informations relatives à chacun des nœuds.

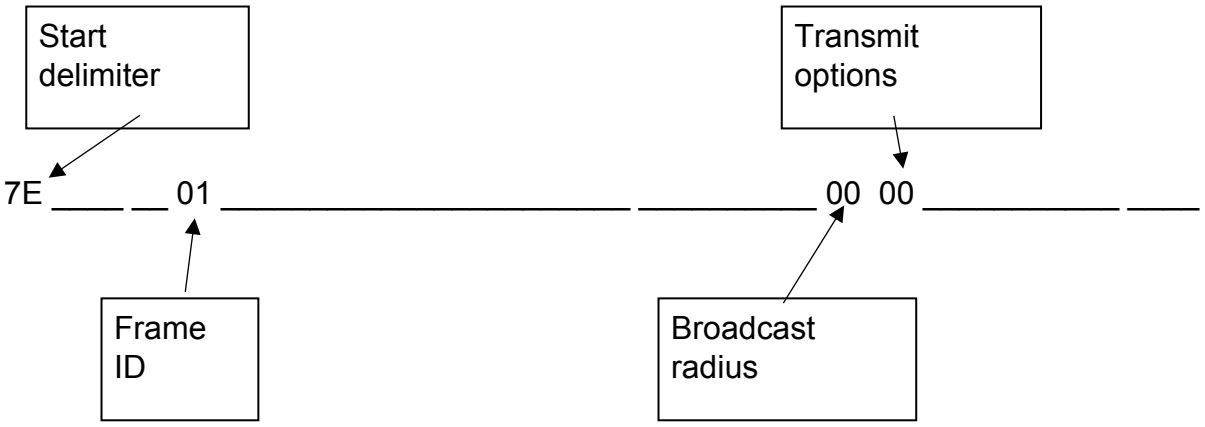
Rappel des trames

Trame Node Identification Indicator nœud 1 (notation hexadécimale) :
7E 00 23 95 00 13 A2 00 41 B1 AB 01 93 47 02 93 47 00 13 A2 00 41 B1 AB 01 43 30 30 35 00 00 00 02 02 C1 05 10 1E 3E

Trame Node Identification Indicator nœud 2 (notation hexadécimale) :
7E 00 23 95 00 13 A2 00 41 B6 3D 21 C1 A6 02 C1 A6 00 13 A2 00 41 B6 3D 21 43 30 31 34 00 00 00 02 02 C1 05 10 1E B6

Paramètre	Noeud 1	Noeud 2
Adresse 64 bits (hexadecimal)		
Adresse réseau 16 bits (hexadecimal)		
Identifiant du noeud (Node identifier) (caractères)		

Q15. Compléter la trame de Transmit Request (notation hexadécimale)



Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR4 à DR6

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR4. FONCTION C++ A COMPLETER (Q18)

La fonction `receive_cb` est exécutée en cas de réception d'une trame Receive Packet, à compléter

La variable data contient uniquement le champ « received data » de la trame Receive Packet.

Le tableau `uint16_t feuille_route[600][2] ;` permettant de stocker la feuille de route est déclaré en variable globale.

```
static void receive_cb(const RemoteXBeeZB& remote, bool broadcast,
const uint8_t *const data, uint16_t len)
{
    uint32_t heure = 0;

    static uint8_t indice=0;









}
```

Document réponse DR5. INITIALISATION D'UN GRAPHE (Q27)

```
def lecture(self, filename):

        document = list()
        with open(filename) as f:
            for line in f:
                line = line.strip('\n')
                element = line.split(' ')
                document.append(element)
        for element in document:
            if element[0] == 'N':
                numero = int(_____________) # Compléter ici
                coordx = int(_____________) # Compléter ici
                coordy = int(_____________) # Compléter ici
                type_point = int(_____________) # Compléter ici
                point = Point((coordx, coordy),
type_point=type_point, numero=numero)
                self.ajoute_point(point)

        dict_point = {i.numero: i for i in self.liste_point}
        for element in document:
            if element[0] == 'S':
                numero = int(element[1])
                numero_noeud_origine = int(element[2])
                numero_noeud_destination = int(element[3])
                noeud_origine = dict_point[numero_noeud_origine]
                noeud_destination =
dict_point[numero_noeud_destination]
                limitation_vitesse = float(element[4])
                if np.allclose(limitation_vitesse, -1):
                    limitation_vitesse = self.v_max
                s = Segment(_________________,_________________ ,______
_____________________________) # Compléter ici

                self.ajoute_segment(s)
            elif element[0] == 'R':
                self.v_max = int(element[1])
                self.at_max = int(element[2])
                self.an_max = int(element[3])
```


Tournez la page S.V.P.

Document Réponse DR6. CALCUL DES TRAJECTOIRES (Q28 ET Q35)

Q28. Vecteur d'état

```
def suivant(etat,delta_t,acceleration,vitesseMax):
    distanceParcourue=etat[0]
    vitesse=etat[1]

etat=[_____________________________________________________________________
,_____________________________________________]

    return etat
```

Q35. Calcul de la distance à parcourir avant le prochain freinage

```
def DistanceProchainFreinage(vitesseEntree,listeSegments,gamma):
    """retourne la distance du prochain freinage par rapport au
point d'entrée dans le segment 0"""
    distance_cumulee=0
    resultat=float('inf')
    for i in range(1, len(listeSegments)):
        vitesse_cible=listeSegments[i][1]









    return resultat

listeSegments=[(100,20),(50,18),(20,10),(100,20),(50,18),(20,10)]
#liste des segments devant le véhicule
#segment 0: longueur 100m, vitesse maximale:20 m/s
#segment 1: longueur 50m, vitesse maximale:18 m/s
#segment 2: longueur 20m, vitesse maximale:10 m/s
#...
gamma=-1.35 #décélaration maximale admissible en m/s²
vitesseEntree=listeSegments[0][1] #vitesse d'entrée du véhicule sur
le segment 0 en m/s

print("Le prochain freinage aura lieu à une distance de
",int(DistanceProchainFreinage(vitesseEntree,listeSegments,gamma)),"
m après l'entrée dans le segment 0 ")
```

Resultat dans la console : « le Prochain freinage aura lieu 38 m après l'entrée dans le segment 0»

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR7

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR7. CONFIGURATION DU TIMER A0 (Q42)

```
// Intialisation du Timer en mode continu
Timer_A_initContinuousModeParam Modeparam = {0};
Modeparam.clockSource = _____________________________;
Modeparam.clockSourceDivider = TIMER_A_CLOCKSOURCE_DIVIDER_1;
Modeparam.timerInterruptEnable_TAIE =
TIMER_A_TAIE_INTERRUPT_DISABLE;
Modeparam.timerClear = TIMER_A_DO_CLEAR;
Modeparam.startTimer = false;
Timer_A_initContinuousMode(TIMER_A0_BASE, &Modeparam);

Timer_A_clearCaptureCompareInterrupt(TIMER_A0_BASE,
    TIMER_A_CAPTURECOMPARE_REGISTER_0);
Timer_A_initCaptureModeParam CompParam = {0};
CompParam.captureRegister = ____________________________________;
CompParam.captureMode = _____________________________;
CompParam.captureInputSelect = ________________________________;
CompParam.synchronizeCaptureSource = ____________________________;
CompParam.captureInterruptEnable =
TIMER_A_CAPTURECOMPARE_INTERRUPT_ENABLE;
CompParam.captureOutputMode = TIMER_A_OUTPUTMODE_OUTBITVALUE;
Timer_A_initCaptureMode(TIMER_A0_BASE, _________________);

Timer_A_startCounter(TIMER_A0_BASE, TIMER_A_CONTINUOUS_MODE);
```