



MINISTÈRE
DE L'ÉDUCATION
NATIONALE, DE
L'ENSEIGNEMENT
SUPÉRIEUR ET DE
LA RECHERCHE

EDE NUM 2

SESSION 2017

**CAPET
CONCOURS EXTERNE
TROISIÈME CONCOURS
ET CAFEP CORRESPONDANTS**

Section : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR

Option : INGÉNIERIE INFORMATIQUE

**ETUDE D'UN SYSTÈME, D'UN PROCÉDÉ OU D'UNE
ORGANISATION**

Durée : 4 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.

Dans le cas où un(e) candidat(e) repère ce qui lui semble être une erreur d'énoncé, il (elle) le signale très lisiblement sur sa copie, propose la correction et poursuit l'épreuve en conséquence.

De même, si cela vous conduit à formuler une ou plusieurs hypothèses, il vous est demandé de la (ou les) mentionner explicitement.

NB : La copie que vous rendrez ne devra, conformément au principe d'anonymat, comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé comporte notamment la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de signer ou de l'identifier.

Tournez la page S.V.P.

A

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 externe du CAPET de l'enseignement public :**

Concours	Section/option	Epreuve	Matière
EDE	1413E	102	4715

► **Concours externe du CAFEP/CAPET de l'enseignement privé :**

Concours	Section/option	Epreuve	Matière
EDF	1413E	102	4715

► **Troisième concours du CAPET de l'enseignement public :**

Concours	Section/option	Epreuve	Matière
EDV	1413E	102	4715

Constitution du sujet

- **texte**..... pages 2 à 21
(mise en situation et questions à traiter par le candidat)
- **documents techniques** pages 22 à 36
- **documents réponses** pages 37 à 43



**Les documents réponses DR1 à DR7 (pages 37 à 43)
doivent être rendus avec la copie.**

Contexte de l'étude

Les ouvrages d'art constituent un patrimoine important des infrastructures et exigent, comme tout ouvrage public, un entretien régulier. Les difficultés inhérentes à la surveillance et à l'entretien de ces ouvrages sont liées, paradoxalement, à leur grande durée de vie et à l'occurrence d'événements trop épisodiques pour maintenir une surveillance constante du gestionnaire (services préfectoraux ou concessionnaires).

Comme tout ouvrage d'art, les ponts à haubans doivent faire l'objet d'une surveillance dont la responsabilité incombe au gestionnaire.



Pont d'Aquitaine

L'Institut Français des Sciences et Technologies et Transport, de l'Aménagement et des Réseaux (IFSTTAR) est un établissement public dont une des missions est de réaliser ou faire réaliser des développements et des innovations dans le domaine du génie civil et de la sécurité des infrastructures. Le centre de Bouguenais (Loire Atlantique) a développé un système de surveillance des haubans de ponts, notamment la surveillance des ruptures de câbles.

Pour assurer cette surveillance, un système de détection par contrôle acoustique, appelé CASC (Contrôle Acoustique de Surveillance de Câbles), a été développé. Les câbles de ponts suspendus sont constitués d'un ensemble de torons, eux-mêmes constitués de fils métalliques. Au moment de la rupture d'un fil, une onde est générée et se propage le long du câble, de part et d'autre de la rupture. La détection et la datation de cette onde par différents capteurs positionnés le long du câble permet de localiser cette rupture et d'évaluer son importance. Les fréquences mesurées dans le cas d'une rupture (de 1 kHz à quelques kHz) sont audibles, si bien que l'on parle de surveillance acoustique. Les capteurs utilisés sont des accéléromètres piézo-électriques fixés au plus près du câble.



Capteurs CASC

Chaque capteur CASC effectue une datation et un enregistrement du signal et transmet ces informations à un superviseur situé à proximité de l'ouvrage. Le superviseur évalue alors la vitesse de l'onde et, connaissant la position relative des capteurs, détermine le lieu de la rupture.

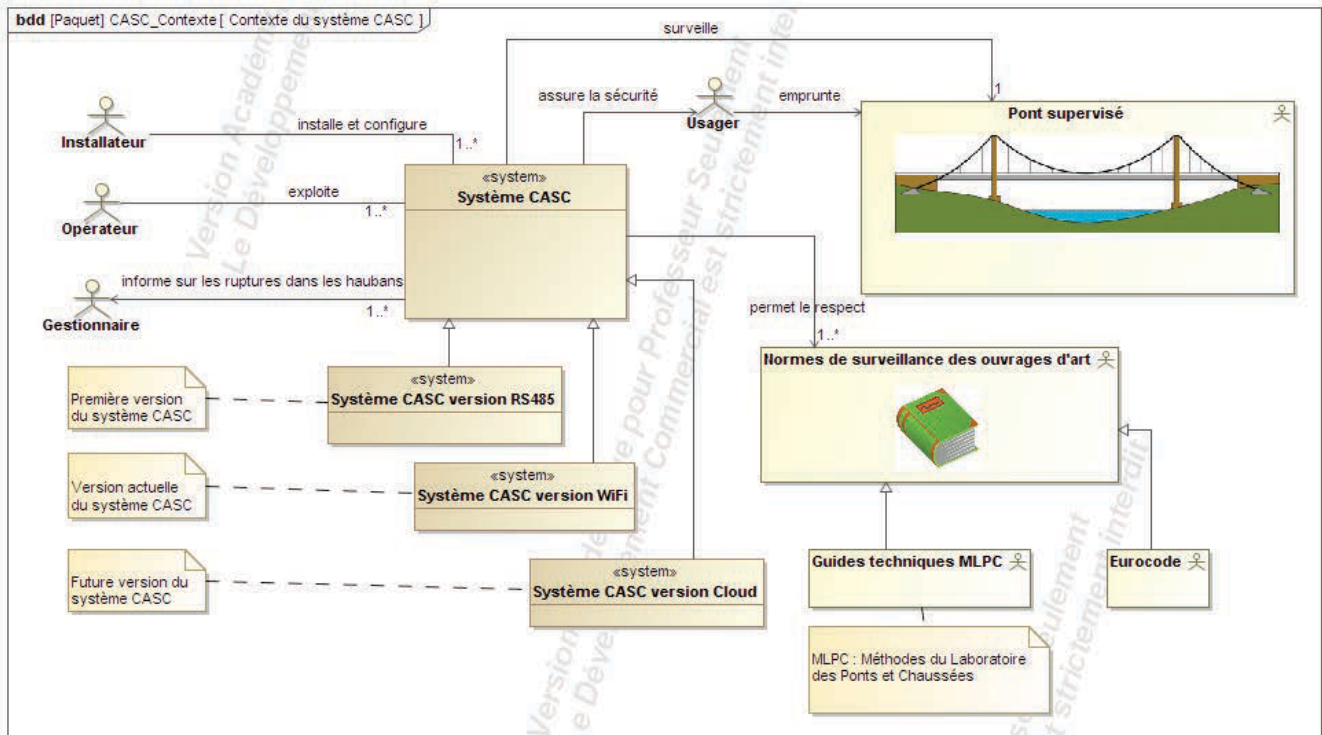


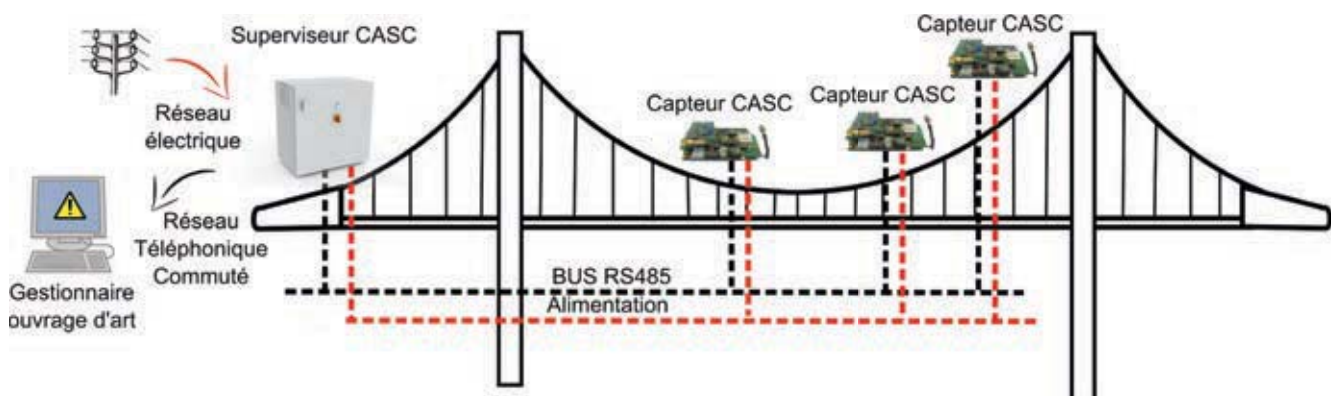
Diagramme de contexte du capteur CASC

Historiquement, plusieurs versions du capteur CASC ont été mises au point par le laboratoire de l'IFSTTAR. Ces différentes versions se distinguent les unes des autres par les choix technologiques concernant l'acquisition et la transmission des données.

Version RS485

Cette génération, en place sur de nombreux ouvrages, implique l'installation de quelques dizaines à quelques centaines de capteurs par pont (3 capteurs au minimum par câble).

Les cartes d'acquisition associées aux capteurs sont reliées au superviseur par une liaison RS485. Le superviseur intègre la gestion des capteurs ainsi que l'algorithme de localisation de la rupture et communique les données au gestionnaire via le réseau téléphonique. La forme de l'onde n'est pas accessible.



Version Wifi

L'objectif de cette nouvelle génération est d'enregistrer la forme des ondes se propageant dans les câbles et de moderniser la supervision des capteurs. La carte Pégase 1 (Plateforme Experte Générique pour Applications Sans-fil Embarquées) est une carte conçue par l'IFSTTAR, développée et commercialisée par A3IP (Sautron, Loire Atlantique) et par Powerlan (Saint Mars du Désert, Loire Atlantique) permettant de réaliser certaines fonctions réutilisables sur différents systèmes d'instrumentations.

Le développement de cette carte s'inscrit dans le domaine de l'instrumentation sans fil. Elle permet de prendre en compte les évolutions au niveau électronique (protocoles sans fil, composants basse consommation, ...) associées à la recherche de systèmes plus économiques.

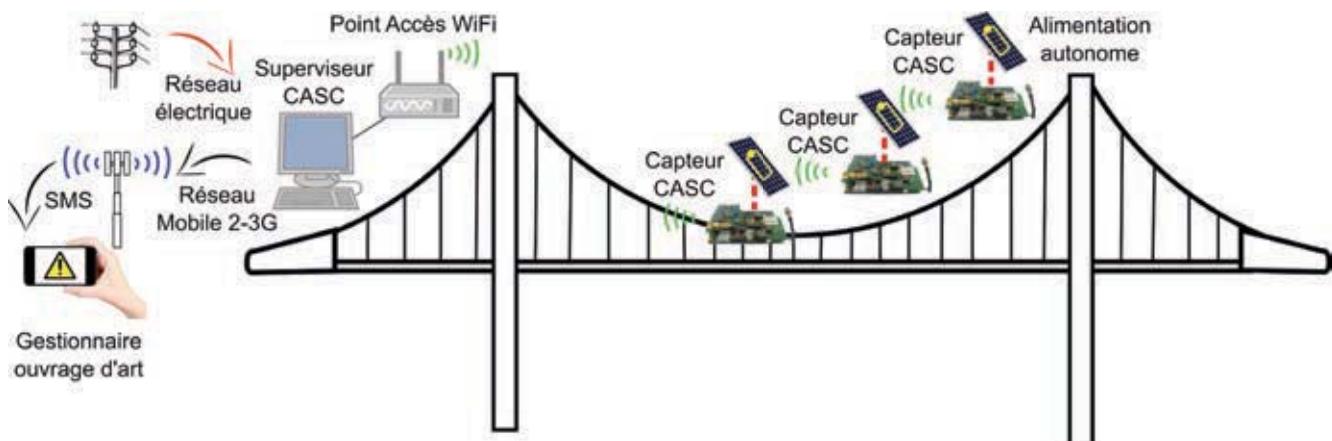
La généricité matérielle est assurée par la possibilité d'ajouter différentes cartes filles réalisant les fonctions spécifiques à l'instrumentation envisagée, surveillance de ponts ou autres missions définies par le commanditaire.

La généricité logicielle est aussi assurée (programmation en C++, noyau Linux embarqué au niveau de la carte mère) afin de faciliter la prise en main par l'utilisateur.

Par ailleurs, l'alimentation de l'ensemble a été révisée pour améliorer l'autonomie des capteurs et cartes en utilisant des panneaux photovoltaïques et des batteries au niveau de chaque capteur.



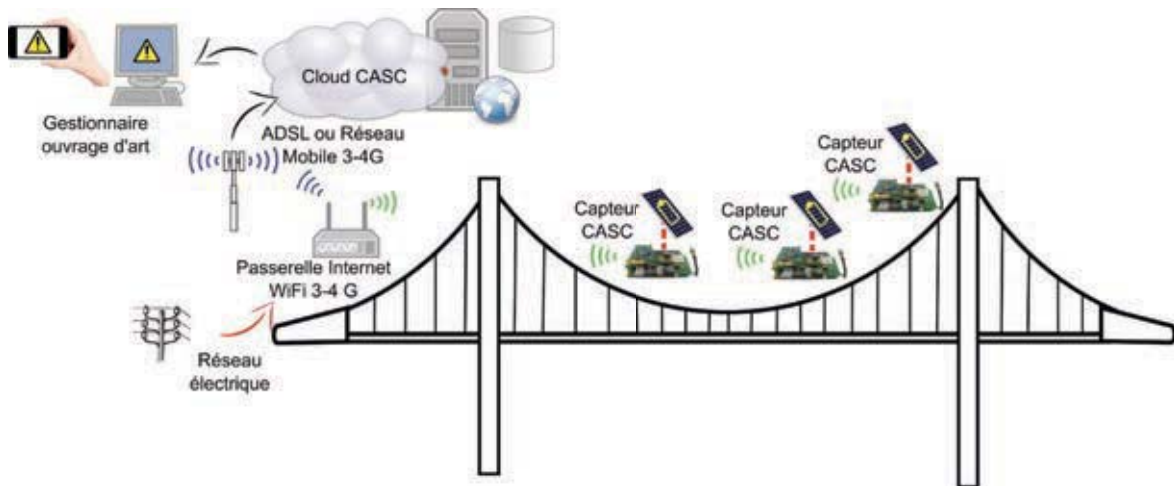
Carte Pégase 1



Le superviseur évolue d'une armoire à un ordinateur en permanence sous-tension et situé dans une des piles du pont.

Version Cloud

Les évolutions futures, en phase finale de développement, permettront de gérer un ensemble de ponts. Pour cela, le traitement et le stockage des données se feront sur le Cloud. La transmission des données ne nécessitera plus la présence sur chaque pont d'un superviseur, qui sera remplacé par une passerelle Internet constituée d'un point d'accès Wifi et d'un accès au réseau 3G ou 4G.



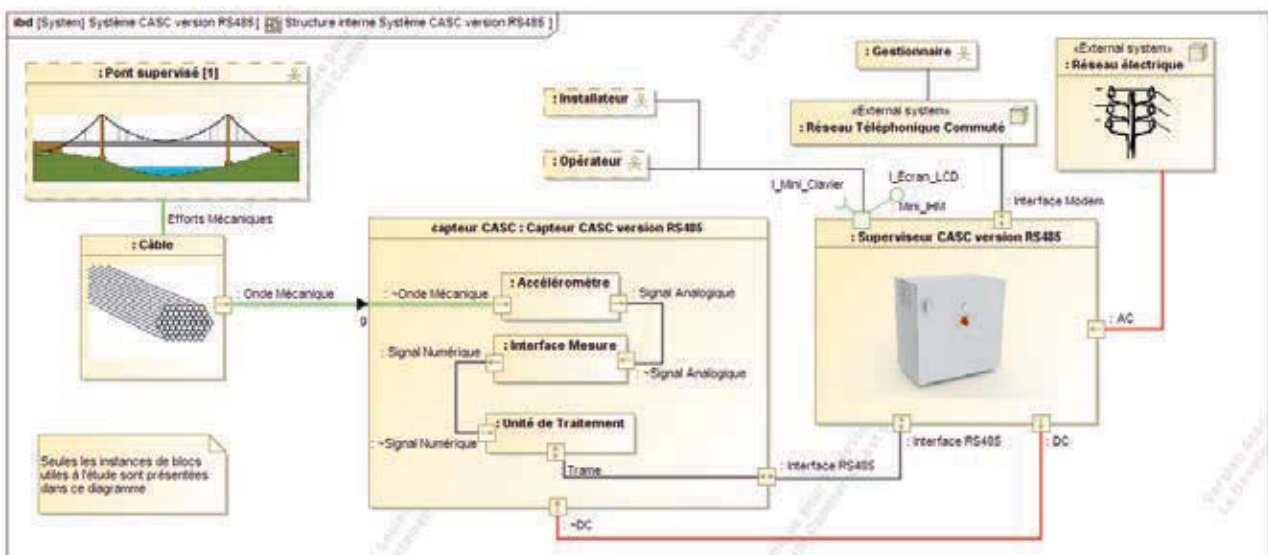
L'objectif de cette étude, constituée de 6 parties indépendantes, est d'étudier ces différentes générations du capteur CASC, notamment l'impact en termes d'équipement et de logiciels, en suivant le cheminement des données associées à une onde de rupture, de l'acquisition du capteur jusqu'à l'exploitation des données collectées.

Partie 1 – Étude des différentes générations de capteur CASC

Objectif : expliquer les conséquences de l'évolution de la transmission de type RS485 vers une transmission de type Wifi puis vers un traitement localisé dans le Cloud, du point de vue matériel.

Les différentes générations de capteurs CASC étudiées ont toutes pour objectif la surveillance de haubans de ponts. Si la base du système reste la même (présence d'un accéléromètre émettant un signal analogique vers un CAN qui convertit le signal, d'une carte mère qui effectue un premier traitement des données puis les transmet à un superviseur qui réalise l'analyse et le stockage des informations et alerte le gestionnaire), les technologies employées pour réaliser la transmission des données et leur traitement ont considérablement évolué, entraînant une évolution tant matérielle que logicielle.

Le diagramme de blocs internes ci-dessous décrit la première version du capteur CASC (version RS485).



Question 1. Sur le document réponse DR1, compléter le diagramme de définition de blocs du capteur CASC, version RS485, en ajoutant dans les cadres prévus les composants et cardinalités attendus.

Dans la deuxième génération, le bus RS485 est remplacé par une communication Wifi.

Question 2. Décrire les avantages et les inconvénients des deux modes de transmission de ces deux générations de capteurs CASC. Justifier alors le choix de l'évolution de la transmission RS485 vers la transmission Wifi.

Question 3. Le protocole Wifi correspond aux couches physique et liaison de données du modèle OSI. Au niveau de la couche liaison de données, la méthode d'accès au support est de type CSMA/CA (Carrier Sense Multiple Access Collision Avoidance). Après avoir décrit la méthode CSMA/CD (Carrier Sense Multiple Access Collision Detection) utilisée dans les réseaux Ethernet filaires, préciser la différence avec la méthode CSMA/CA.

Question 4. Indiquer quelles sont les conséquences en termes de déterminisme sur la transmission.

Question 5. Après avoir consulté la documentation sur le composant GPS (document technique DT1), expliquer en quoi l'ajout d'une telle fonctionnalité à la carte Pégase permet de remédier au problème lié à la datation des événements dans le cas d'une transmission Wifi.

Lors de cette évolution, les commanditaires ont aussi exprimé le besoin d'avoir accès à la forme de l'onde. Cette nouvelle fonctionnalité, ainsi que l'interface graphique qui l'accompagne, a conduit le laboratoire à modifier le superviseur : un ordinateur réalise désormais la supervision de l'ensemble des capteurs présents sur un ouvrage. Afin de protéger cet ordinateur, celui-ci est placé à l'intérieur d'une pile du pont surveillé. Un point d'accès Wifi est alors placé à l'extérieur de la pile et la transmission entre les capteurs et le superviseur est réalisée suivant le protocole TCP/IP.

Au niveau du capteur CASC, une carte fille dédiée à la mesure (acquisition et mise en forme du signal) et une carte fille gérant l'alimentation en énergie (batteries et panneaux solaires) sont ajoutées à la carte mère Pégase 1.

Question 6. Sur le document réponse DR1, compléter le diagramme de définition de blocs du capteur CASC pour la version Wifi. Pour chaque bloc complété, indiquer sa cardinalité.

La dernière génération de capteur CASC permettra au gestionnaire de surveiller un ensemble de ponts sans avoir à se déplacer pour relever les données enregistrées par les différents superviseurs. Le traitement des données est alors effectué par une application CASC installée dans le Cloud. Le superviseur disparaît au profit d'une passerelle internet, constituée d'un point d'accès Wifi associé à un routeur 3G/4G.

Par ailleurs, sur cette nouvelle génération, la carte mère Pégase évolue afin de gérer l'alimentation en l'énergie.

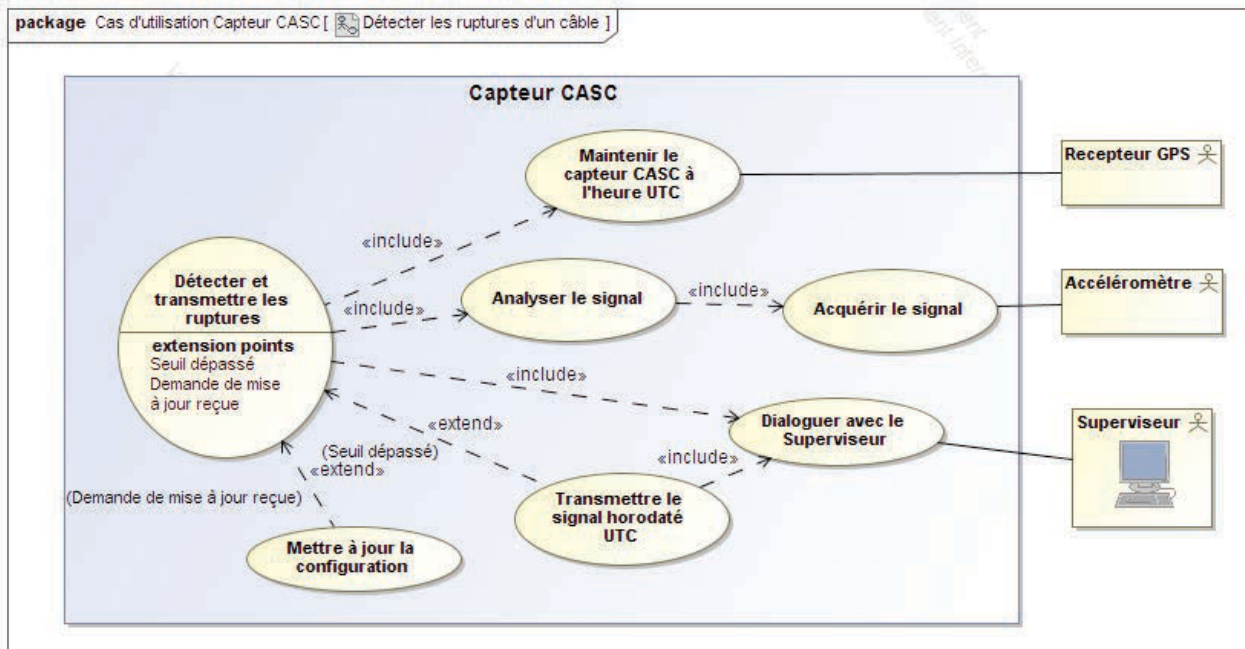
Question 7. Sur le document réponse DR2, compléter le diagramme de définition de blocs du capteur CASC pour la version Cloud. Pour chaque bloc complété, indiquer sa cardinalité.

Partie 2 – Transmission des données du capteur CASC vers le superviseur

Objectif : valider les données transmises par le capteur CASC au superviseur dans le cas d'un capteur CASC de deuxième génération (transmission Wifi).

Lorsqu'une rupture se produit, une onde acoustique se propage le long du toron. Cette onde peut être détectée au moyen d'un accéléromètre. De nombreuses autres ondes sont présentes dans la structure, chacune ayant une fréquence et une amplitude spécifiques. Parmi les signaux émis par l'accéléromètre, donc correspondant à la gamme de fréquences intéressante dans le cadre de la détection d'une rupture, seuls ceux ayant une amplitude supérieure à un seuil défini par l'utilisateur seront transmis au superviseur pour être traités, car correspondant à la rupture d'un fil d'acier.

Le capteur CASC doit donc réaliser un certain nombre de fonctions, décrites dans le diagramme des cas d'utilisation ci-dessous.

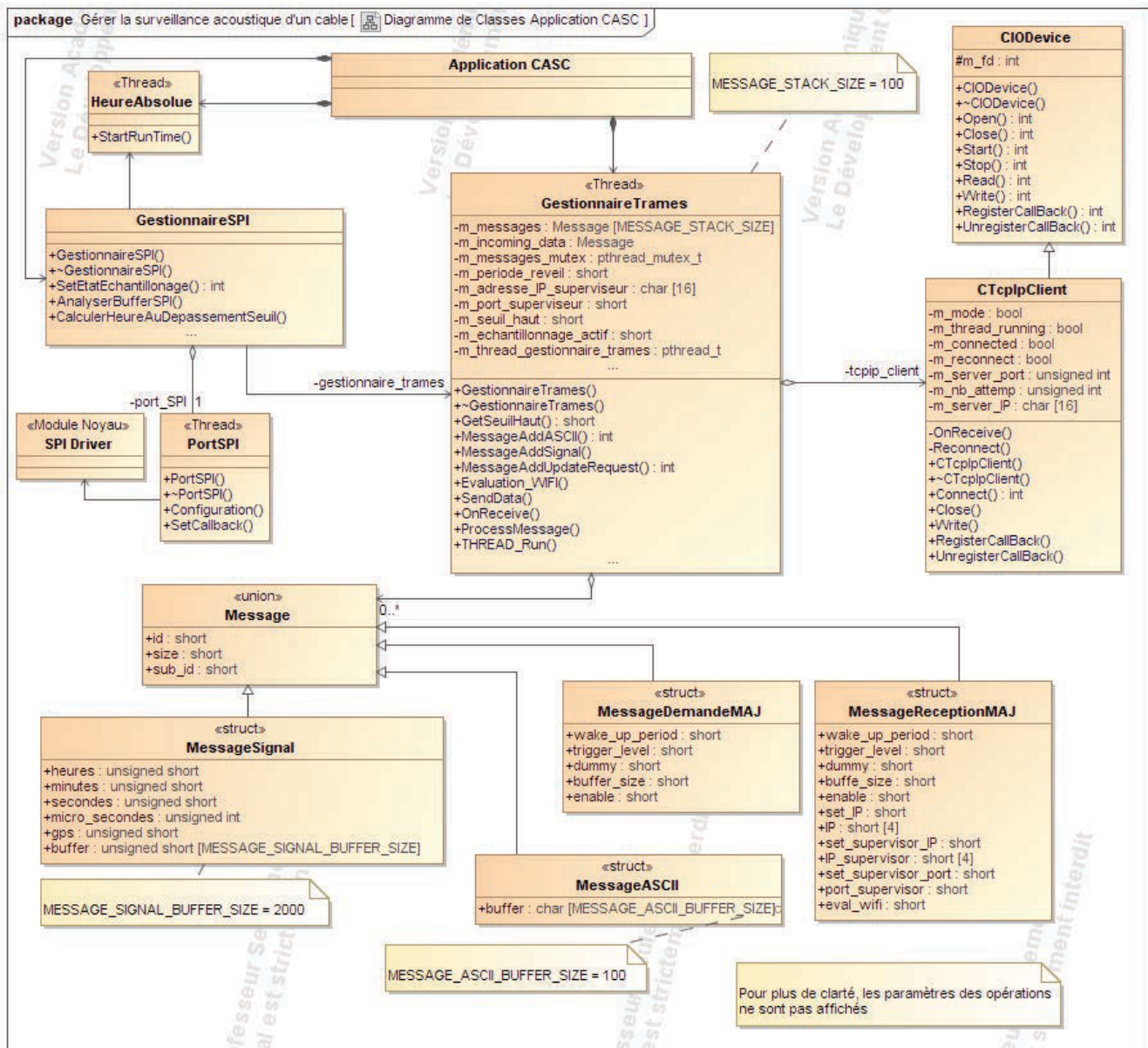


Question 8. À partir du diagramme des cas d'utilisation de l'application CASC, compléter le diagramme de séquence sur le document réponse DR3.

Fonction : Analyser le signal

L'application CASC sur le capteur est réalisée en programmation orientée objet en langage C++. Le document technique DT2 rappelle quelques commandes courantes utilisées dans ce langage.

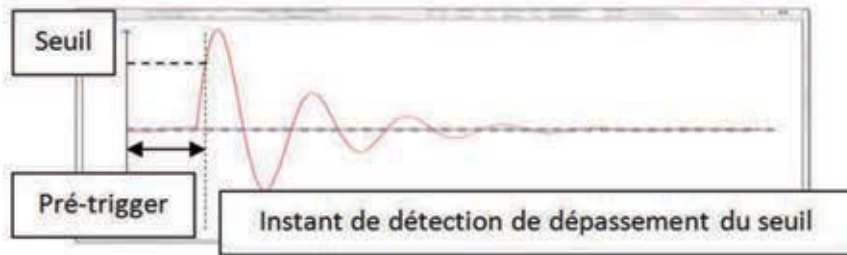
Le diagramme de classes de cette application est donné ci-dessous.



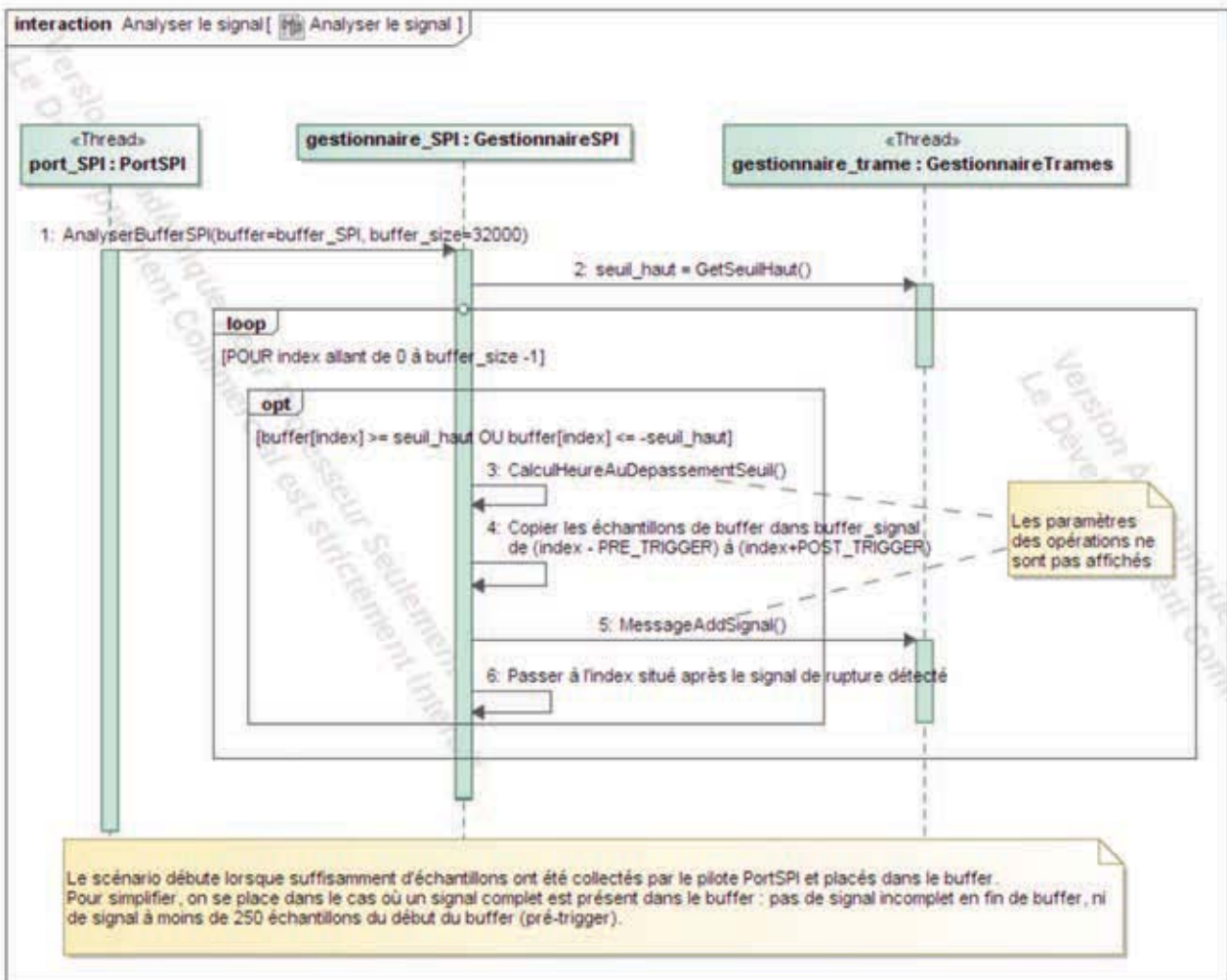
L'acquisition du signal se fait par le module noyau du port SPI sur lequel est connecté le convertisseur analogique-numérique (CAN) chargé de la numérisation des ondes détectées par l'accéléromètre. Dès que la quantité de données collectées est suffisante, le GestionnaireSpi procède à l'analyse de celles-ci.

Question 9. Préciser le type de relation UML existant entre la classe GestionnaireSPI et la classe GestionnaireTrames. Justifier l'existence de cette relation.

Parmi les différentes méthodes de la classe GestionnaireSPI, la méthode AnalyserBufferSPI() permet de sélectionner les données à transmettre au superviseur. Si l'amplitude du signal dépasse un seuil déterminé au préalable par l'opérateur, les données transmises par le port SPI sont stockées dans un buffer. Afin de permettre l'analyse de ce signal et de conclure à une rupture, l'opérateur doit pouvoir consulter les données enregistrées un peu avant la détection du dépassement de seuil. Un pré-trigger et un post-trigger sont donc définis pour avoir une fenêtre suffisamment large du signal, et conclure ou non à une rupture de fil.



La méthode AnalyserBufferSPI() est définie grâce au diagramme de séquence suivant :



Les signatures des méthodes appelées depuis AnalyserBufferSPI() sont les suivantes :

- **CalculHeureAuDepSeuil** (unsigned int index, short *heure, short *minute, short *seconde, int *microseconde, int *gps)
Cette méthode détermine et retourne, à partir des données du GPS, l'heure absolue à laquelle l'événement est détecté sous forme heures, minutes, secondes, microsecondes ainsi que le nombre de satellites détectés par le récepteur GPS (passage des paramètres par adresse) ;
- **MessageAddSignal** (short heures, short minutes, short secondes, int micro_secondes, short gps, short *data, unsigned int data_size).

Cette méthode de la classe GestionnaireTrames permet de constituer la trame des données à transmettre au superviseur. Ces données comprennent l'heure où le dépassement de seuil a été détecté (3 shorts et un integer), le nombre de satellites détectés par le récepteur GPS, les données du capteur (2000 échantillons, soit 250 avant la détection de seuil et 1750 après) et enfin la taille du buffer.

On donne le début de la méthode AnalyserBufferSPI() :

```
void GestionnaireSPI::AnalyserBufferSPI(short *buffer, unsigned int buffer_size)
{
    // Variables de parcours du tableau lu
    unsigned int index = 0;

    // Variables pour la détection du signal de rupture
    short buffer_signal[NOMBRE_ECHANTILLONS_TOTAL];
    int heure, minute, seconde, microseconde, gps;

    // Parcours et analyse le nouveau tableau acquis à partir de 'index'
    short seuil_haut = this->gestionnaire_trames->GetSeuilHaut();
    ...
}
```

Question 10. Compléter en C++ la méthode AnalyserBufferSPI() sur le document réponse DR4.

La méthode MessageAddSignal() constitue une zone critique : le message ne doit être émis que si la trame est construite dans son intégralité. Or, la classe GestionnaireTrames est un thread, donc pouvant faire l'objet d'une préemption de la part de l'ordonnanceur, par exemple, si le port SPI émet de nouvelles données correspondant à une nouvelle rupture.

Question 11. Indiquer quel mécanisme peut garantir que le message est construit intégralement avant son émission.

L'allocation des messages en mémoire se fait de manière dynamique (instructions C++ `new` et `delete`).

Question 12. Après avoir rappelé le nom de la zone mémoire où seront stockés les messages, indiquer les problèmes éventuels pouvant survenir avec ce type d'allocation mémoire et préciser les solutions permettant d'y remédier.

Fonction : Dialoguer avec le superviseur

Une fois le message mis en forme par le gestionnaire de trames, celui-ci est transmis au superviseur. Le protocole de transmission utilisé est TCP/IP.

Question 13. Décrire les avantages et les inconvénients des protocoles de transmission UDP et TCP. Justifier le choix de la transmission en TCP/IP.

Le code source des structures (Message, MessageSignal, ...) est fourni dans le document technique DT3.

Question 14. À partir du diagramme de classes de l'application CASC, de l'extrait de code source du document technique DT3 et du document technique DT2 indiquant les tailles de différentes variables en C++, déterminer la taille maximale en octets d'un message (de type Message) émis par le capteur CASC vers le superviseur.

Lors d'un test de rupture, le superviseur ne reçoit aucun message de la part de l'un des capteurs CASC présent sur le pont. Le journal de l'application CASC du capteur concerné s'achève par le message « Connection time out ». Le test de connectivité entre le capteur et le superviseur (ping) est correct. Le superviseur est configuré pour recevoir les messages de cette application CASC sur le port TCP 4000.

Les masques permettant d'analyser les trames sont fournis dans le document technique DT4. Afin d'identifier le problème, l'opérateur de maintenance contrôle la communication au moyen du logiciel Wireshark. La capture d'écran du document technique DT5 correspond aux échanges entre le superviseur et l'application défaillante.

Question 15. Préciser le rôle (client ou serveur) du superviseur et de l'application CASC exécutée sur le capteur.

Question 16. Identifier le problème rencontré lors de cette communication.

Ce problème étant résolu, une nouvelle capture d'écran correspondant à une communication normale est réalisée. Elle est fournie dans le document technique DT6.

Question 17. Compléter le document réponse DR5 en indiquant les adresses MAC, IP et les ports du capteur et du superviseur.

Question 18. Donner la signification des trames 2 et 3.

Question 19. Détailler le rôle des trames 1, 4 et 5.

Comme indiqué dans le diagramme de classes, la communication entre le superviseur et le capteur CASC peut contenir différents types de messages décrits dans le diagramme de classes de l'application CASC. Chaque type de message est précédé d'un identifiant. La spécification de ces trames est donnée ci-dessous.

Identifiant	Description	Émetteur	Destinataire
14	Transmission d'un signal. Le capteur transmet des données au superviseur.	Capteur	Superviseur
15	Demande de mise à jour. Le capteur demande au superviseur si l'un des paramètres a été changé.	Capteur	Superviseur
20	Réception d'une mise à jour. Le capteur reçoit les nouveaux paramètres de fonctionnement.	Superviseur	Capteur
21	Réception d'une mise à jour sans contenu. Aucun paramètre à mettre à jour.	Superviseur	Capteur
24	Transmission d'un message ASCII. Le capteur transmet un message texte au superviseur.	Capteur	Superviseur

Une capture d'écran correspondant aux données du message émis par le capteur CASC est fournie ci-dessous (détail de la trame N°6 de l'échange précédent).

```

6 0.001725000 192.168.0.11 192.168.0.20 TCP 1514 51220 > terabase [ACK] Seq=1 Ack=1 Win=293
▶ Frame 6: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▶ Ethernet II, Src: Sony_5a:ed:bd (54:42:49:5a:ed:bd), Dst: Dell_c5:b...
▶ Internet Protocol Version 4, Src: 192.168.0.11 (192.168.0.11), Dst: 192.168.0.20 (192.168.0.20)
▶ Transmission Control Protocol, Src Port: 51220 (51220), Dst Port: t...
▼ Data (1448 bytes)
Data: 0e00ac0f00000e0014000500000003140100c8ffc6ffc5ff...
[Length: 1448]

0040  ca dd 0e 00 ac 0f 00 00 0e 00 14 00 05 00 00 00 00 00 00 00 00 00
0050  03 14 01 00 c8 ff c6 ff c5 ff c3 ff c1 ff c0 ff c0 ff c0 ff
0060  be ff bd ff bb ff ba ff b8 ff b7 ff b5 ff b4 ff b3 ff b2 ff
0070  b3 ff b1 ff b0 ff af ff ae ff ad ff ab ff aa ff a9 ff a8 ff
0080  a9 ff a8 ff a7 ff a6 ff a5 ff a5 ff a4 ff a3 ff a2 ff a1 ff
0090  a2 ff a2 ff a1 ff a0 ff a0 ff 9f ff 9f ff 9e ff 9d ff 9d ff
00a0  9e ff 9e ff 9d ff 9d ff 9d ff 9d ff 9d ff 9d ff 9d ff 9d ff

```

Question 20. Préciser le type de message émis dans cette trame.

Question 21. À partir de la capture d'écran présentée dans le document technique DT6, indiquer le numéro des trames contenant les données du message émis par le capteur CASC et justifier le nombre de trames nécessaires à cette émission.

Partie 3 – Évolution vers la gestion de n ponts : troisième génération du capteur CASC

Objectif : mettre en place une partie des modifications imposées par la délocalisation du stockage et du traitement des données sur le Cloud.

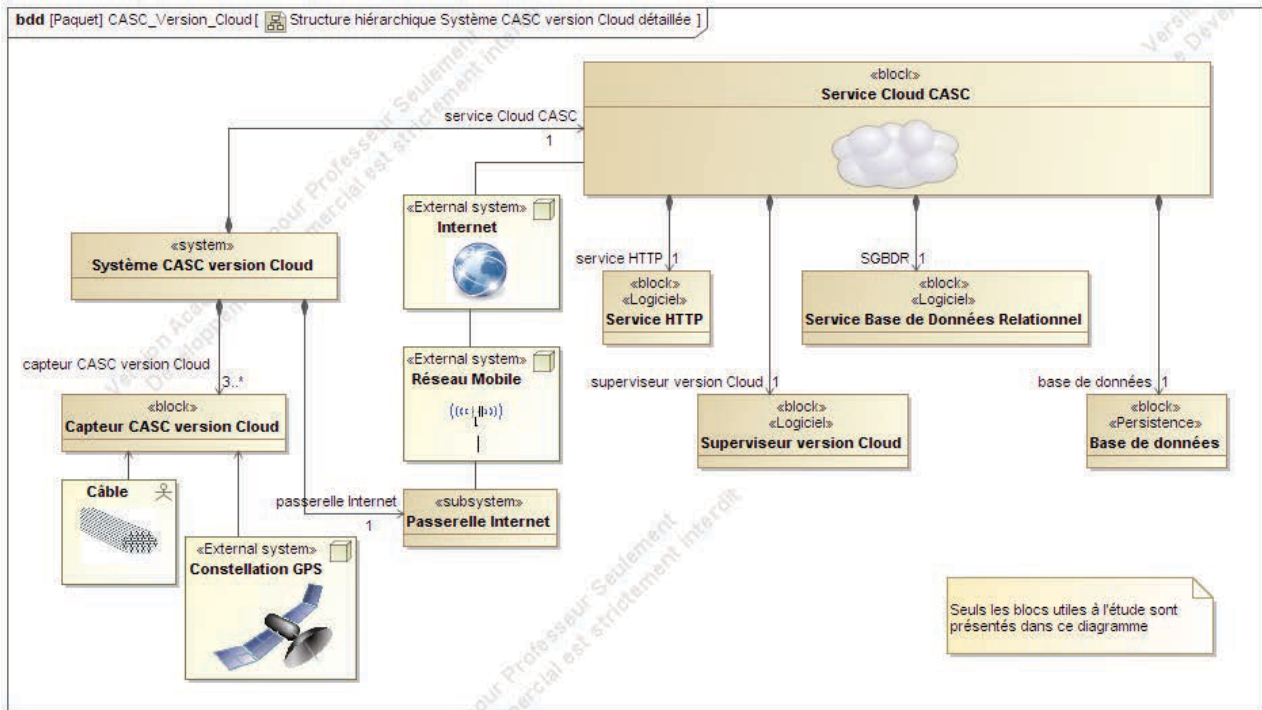
Le capteur CASC de deuxième génération (transmission par Wifi vers un superviseur localisé dans une des piles du pont) impose la mise en place d'un ordinateur constamment en service sur chaque ouvrage. Le gestionnaire et l'équipe de maintenance doivent donc se déplacer pour accéder aux données recueillies par le système.

Afin de faciliter le travail des différents intervenants, la troisième génération du capteur CASC permet la suppression du superviseur local au profit d'un service qui assure le stockage et le traitement des données, localisé dans le Cloud. Un gestionnaire peut ainsi suivre les événements détectés sur tous les ponts dont il a la responsabilité depuis l'ordinateur de son bureau.

La mise en place de cette génération nécessite des changements au niveau matériel (abandon du superviseur local au profit d'une passerelle internet 3-4 G, évolution de la carte mère Pégase 1 pour stocker les données, ...) et logiciel (évolution du noyau linux, protocole http, ...). Sur le capteur CASC, le système d'exploitation utilisé est désormais Linux Debian.



Carte Pégase 2



Afin de tester cette nouvelle version, la mise en place d'un nouveau capteur est étudiée dans cette partie.

Étude du processeur du nouveau capteur CASC

Le document technique DT7 détaille l'architecture du processeur DM3730 implanté sur le module Gumstix OVERO-Fire utilisé pour la carte Pégase 2 (carte mère du nouveau capteur CASC).

Question 22. Le processeur utilisé est basé sur une architecture ARM Cortex A8 possédant un moteur SIMD. Décrire succinctement ce mode de fonctionnement et justifier son intérêt pour l'application CASC.

La nouvelle version du capteur CASC mise en œuvre devra permettre des analyses plus fines des signaux. En complément de la détection des seuils, des analyses plus fines, telles que l'analyse spectrale des signaux permettant de détecter puis d'éliminer les faux-positifs devra être possible.

Question 23. Expliquer les raisons pour lesquelles le choix de ce processeur améliore les performances de l'application conformément à l'évolution de traitement du signal souhaité.

Phase de compilation du programme CASC embarqué sur le capteur CASC

Le programme CASC embarqué sur le capteur CASC chargé de la détection des ruptures et de la transmission des signaux vers le service Cloud est écrit en C++ et les différents fichiers sources le composant doivent être compilés afin de produire l'exécutable. Le document technique DT8 rappelle les différentes phases de la compilation d'un fichier.

Afin de pouvoir suivre certaines étapes de la compilation, les commandes ci-dessous sont proposées, avec les réponses affichées lors de la phase de compilation.

```

1  $ cd casc/src
2  $ ls Classes
3  Casc.cpp
4  Casc.h
5  GestionnaireSPI.cpp
6  GestionnaireSPI.h
7  GestionnaireTrames.cpp
8  GestionnaireTrames.h
9  ...
10 $ make

11 g++ -g -Wall -I -c Classes/GestionnaireSPI.cpp -o
12 build/obj/GestionnaireSPI.o
13 g++ -g -Wall -I -c Classes/GestionnaireTrames.cpp -o
14 build/obj/GestionnaireTrames.o
15 ...
16 g++ -g -Wall -I -c Classes/Casc.cpp -o build/obj/Casc.o
17 g++ -g -Wall -I build/obj/GestionnaireSPI.o
18 build/obj/GestionnaireTrames.o ... build/obj/Casc.o -lboost_thread
19 -lboost_system -lboost_system -lboost_date_time -lboost_regex -lpthread
20 -o build/Release/bin/Casc

```

Question 24. Compléter le tableau du document réponse DR5 en indiquant par une croix les possibilités associées aux différents types de fichiers utilisés lors de la compilation et de la production de l'exécutable.

Question 25. Préciser la phase de compilation correspondant à la commande de compilation commençant à la ligne 17 (de g++ jusqu'à build/Release/bin/Casc).

Phase de démarrage d'un capteur CASC

Lorsqu'un capteur CASC est mis sous tension, après le chargement du noyau Linux, le script shell figurant sur le document technique DT9 s'exécute automatiquement.

Question 26. Préciser le rôle des lignes 28 à 43 de ce script.

Lors de l'exécution de ce script, le journal du système affiche le message d'erreur ci-dessous :

```
bash: ./autorun: Aucun fichier ou dossier de ce type
```

L'installateur interrompt le démarrage du capteur et entre alors la commande « ls -l » dans le dossier du script afin d'identifier le problème. La capture d'écran ci-dessous correspond à cette phase :

```

# cd /etc/script_autorun
# ls -l
total 4-rw-r--r-- 1 root root 994 juil. 11 12:33 autorun.sh

```

Question 27. Préciser la nature du problème rencontré.

Question 28. À l'aide du document technique DT10 décrivant les commandes relatives aux droits associés à un fichier sous Linux, proposer une commande qui résolve le problème rencontré.

Transmission des données : protocole http

Question 29. Sur le document réponse DR6 représentant le diagramme de blocs internes correspondant au système CASC version Cloud décrit en début de cette partie, indiquer le cheminement des données issues d'une onde de rupture détectée par le capteur vers le lieu de leur stockage dans le Cloud.

Lorsqu'une rupture est détectée sur un hauban de pont, les données relatives à cet événement sont émises vers le Cloud. Ces données sont : l'instant où la rupture a été détectée (timestamp), l'adresse mac du capteur (capteur), le nombre de satellites détectés par le récepteur GPS (gps) et les 2000 valeurs enregistrées par le capteur (signal). L'acheminement de ces données se fait au moyen de requêtes http entre le capteur CASC et le service Cloud CASC.

Une capture d'écran du logiciel wireshark correspondant à l'émission des données est donnée ci-dessous :

```

1  POST /cloudcascapp/evenement_casc/HTTP/1.1
2  Host: casc.ifstar.fr
3  Content-Type: application/x-www-form-urlencoded
4  Accept: */*
5  Connection: close
6  Content-Length:16241
7
8  timestamp = 2016-07-09 T13:25:28.723773Z
   À compléter sur copie gps=5&signal=-56:-58:
   :194:195:195:195:195:
9
10
11 HTTP/1.1 201 Created
12 Date: Sat, 09 Jul 2016 13:25:28 GMT
13 Server: Apache/2.2.22 (Debian)
14 X-Frame-Options: SAMEORIGIN
15 Vary: Accept-Encoding
16 Connection: close
17 Transfer-Encoding: chunked
18 Content-Type: text/html; charset=utf-8

```

Question 30. Compléter le document réponse DR6 en indiquant le rôle des différents champs de la requête http.

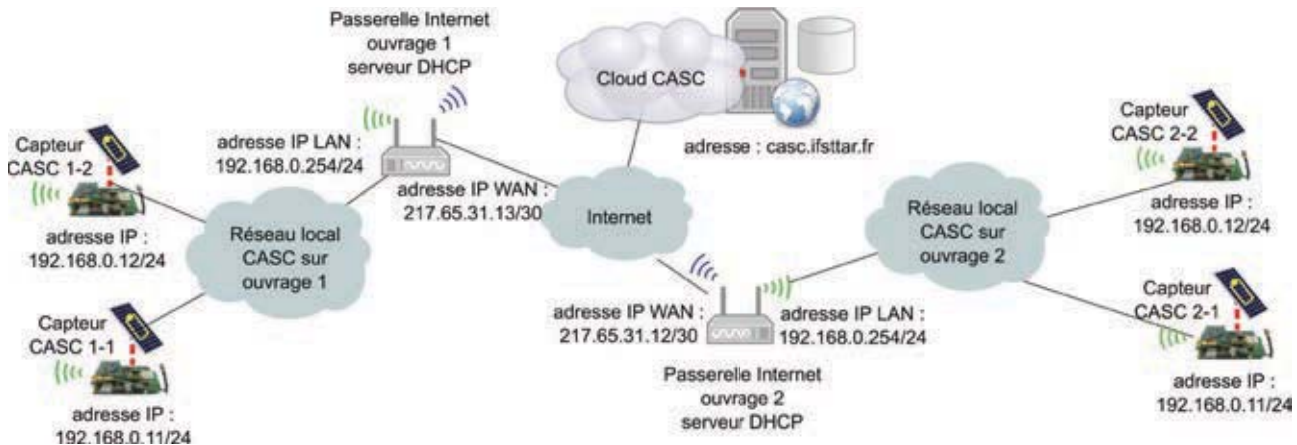
Question 31. La requête http adressée par le capteur au service Cloud CASC est une requête POST. Après avoir précisé la différence entre les requêtes POST et GET, justifier l'utilisation du POST dans la situation étudiée.

Question 32. Compléter la partie manquante de la requête http (cadre « À compléter sur copie » de la capture d'écran ci-dessus).

Question 33. La réponse du serveur commence à la ligne « http/1.1 201 Created ». Indiquer la signification du « 201 » sur cette ligne. Préciser la nature du problème rencontré dans le cas où le nombre « 404 » est renvoyé à la place de « 201 ».

Gestion de plusieurs ponts

La mise en place de la gestion de plusieurs ouvrages conduit à une situation représentée ci-dessous.



Question 34. Les capteurs CASC 1-1 et CASC 2-1 ont la même adresse IP. Justifier le fait que le routage ne présente aucune ambiguïté.

Question 35. Nommer et décrire le mécanisme mis en jeu au niveau de la passerelle pour assurer la communication avec le Cloud.

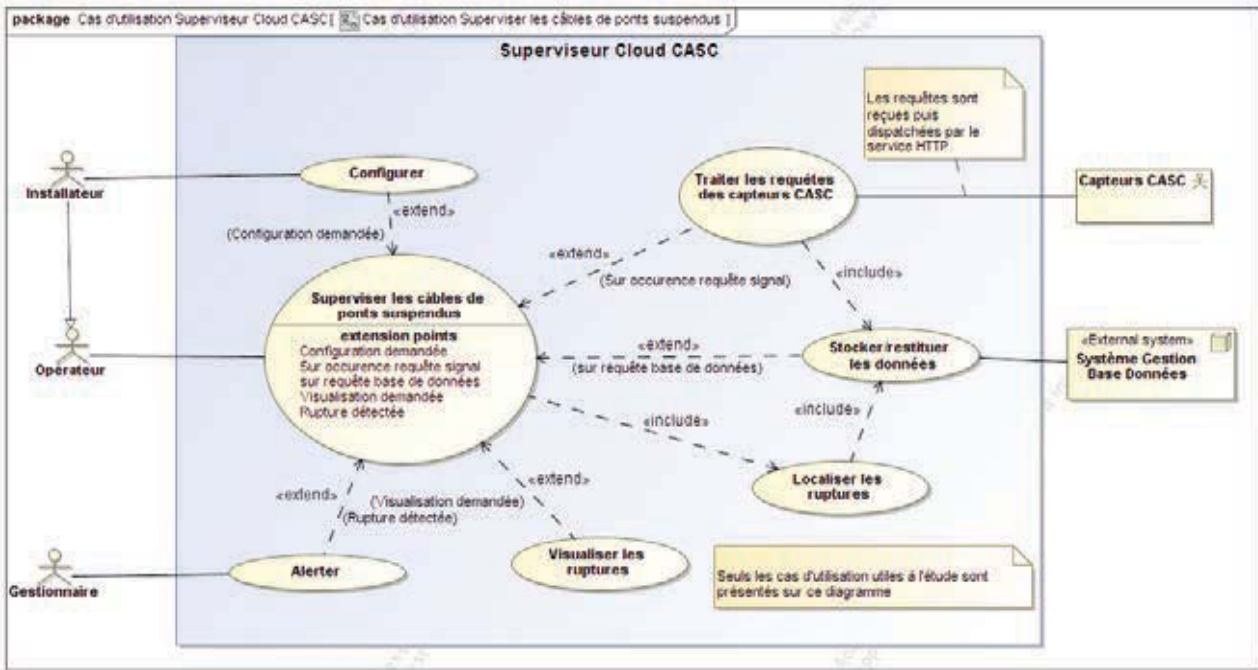
Question 36. Préciser l'adresse IP de la passerelle par défaut devant être configurée sur le capteur CASC 1-1.

Partie 4 – Évolution vers la gestion de n ponts : troisième génération

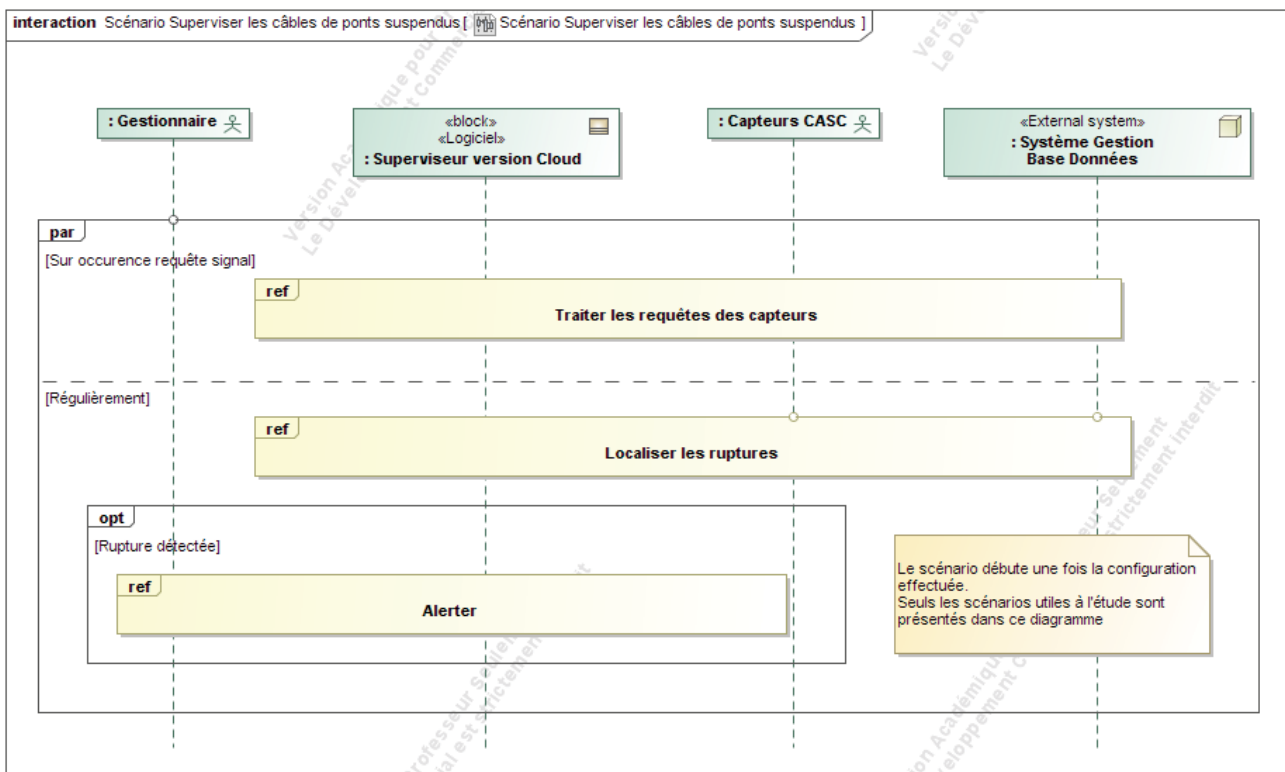
Objectif : stocker et traiter les informations relatives aux ruptures.

Lorsqu'une rupture est détectée, les données issues des capteurs sont émises vers le Cloud pour y être stockées et traitées.

Le diagramme des cas d'utilisation suivant présente les fonctionnalités réalisées par la version Cloud du superviseur.



Le diagramme de séquence suivant présente un scénario de traitement des requêtes issues des capteurs et de localisation des ruptures.



Le traitement des requêtes des capteurs ou la localisation des ruptures font appel au cas d'utilisation « Stocker/restituer les données ». La mission de cette fonction est l'interface avec le système de gestion de base de données chargé de la gestion de la persistance des données dans le Cloud. Une base de données contient l'ensemble des tables nécessaires au fonctionnement de l'application de supervision version Cloud.

Stockage des données

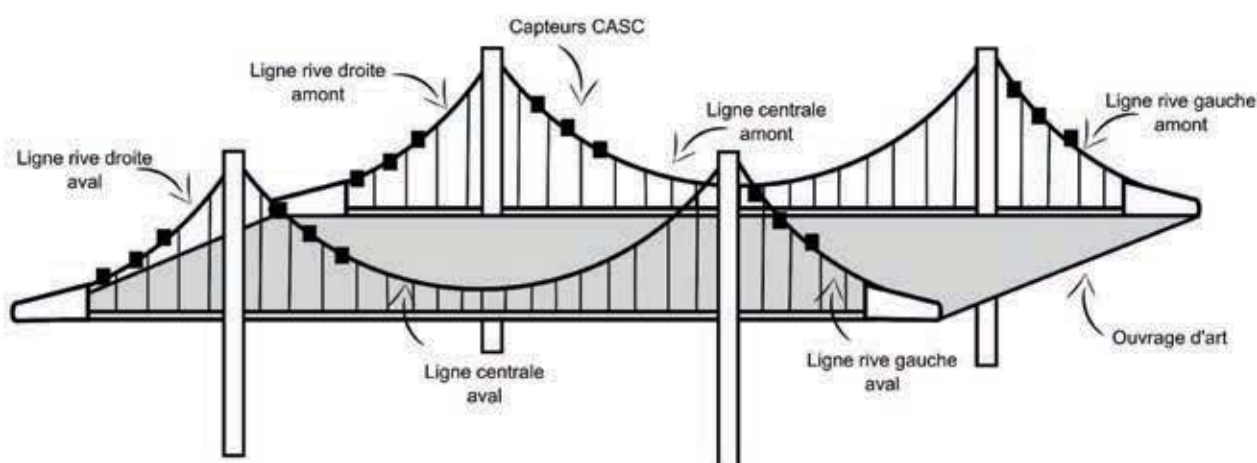
La table `cloudcascapp_ouvrage` contient les renseignements relatifs aux ponts surveillés.

Sur ces ponts, des lignes de capteurs sont définies (table `cloudcascapp_ligne`).

Une ligne est constituée d'un ensemble de capteurs (renseignements relatifs aux capteurs contenus dans la table `cloudcascapp_capteur`).



Ligne de capteurs CASC
(pont d'Ancenis (44))



La table permettant le stockage de toutes les données relatives aux différents ouvrages est présentée dans le document technique DT11.

Pour chaque pont, un gestionnaire est défini (table `auth_user`). Afin de consulter les données relatives aux ponts dont il est chargé, un gestionnaire d'ouvrage doit pouvoir consulter les données stockées (`auth_permission` et `auth_user_user_permission`).

Lorsqu'une rupture se produit les données enregistrées par les capteurs sont stockées dans les tables `cloudcascapp_evenementcapteur` (à chaque événement) et `cloudcascapp_rupture` (après localisation de la rupture).

Les requêtes seront écrites en langage SQL.

Question 37. Préciser la nature et le rôle des attributs `adresse_mac` et `ligne_id` de la table `cloudcascapp_capteur`.

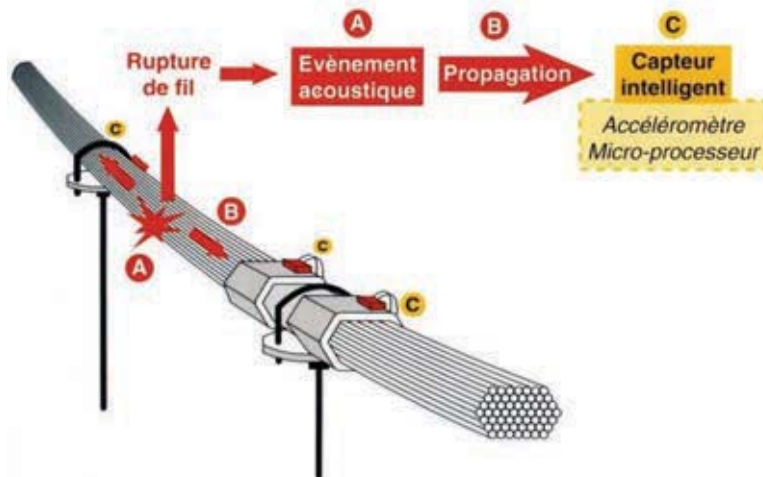
Question 38. Écrire une requête qui modifie l'attribut `seuil_haut` de tous les capteurs référencés dans la table `cloudcascapp_capteur`. La nouvelle valeur du seuil sera de 150.

Question 39. Une nouvelle ligne doit être insérée sur le pont d'identifiant `ouvrage_id = 2`. Le nom de cette ligne est « Ligne RD aval » (pour ligne rive droite aval) et un commentaire doit être inséré, contenant la date de mise en service de la ligne. Le filtre temporel doit être activé (valeur de `filtretemporel_actif` : 1) et sa valeur de 5000. Écrire la requête SQL permettant l'insertion de cette ligne.

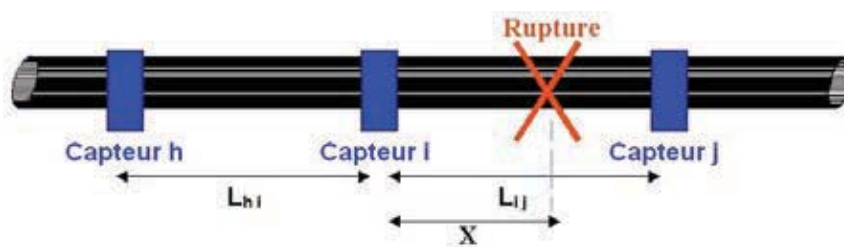
Question 40. Une rupture est détectée sur le pont d'identifiant `ouvrage_id = 5`. Écrire une requête qui renvoie l'adresse mail du gestionnaire responsable de cet ouvrage.

Traitement des données : localisation de la rupture

Lorsqu'une rupture se produit à un instant noté t_0 sur un fil, une onde acoustique se propage le long du toron. Chaque capteur C_i détecte le passage de l'onde à un instant noté t_i .

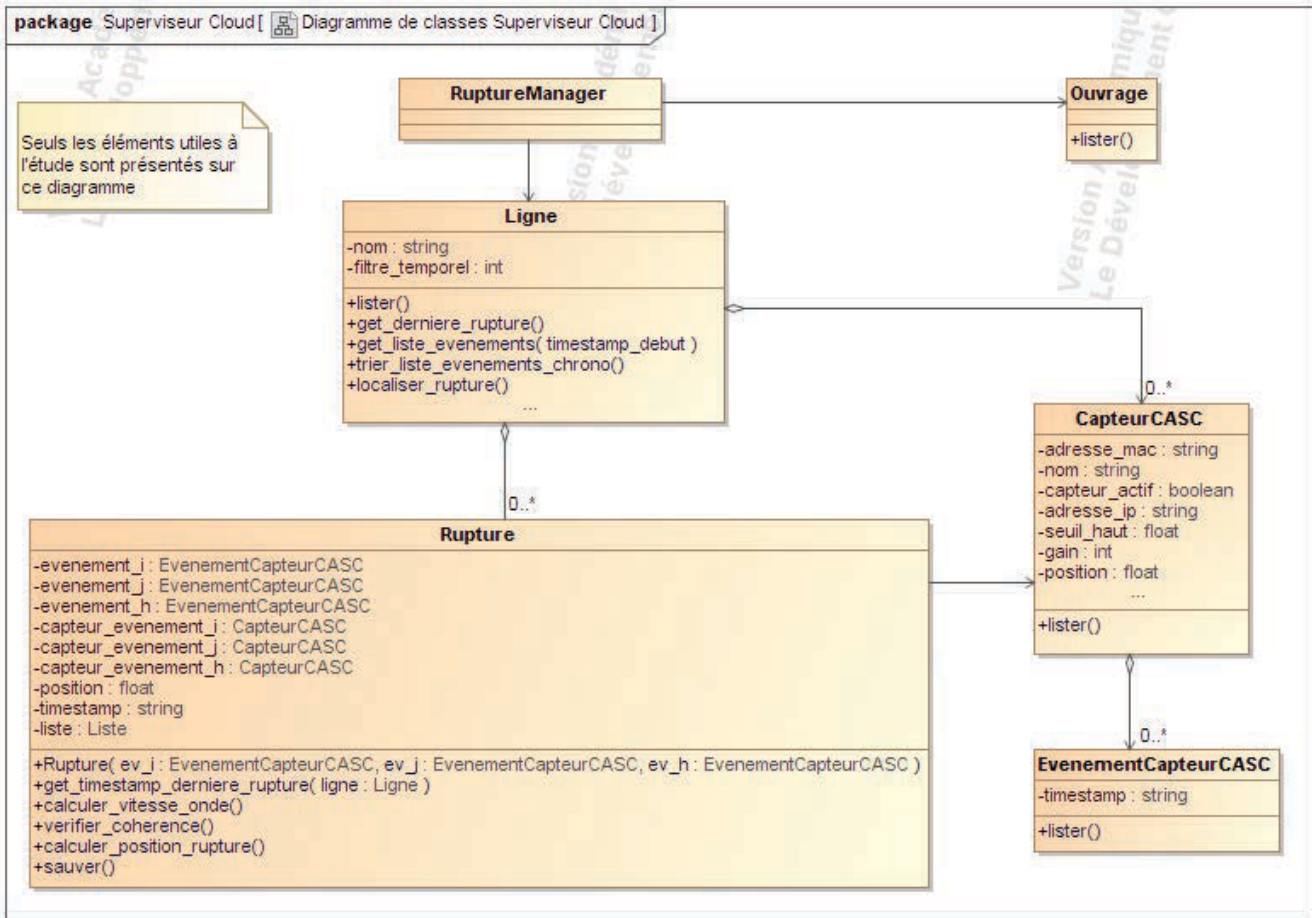


On note L_{ij} la longueur séparant deux capteurs C_i et C_j . Ces longueurs sont supposées voisines les unes des autres. On suppose que le capteur C_0 est situé à une extrémité de la chaîne de capteurs et qu'aucun d'eux n'est défaillant, si bien que tous détectent à des instants cohérents le passage de l'onde générée par la rupture d'un fil. On note X la distance entre le capteur le plus proche du capteur C_0 et le lieu de rupture du fil.



Sur la troisième génération du capteur CASC, le gestionnaire a accès à de nombreuses informations concernant le système et l'événement enregistré à partir de la base de données stockées dans le Cloud. Un programme spécifique, développé en Python, permet de visualiser les signaux enregistrés par chaque capteur. Les programmes associés à cette gestion et aux interfaces graphiques sont réalisés en programmation orientée objet.

Le diagramme de classes correspondant à la localisation d'une rupture est le suivant :



Question 41. Préciser le type de relation UML existant entre Ligne et Rupture.

On note V_{onde} la vitesse de propagation de l'onde dans le câble. Cette vitesse, en $\text{m} \cdot \text{s}^{-1}$, est supposée uniforme sur l'ensemble du câble.

Question 42. Donner l'expression de la longueur X en fonction de L_{ij} , V_{onde} et $\Delta t_{ij} = t_j - t_i$.

Lorsque trois événements sont détectés sur une ligne, le constructeur de la classe Rupture initialise une liste dont le format est le suivant :

$$\text{liste} : [[\text{id_capteur_h}, \text{th}, \text{Lh}], [\text{id_capteur_i}, \text{ti}, \text{Li}], [\text{id_capteur_j}, \text{tj}, \text{Lj}]]$$

avec :

- id_capteur_i est l'adresse mac du capteur C_i (type : str) ;
- t_i correspond à l'instant auquel le passage de l'onde a été détecté par le capteur C_i (type float) ;
- L_i correspond à la distance séparant le capteur C_i et le capteur C_0 (type : float).

L'étude est menée dans le cadre d'une numérotation correspondant à la figure de la page précédente.

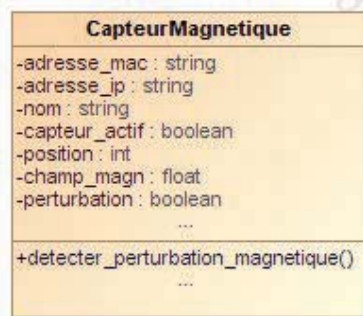
Les principales commandes de Python sont fournies dans le document technique DT2.

Question 43. Écrire en Python la méthode `calculer_position_rupture(self)` retournant une liste contenant l'adresse mac du capteur précédent le lieu de la rupture (capteur C_i), l'adresse mac du capteur situé après la rupture (capteur C_j), la distance X entre la rupture et le capteur C_i et la vitesse de l'onde.

Afin de contrôler les ruptures détectées mais aussi d'autres défauts présents dans les câbles en acier, il est possible d'effectuer une auscultation des torons à l'aide d'un capteur magnétique. Ce capteur est constitué d'un système magnétisant et d'un capteur détectant les perturbations de champ magnétique. Le système est alors déplacé le long des câbles à l'aide de montages de cordes et de poulies.

Afin de compléter les données recueillies par les capteurs CASC, le programme de traitement des données doit être modifié pour prendre en compte ces informations. La classe `CapteurMagnétique` est ajoutée au diagramme de classes précédent.

La description de cette classe est la suivante :



Question 44. Compléter le diagramme de classes du document réponse DR7 en appliquant les principes de la programmation orientée objet sur la généralisation.

Partie 6 – Synthèse

Objectif : gestion de projet

La mise en place d'un projet de l'envergure de l'application développée dans le laboratoire de l'IFSTTAR dans le cadre de la surveillance des haubans de ponts implique un grand nombre d'intervenants, tant du point de vue matériel que du point de vue logiciel. Ainsi, plusieurs personnes collaborent à l'élaboration des différents programmes indispensables à la gestion de l'acheminement, au stockage et au traitement des données recueillies sur le terrain ou au développement de l'interface graphique qui permettra au gestionnaire de consulter l'état des ouvrages surveillés.

Question 45. Indiquer en quelques lignes quelles sont les bonnes pratiques de programmation devant être adoptées lors d'un travail d'équipe permettant le développement et la mise en place d'une application telle que celle développée par le laboratoire.

Question 46. Après avoir rappelé les principales étapes d'un cycle de développement de projet en V, indiquer les raisons pour lesquelles la méthode AGILE est plus adaptée au développement d'un projet informatique.

Document technique DT1

Partie 1 – Documentation technique de la carte GPS (document sur 2 pages)

Copernicus GPS Receiver

Ultra-Thin, Low Power, Surface Mount GPS Module

Key Features and Benefits

THUMB NAIL-SIZED:
Just 2.54 mm THIN
19 mm W x 19 mm L

FAST MANUFACTURING
Tape & reel packaging;
Pick & place assembly;
Reflow solderable

HIGH PERFORMANCE
Low power usage: 93.9mW (typ)
Highly sensitive: -152 dBm
Fast TTFF (cold start): 39 sec

FLEXIBLE
Supports active or
passive antennas
NMEA, TSIP, TAIP protocols

TRIMBLE VALUE
High quality, low price
RoHS compliant

Drop-in Performance

Trimble's Copernicus™ GPS receiver delivers proven performance and Trimble quality for a new generation of position-enabled products. It features the Trimble revolutionary TrimCore™ software technology for extremely fast startup times and high performance in foliage canopy and urban canyon environments.

Designed for the demands of automated, high-volume production processes, the Copernicus module is a complete 12-channel GPS receiver in a 19mm x 19mm x 2.54mm, thumbnail-sized shielded unit. The small, thin, single-sided module is packaged in tape and reel for pick-and-place manufacturing processes; 28 reflow-solderable edge castellations provide interface to your design without costly I/O and RF connectors. Each module is manufactured and factory tested to Trimble's highest quality standards.

The ultra-sensitive Copernicus GPS receiver can acquire GPS satellite signals and generate fast position fixes with high accuracy in extremely challenging environments and under poor signal conditions. The module consumes typically 93.9 mW at full power with continuous tracking. The module is RoHS (lead-free) compliant.

The Copernicus GPS module is a complete drop-in, ready-to-go receiver that provides position,



The Copernicus GPS Surface Mount Module and Shield

velocity and time data in a user's choice of three protocols. Trimble's powerful TSIP protocol offers complete control over receiver operation and provides detailed satellite information. The TAIP protocol is an easy-to-use ASCII protocol designed specifically for track and trace applications. The bi-directional NMEA 0183 v3.0 protocol offers industry-standard data messages and a command set for easy interface to mapping software.

Applications

Compatible with active or passive antennas, the Copernicus GPS receiver is perfect for portable handheld, battery-powered applications. The receiver's small size and low power requirement make it ideal for use in Bluetooth appliances, sport accessories, personal navigators

or cameras and computer and communication peripherals, as well as vehicle tracking, navigation, and security applications.

Copernicus Starter Kit

The Copernicus Starter Kit provides everything you need to get started integrating state-of-the-art GPS capability into your application. The kit includes the reference interface board, which gives designers a visual layout of the Copernicus module on a PCB including the RF signal trace and RF connector, as well as the I/O connections of the 28 signal pins. Also included are a power converter, power adapter, GPS antennas, and the software for you to readily check out how easy it is to add Copernicus GPS to your application.

Trimble

Copernicus GPS Receiver

Ultra-Thin, Low Power, Surface Mount GPS Module

PERFORMANCE SPECIFICATIONS

General	L1 (1575.42 MHz) frequency, C/A code, 12-channel, continuous tracking receiver
Update Rate	TSIP @ 1 Hz; NMEA @ 1 Hz; TAIP @ 1 Hz
Accuracy	Horizontal: <3 meters (50%), <8 meters (90%) Altitude: <10 meters (50%), <16 meters (90%) Velocity: 0.06 m/sec
Acquisition (Autonomous Operation)	PPS (static): ±50 nanoseconds Reacquisition: 2 sec Hot Start: 9 sec Warm Start: 35 sec Cold Start: 39 sec Out of the Box: 41 sec
Sensitivity	Tracking: -152 dBm Acquisition: -142 dBm
Operational (COCOM) Limits Velocity	515 m/s

INTERFACE CHARACTERISTICS

Connectors	28 surface-mount edge castellations
Serial Port	2 serial ports
PPS	3.0 V CMOS-compatible, TTL-level pulse, once per second
Protocols	TSIP, TAIP, NMEA 0183 v3.0 Bi-directional NMEA messages Messages selectable by NMEA commands Selection stored in flash memory

ELECTRICAL CHARACTERISTICS

Prime Power	+2.7 VDC to 3.3 VDC
Power Consumption	(typ.) 30.7 mA (82.9 mW) @ 2.7 V (typ.) 31.3 mA (93.9 mW) @ 3.0 V
Backup Power	+2.7 VDC to +3.3 VDC
Ripple Noise	Max 50 mV, peak-to-peak from 1 Hz to 1 MHz

ENVIRONMENTAL SPECIFICATIONS

Operating Temperature	-40° C to +85° C
Storage Temperature	-55° C to +105° C
Vibration	0.008 g ² /Hz 5 Hz to 20 Hz 0.05 g ² /Hz 20 Hz to 100 Hz -3 dB/octave 100 Hz to 900 Hz
Operating Humidity	5% to 95% R.H. non-condensing, at +60° C

PHYSICAL CHARACTERISTICS

Enclosure	Metal shield
Dimensions	19 mm W x 19 mm L x 2.54 mm H (0.75" W x 0.75" L x 0.1" H)
Weight	1.7 grams (0.06 ounce) including shield

PINOUT ASSIGNMENTS

GND	1	26	GND
GND	2	27	GND
RF-IN	3	28	Reserved
GND	4	29	Reserved
LNA	5	30	TXD-B
Reserved	6	31	TXD-A
Open	7	32	Reserved
Short	8	33	RXD-A
Reserved	9	34	RXD-B
Boot	10	35	PPS
Xreset	11	36	Reserved
Vcc	12	37	Reserved
GND	13	38	Xstandby
GND	14	39	GND

ORDERING INFORMATION & ACCESSORIES

Module	Copernicus GPS Receiver Module, in metal enclosure Single modules Tape on reel (100 pieces) Tape on reel (500 pieces)
Reference Board	Copernicus GPS module mounted on a carrier board with I/O and RF connectors, including the RF circuitry with the antenna open detection, as well as antenna short detection and protection.
Starter Kit	Includes Copernicus Reference Board mounted on interface motherboard in a durable metal enclosure, AC/DC power converter, compact magnetic-mount GPS antenna, ultra-compact embedded antenna, serial interface cable, cigarette lighter adapter, TSIP, NMEA, and TAIP protocols, software toolkit and manual on CD-ROM

Ultra-Compact Embedded Antenna



3.3V active miniature unpackaged antenna
Cable length: 8 cm
Dim: 22 mm W x 21 mm L x 7.5 mm H
(0.866" x 0.827" x 0.295")
Connector: HFL

Compact Magnetic-Mount Antenna, MCX or SMA



3V active micropatch antenna with magnetic mount
Cable length: 5 m
Dim: 30.4 mm W x 35.5 mm L x 11.7 mm H
(1.197" x 1.398" x 0.461")
Connectors: MCX or SMA

Parts of this product are patent protected.

Specifications subject to change without notice.

Trimble Navigation Limited is not responsible for the operation or failure of operation of GPS satellites or the availability of GPS satellite signals.



Trimble Navigation Limited
Corporate Headquarters
935 Stewart Drive
Sunnyvale, CA 94085
1-800-787-4225
1-408-481-7741

Trimble Navigation Europe
Phone: +49-6142-2100-161

Trimble Export Ltd, Korea
Phone: 82-2-5555-361
korea_sales@trimble.com

Trimble Navigation Ltd,
China
Phone: 86-21-6391-7814



Document technique DT2

Partie 2 – Commandes de base en langage C/C++

(Document sur 2 pages)

La structure de contrôle if...else

Cette structure de contrôle permet d'exécuter une instruction si une condition est vraie, ou une autre instruction si elle est fausse.

```
if (condition) {
    instruction1 ;
    instruction2 ;
}
else {
    instruction3 ;
    instruction4 ;
}
```

Remarques

- Les accolades ne sont pas nécessaires s'il n'y a qu'une seule instruction dans le bloc.
- Le « else » est facultatif.

La boucle for

Le « for » est une structure de contrôle qui permet de répéter un certain nombre de fois une partie d'un programme.

```
for(instruction_init ; condition ; instruction_suivant)
instruction_repetée
```

Exemple

```
for(i=0 ; i<10 ; i++) {
    cout << "Bonjour ";
    cout << "le Monde" << endl;
}
```

La boucle while :

```
while (condition) instruction;
```

On teste la condition :

- si elle est vraie, on exécute l'instruction et on recommence.
- si elle est fausse, la boucle est terminée, on passe à l'instruction suivante.

L'instruction peut être une suite d'instructions entre accolades.

Exemple

```
int i = 0;
while (i < 10){
    cout << "La valeur de i est : " << i << endl;
    i++;
}
```

Partie 2 – Taille des différents types de données

Type de données	Signification	Taille en octets	Plage de valeurs acceptées
char	Caractère	1	-128 à 127
short int (ou short)	Entier court	2	- 32768 à 32767
unsigned short int (ou unsigned short)	Entier court non signé	2	0 à 65535
int	Entier	4	- 2147486648 à 2147486647
unsigned int	Entier non signé	4	0 à 4 294 967 295

Partie 4 – Quelques commandes de base en Python

Manipulation des listes en Python	
<code>L = [5, 8, 3, 4, 9]</code>	Définition de la liste L.
<code>L[0] = 5</code>	La numérotation des éléments d'une liste commence à 0.
<code>L = [[1, 3], [2, 4], [5, 7]]</code>	Définition d'une liste de listes.
<code>L[2][0] = 5</code>	Accès aux éléments de la liste de listes.

Programmation orientée objet en python	
<pre>def methode (self, v1): var = v1 * self.a1 return var</pre>	<p>Définition d'une méthode nommée <i>methode</i> utilisant les attributs de l'objet (<i>self</i>) ainsi que la variable <i>v1</i>.</p> <p>Le résultat du produit de la variable <i>v1</i> et de l'attribut <i>a1</i> de l'objet est stocké dans la variable <i>var</i> avant d'être retourné.</p>
<code>resultat = mon_objet.methode(v1)</code>	Stocke dans la variable <i>resultat</i> la variable retournée par la méthode <i>methode</i> .

Document technique DT3

Partie 2 – Code source des structures Message

(Document sur 2 pages)

```
#define MESSAGE_SIGNAL_BUFFER_SIZE 2000

typedef struct _MessageSignal MessageSignal;
struct _MessageSignal
{
    unsigned short heures;
    unsigned short minutes;
    unsigned short secondes;
    unsigned int micro_secondes;
    unsigned short gps;
    short buffer[MESSAGE_SIGNAL_BUFFER_SIZE];
};

typedef struct _MessageDemandMAJ MessageDemandMAJ;
struct _MessageDemandMAJ
{
    short wake_up_period;
    short trigger_level;
    short dummy; // sLowTriggerLevel
    short buffer_size;
    short enable;
};

typedef struct _MessageReceptionMAJ MessageReceptionMAJ;
struct _MessageReceptionMAJ
{
    short wake_up_period;
    short trigger_level;
    short dummy; // sLowTriggerLevel
    short buffer_size;
    short enable;
    short set_IP; // Bool
    short ip[4];
    short set_supervisor_IP; // Bool
    short IP_supervisor[4];
    short set_supervisor_port; // Bool
    short port_supervisor;
    short eval_wifi;
};

#define MESSAGE_ASCII_BUFFER_SIZE 100

typedef struct _MessageASCII MessageASCII;
struct _MessageASCII
{
    char buffer[MESSAGE_ASCII_BUFFER_SIZE];
};
```

```
//      Définition du Meta Message

typedef struct _Message Message;
struct _Message
{
    short id;
    short size;
    short sub_id;
    union {
        MessageSignal signal;
        MessageDemandMAJ maj;
        MessageReceptionMAJ maj_rep;
        MessageASCII ascii;
    } data;
};
```

Document technique DT4

Partie 2 - Masques des entêtes des données transmises entre le capteur CASC et le superviseur

(Document sur 2 pages)

Structure d'une trame Ethernet visualisée avec Wireshark (sans préambule ni CRC)

[Entête Ethernet (14 octets)][Entête IP (20 octets sans option)][Entête TCP][Données]

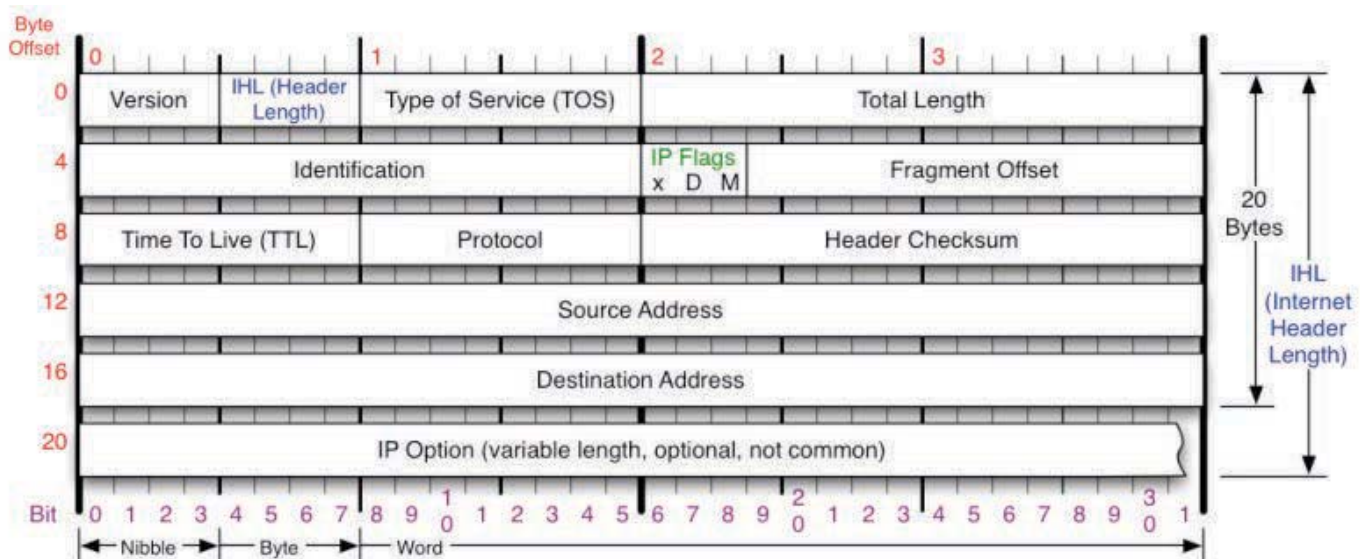
Entête Ethernet

[Adresse MAC destination (6 octets)][Adresse MAC source (6 octets)][Type (2 octets)]

Quelques types courants :

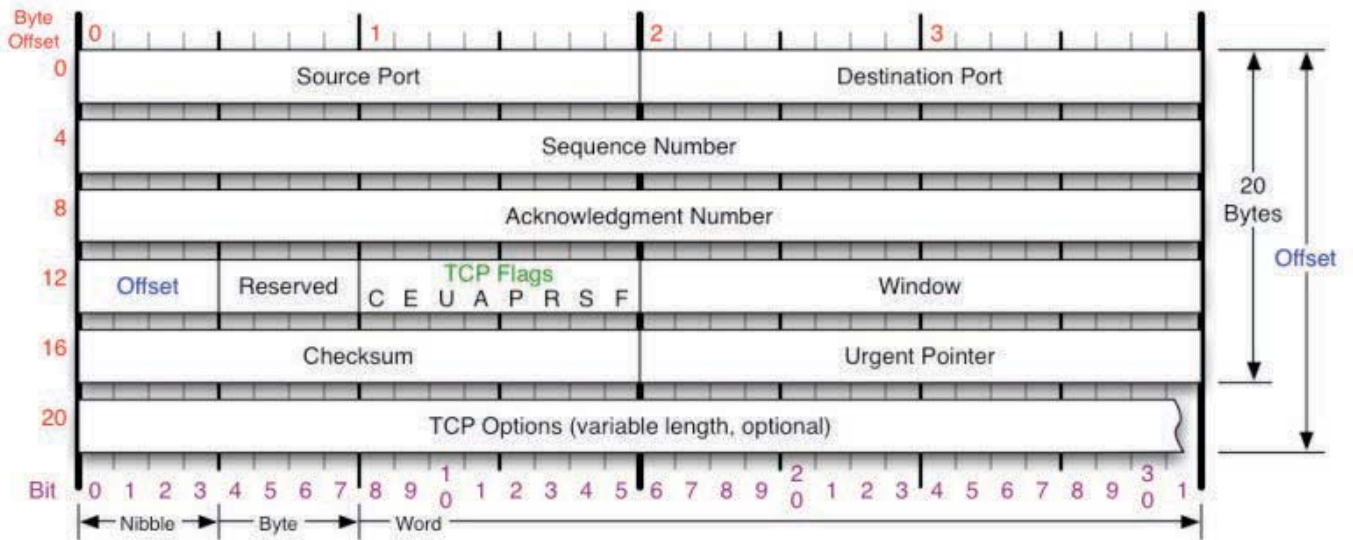
- 0x0800 : IPv4 (Internet Protocol version 4)
- 0x0806 : ARP (Address Resolution Protocol)

Entête IP



<p style="text-align: center;">Version</p> <p>Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.</p>	<p style="text-align: center;">Protocol</p> <p>IP Protocol ID. Including (but not limited to):</p> <table style="width: 100%; border: none;"> <tr> <td>1 ICMP</td> <td>17 UDP</td> <td>57 SKIP</td> </tr> <tr> <td>2 IGMP</td> <td>47 GRE</td> <td>88 EIGRP</td> </tr> <tr> <td>6 TCP</td> <td>50 ESP</td> <td>89 OSPF</td> </tr> <tr> <td>9 IGRP</td> <td>51 AH</td> <td>115 L2TP</td> </tr> </table>	1 ICMP	17 UDP	57 SKIP	2 IGMP	47 GRE	88 EIGRP	6 TCP	50 ESP	89 OSPF	9 IGRP	51 AH	115 L2TP	<p style="text-align: center;">Fragment Offset</p> <p>Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.</p>	<p style="text-align: center;">IP Flags</p> <p style="text-align: center;">x D M</p> <p>x 0x80 reserved (evil bit) D 0x40 Do Not Fragment M 0x20 More Fragments follow</p>
1 ICMP	17 UDP	57 SKIP													
2 IGMP	47 GRE	88 EIGRP													
6 TCP	50 ESP	89 OSPF													
9 IGRP	51 AH	115 L2TP													
<p style="text-align: center;">Header Length</p> <p>Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.</p>	<p style="text-align: center;">Total Length</p> <p>Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.</p>	<p style="text-align: center;">Header Checksum</p> <p>Checksum of entire IP header</p>	<p style="text-align: center;">RFC 791</p> <p>Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.</p>												

Entête TCP



TCP Flags

C E U A P R S F

Congestion Window

- C 0x80 Reduced (CWR)
- E 0x40 ECN Echo (ECE)
- U 0x20 Urgent
- A 0x10 Ack
- P 0x08 Push
- R 0x04 Reset
- S 0x02 Syn
- F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

- 0 End of Options List
- 1 No Operation (NOP, Pad)
- 2 Maximum segment size
- 3 Window Scale
- 4 Selective ACK ok
- 8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

Document technique DT5

Partie 2 – Communication défailante superviseur – capteur CASC

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Erase

No.	Time	Source	Destination	Protocol	Length	Info
1	0.960000000	192.168.0.11	192.168.0.20	TCP	74	[SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=58719379
2	0.197174000					:Ignored>
3	0.961008000					92 <Ignored>
4	0.997329000	192.168.0.11	192.168.0.20	TCP		74 [TCP R... sion] [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_F
5	1.093356000					90 <Ignored>
6	1.105851000					90 <Ignored>
7	1.725350000					92 <Ignored>
8	1.830138000					400 <Ignored>
9	1.834584000					591 <Ignored>
10	1.946861000					66 <Ignored>
11	2.533562000					84 <Ignored>
12	2.534047000					64 <Ignored>
13	2.630307000					84 <Ignor
14	2.630332000					64 <Ignored>
15	2.833559000					92 <Ignored>
16	3.001350000	192.168.0.11	192.168.0.20	TCP		74 [TCP Retransmission] [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_F
17	3.093358000					90 <Ignored>

▼ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

Interface id: 0
Encapsulation type: Ethernet (1)
Arrival Time: Jul 4, 2016 18:24:39.453267000 CEST

```

0000 14 fe b5 c5 b9 62 54 42 49 5a ed bd 08 00 45 00 .....bTB IZ....E.
0010 00 3c 81 c3 40 00 40 06 37 89 c0 a8 00 0b c0 a8 .<.@. 7.....
0020 00 14 84 6c 0f a1 cd 71 7d 50 00 00 00 a0 02 ...l...q }P.....
0030 72 10 81 9e 00 00 02 04 05 b4 04 02 08 0a 03 7f F.....
0040 fc 93 00 00 00 01 03 03 07 .....

```

File: ~/home/plegal/vsn/men/ca... Packets: 18 · Displayed: 18 (100,0%) · Ignored... Profile: Default

Document technique DT6

Partie 2 – Communication normale superviseur – capteur CASC

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Enregistrer

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.0.11	192.168.0.20	TCP	74	51220 > 4000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1996633 TSecr=0 WS=1
2	0.000883000	14:fe:b5:c5:b9:62	ff:ff:ff:ff:ff:ff	ARP	60	Who has 192.168.0.11? Tell 192.168.0.20
3	0.000906000	54:42:49:5a:ed:bd	14:fe:b5:c5:b9:62	ARP	42	192.168.0.11 is at 54:42:49:5a:ed:bd
4	0.001670000	192.168.0.20	192.168.0.11	TCP	74	4000 > 51220 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=9681725 TSecr=9685725
5	0.001780000	192.168.0.11	192.168.0.20	TCP	66	51220 > 4000 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1996634 TSecr=9685725
6	0.001725000	192.168.0.11	192.168.0.20	TCP	1514	51220 > 4000 [ACK] Seq=1 Ack=1 Win=29312 Len=1448 TSval=1996634 TSecr=9685725
7	0.001731000	192.168.0.11	192.168.0.20	TCP	1514	51220 > 4000 [ACK] Seq=1449 Ack=1 Win=29312 Len=1448 TSval=1996634 TSecr=9685725
8	0.002769000	192.168.0.11	192.168.0.20	TCP	1188	51220 > 4000 [FIN, PSH, ACK] Seq=2897 Ack=1 Win=29312 Len=1122 TSval=1996634 TSecr=9685725
9	0.002900000	192.168.0.20	192.168.0.11	TCP	66	4000 > 51220 [ACK] Seq=1 Ack=2897 Win=66560 Len=0 TSval=9685725 TSecr=1996634
10	0.003879000	192.168.0.20	192.168.0.11	TCP	66	4000 > 51220 [ACK] Seq=1 Ack=4020 Win=65280 Len=0 TSval=9685725 TSecr=1996634
11	0.007222000	192.168.0.20	192.168.0.11	TCP	66	4000 > 51220 [FIN, ACK] Seq=1 Ack=4020 Win=65280 Len=0 TSval=9685726 TSecr=1996634
12	0.007255000	192.168.0.11	192.168.0.20	TCP	66	51220 > 4000 [ACK] Seq=4020 Ack=2 Win=29312 Len=0 TSval=1996635 TSecr=9685726

▼ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Interface id: 0
Encapsulation type: Ethernet (1)
Arrival Time: Jul 5, 2016 14:22:15.329082000 CEST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1467721335.329082000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1

```

0000 14 fe b5 c5 b9 62 54 42 49 5a ed bd 08 00 45 00 .....bTB IZ....E.
0010 00 3c 16 d4 40 00 06 a2 78 c8 a8 00 0b c0 a8 <...@.@.X.....
0020 00 14 c8 14 0f a0 cb 2f 61 8d 00 00 00 a0 02 ...../ a.....
0030 72 10 81 9e 00 00 02 04 05 b4 04 02 08 0a 00 1e r.....
0040 77 59 00 00 00 01 03 03 07 wY.....

```

File: "home/plegal/svn/men/ca... Packets: 12 - Displayed: 12 (100.0%) - Load ti... Profile: Default

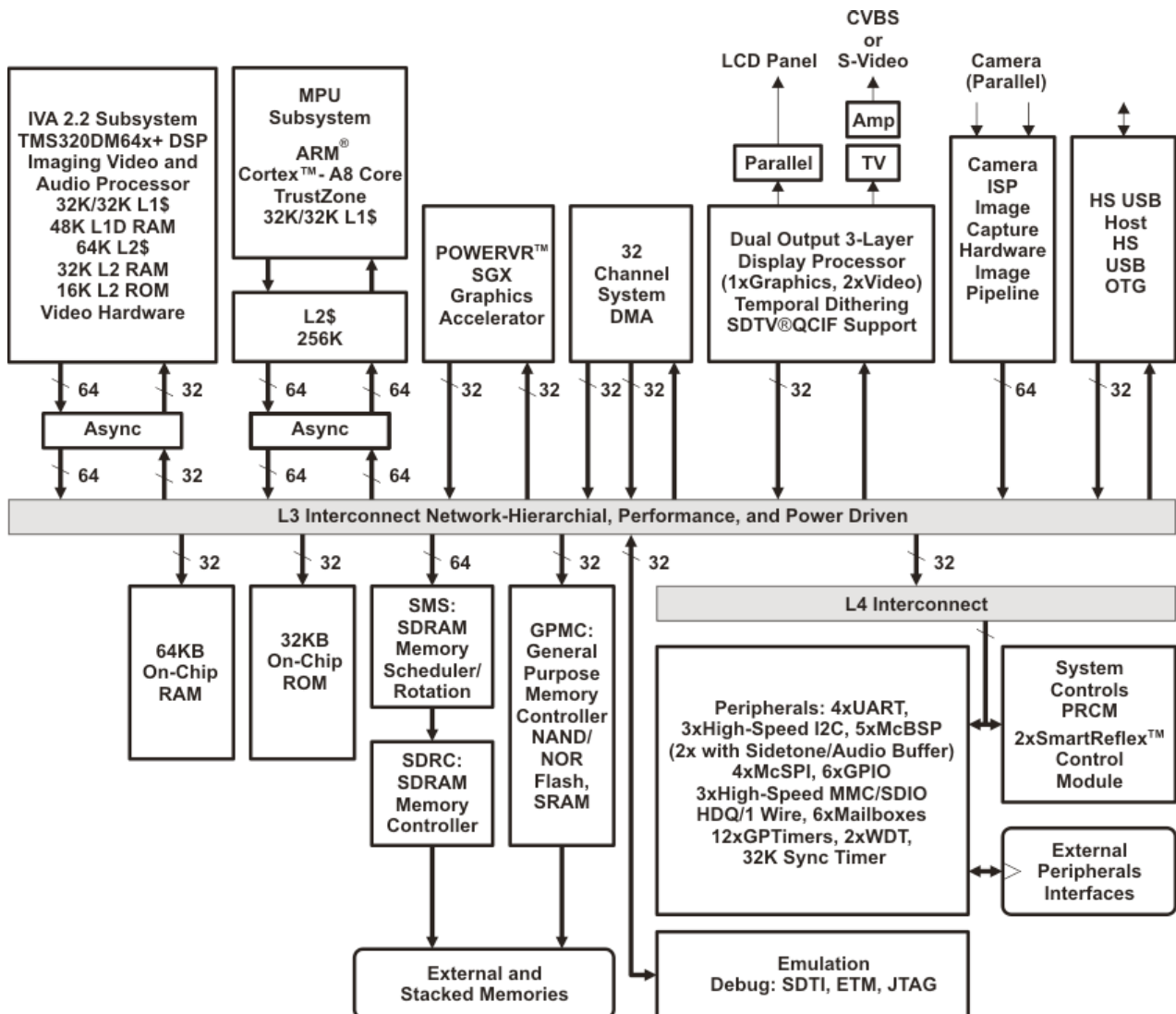
Document technique DT7

Partie 3 – Documentation technique du processeur de la carte Pégase 2

Processeur DM3730 Digital Media Processor



Diagramme fonctionnel

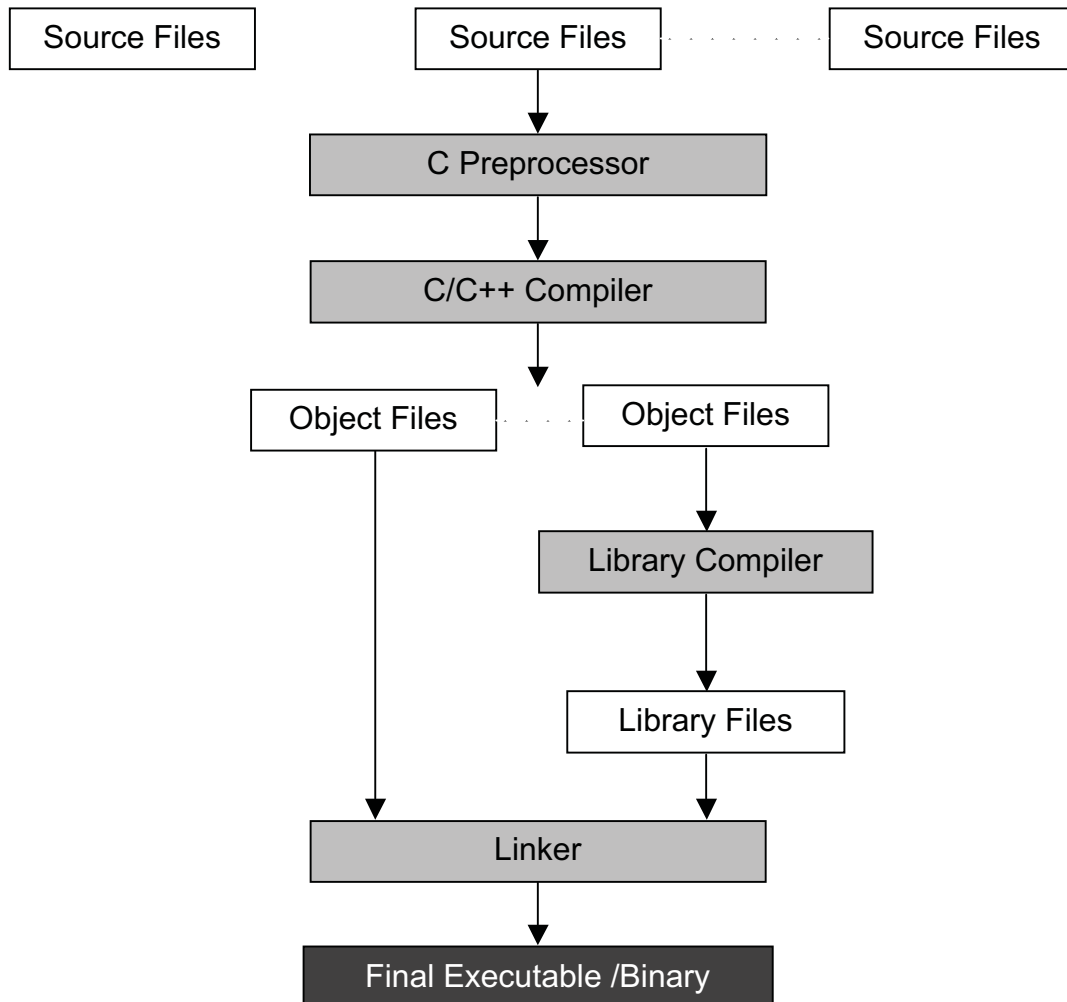


Cortex-A8 Processor

The ARM® Cortex®-A8 was introduced in 2005 and was the first processor supporting the ARMv7-A architecture. It has enabled numerous mobile and embedded designs and accelerated the early development of the smartphone market and further entrenched the primacy of ARM compatible ecosystem software and tools. ARMv7-A incorporates the ARM NEON™ single instruction multiple data (SIMD) engine, ARM TrustZone® security extensions, and the Thumb-2 instruction set for reduced code size.

Document technique DT8

Partie 3 – Phases de compilation d'un programme C/C++



Document technique DT9

Partie 3 – Script shell de démarrage d'un capteur CASC

```

1 #! /bin/sh
2
3 # Set date from RTC
4 hwclock -s
5 # Set loopback interface
6 ifconfig lo 127.0.0.1
7 # Load GPIO driver
8 modprobe ifsttar_pflags
9 # Load kernel class to drive led
10 modprobe leds-ifsttar
11 ## Uncomment if you use ADC over DMA SPI
12 modprobe ifsttar_can_spi
13 # Set GPS in low power mode
14 # Reset (active: 0, inactive:1) don't change
15 echo 1 > /dev/pg1
16 # Standby (active: 0, inactive:1) change to turn on/off gps, default is off
17 echo 0 > /dev/pg0
18 # Load Wireless driver
19 modprobe libertas_cs
20 sleep 3
21 # Configure layer 2 OSI for wlan0 interface
22 iwconfig wlan0 essid AP_SSID_AP1
23 # Configure layer 3 OSI for wlan0 interface
24 dhclient wlan0
25 sleep 3
26 # Set Board hostname
27 hostname PEGASE-`cat /sys/class/net/wlan0/address`
28 counter=0
29 while [ $counter -lt 10 ]; do
30     let counter=counter+1
31     # 3 ping
32     ping -c 3 casc.ifsttar.fr
33     if [ $? -eq 0 ]; then
34         # Start ftp, ssh, telnet server
35         inetd &
36         # Start CASC
37         Casc &
38     fi
39 done
40 # reboot if max count
41 if [ $counter -eq 10 ]; then
42     reboot
43 fi
44 # syntaxe bash
45 # -eq : ==
46 # -lt : <
47 # $? : code réponse commande précédente (0 -> OK, !=0 -> KO)

```

Document technique DT10

Partie 3 – Commandes Linux relatives aux droits d'accès des fichiers

Plusieurs types d'utilisateurs peuvent être définis sous Linux :

- utilisateur (user ou u), propriétaire du fichier;
- le groupe d'utilisateurs (group ou g), qui correspond à un groupe de un ou plusieurs utilisateurs ;
- les autres utilisateurs (other ou o), qui n'appartiennent pas au groupe d'utilisateur du fichier.

La façon la plus courante de consulter les permissions d'un fichier sous Linux est la commande `ls -l`. Les permissions s'affichent alors sous la forme suivante :

```

drwxr-xr-x 2 sammy sammy 4096 Nov 10 16:49 publicly_accessible_directory
  
```

Le diagramme associe les étiquettes suivantes aux champs de la ligne de commande :

- Mode** : drwxr-xr-x
- Owner** : 2
- Group** : sammy
- File Size** : 4096
- Last Modified** : Nov 10 16:49
- Filename** : publicly_accessible_directory

Les permissions accordées pour le fichier `publicly_accessible_directory` sont :

- `rw` : lecture (lettre `r`), écriture (lettre `w`) et exécution (lettre `x`) pour le propriétaire;
- `r-x` : lecture (lettre `r`), aucun droit en écriture (signe `-`), droit d'exécution (lettre `x`) pour le groupe d'utilisateurs;
- `r-x` : lecture (lettre `r`), aucun droit en écriture (signe `-`), droit d'exécution (lettre `x`) pour les autres utilisateurs.

Les accès en droit sont affectés d'un nombre :

- 4 pour la permission en lecture (read) ;
- 2 pour la permission en écriture (write) ;
- 1 pour la permission en exécution (execute).

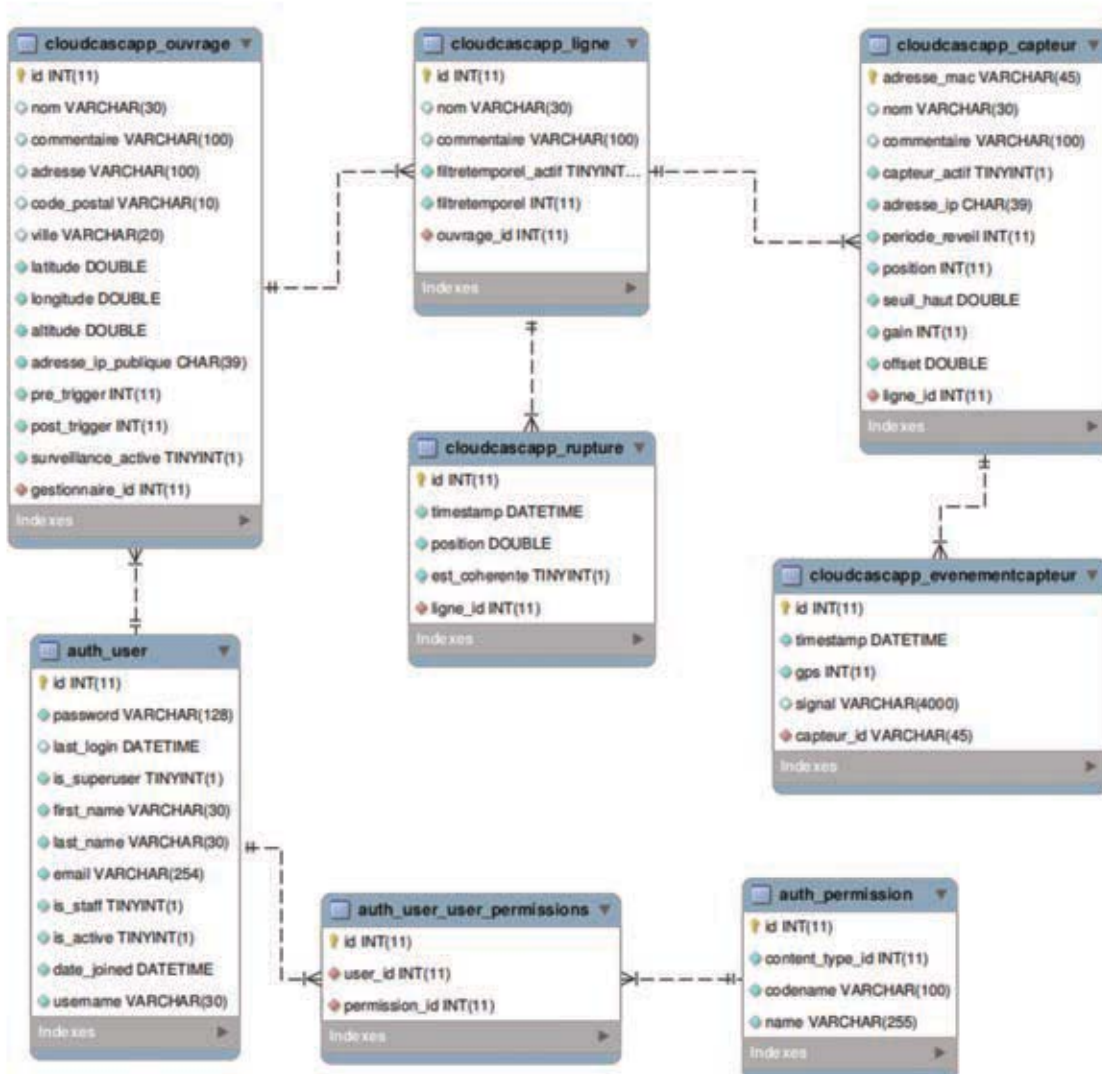
Ainsi, un total de 5 indique que les permissions en lecture et en exécution sont accordées.

Pour modifier les permissions des différents utilisateurs, la commande « `chmod` » peut être utilisée, avec diverses syntaxes :

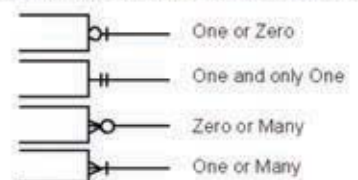
- « `chmod abc filename` », où `a`, `b` et `c` correspondent aux totaux des permissions respectives accordées au propriétaire, au groupe et aux autres utilisateurs, dans cet ordre ;
- « `chmod u = ..., g = ..., o = ... filename` » où les points de suspensions doivent être remplacés par les lettres correspondant aux types de permission accordées (par exemple, `rx` pour une permission en lecture et en exécution) ;
- « `chmod ugo+r filename` » qui ajoute aux permissions déjà données aux trois types d'utilisateurs (user, group et owner) la permission en lecture (lettre `r`).

Document technique DT11

Partie 4 – Tables stockant les informations relatives à la gestion de n ponts



Summary of Crow's Foot Notation

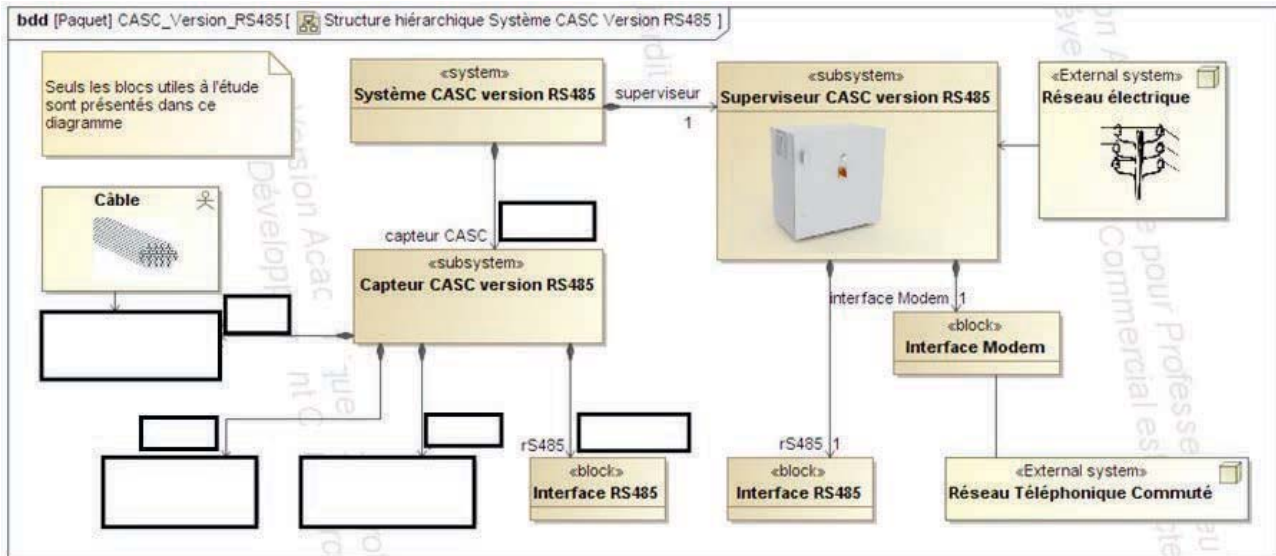


NE RIEN ECRIRE DANS CE CADRE

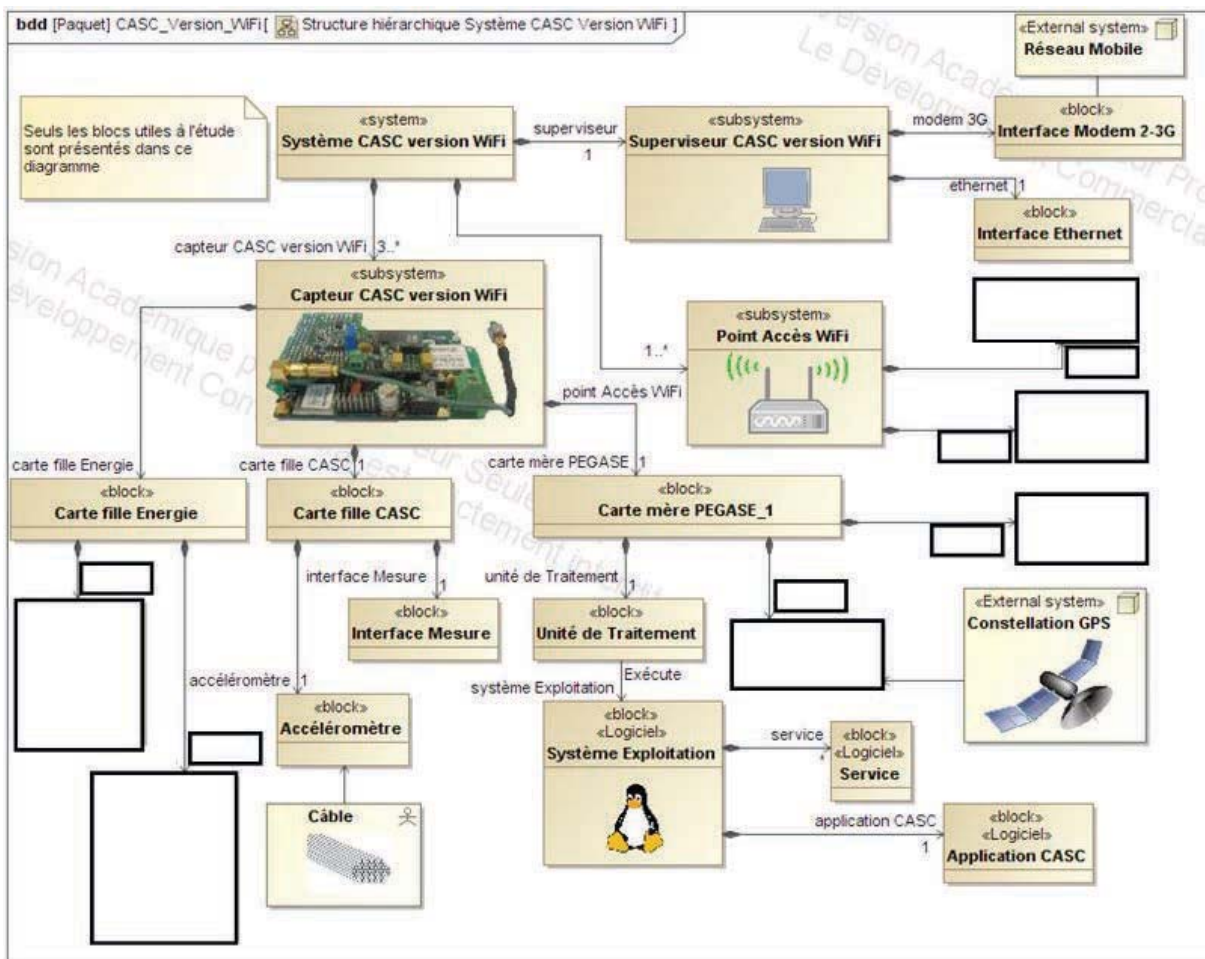
Document réponse DR1

Partie 1 – Étude des différentes générations de capteur CASC

Question 1



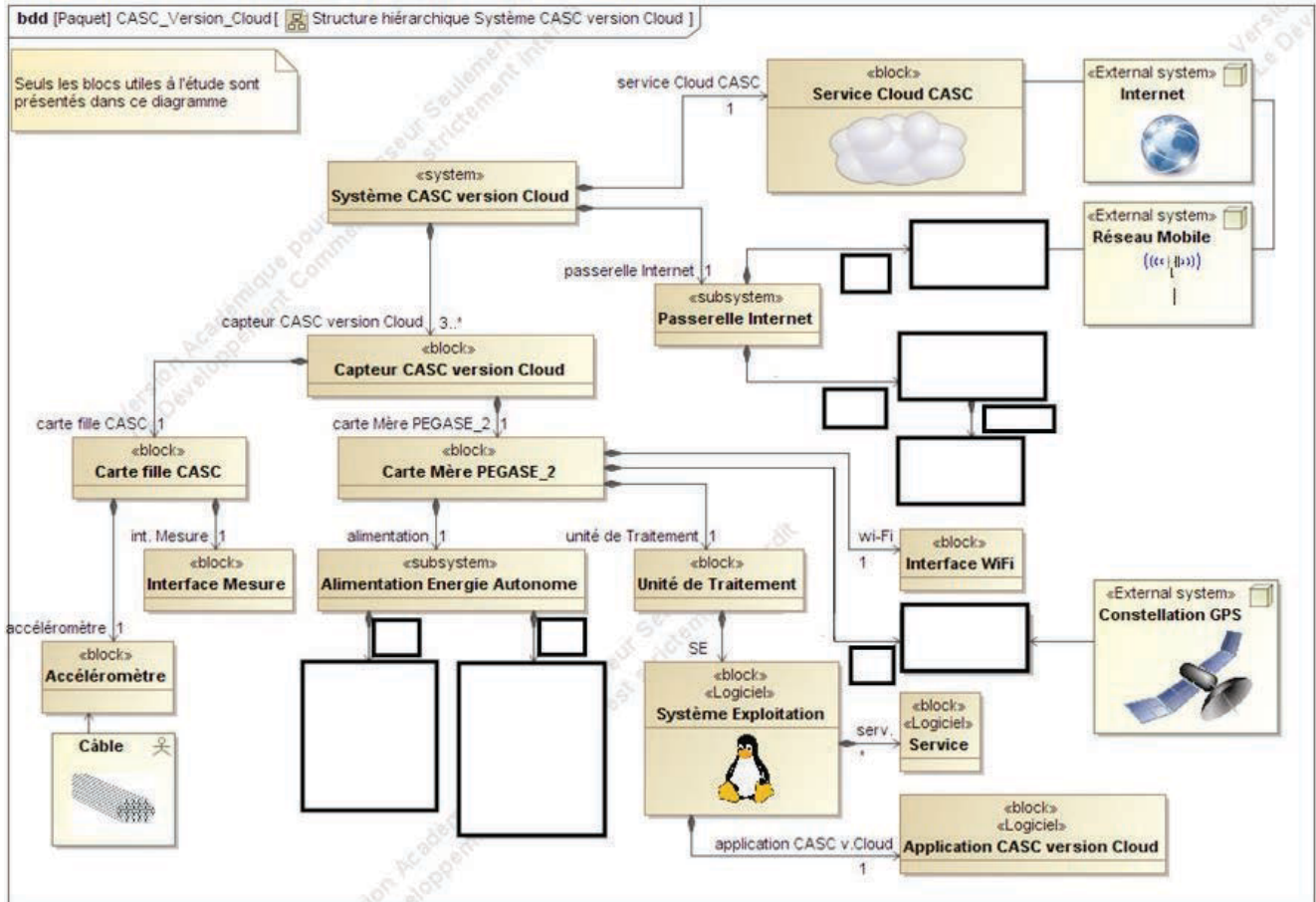
Question 6



Document réponse DR2

Partie 1 – Étude des différentes générations de capteur CASC

Question 7



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

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--



Prénom(s) :

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

**Numéro
Inscription :**

--	--	--	--	--	--	--	--	--	--	--	--

Né(e) le :

		/			/						
--	--	---	--	--	---	--	--	--	--	--	--

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

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

Concours / Examen : **Section/S 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.
- Numéroté 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.

EDE NUM 2

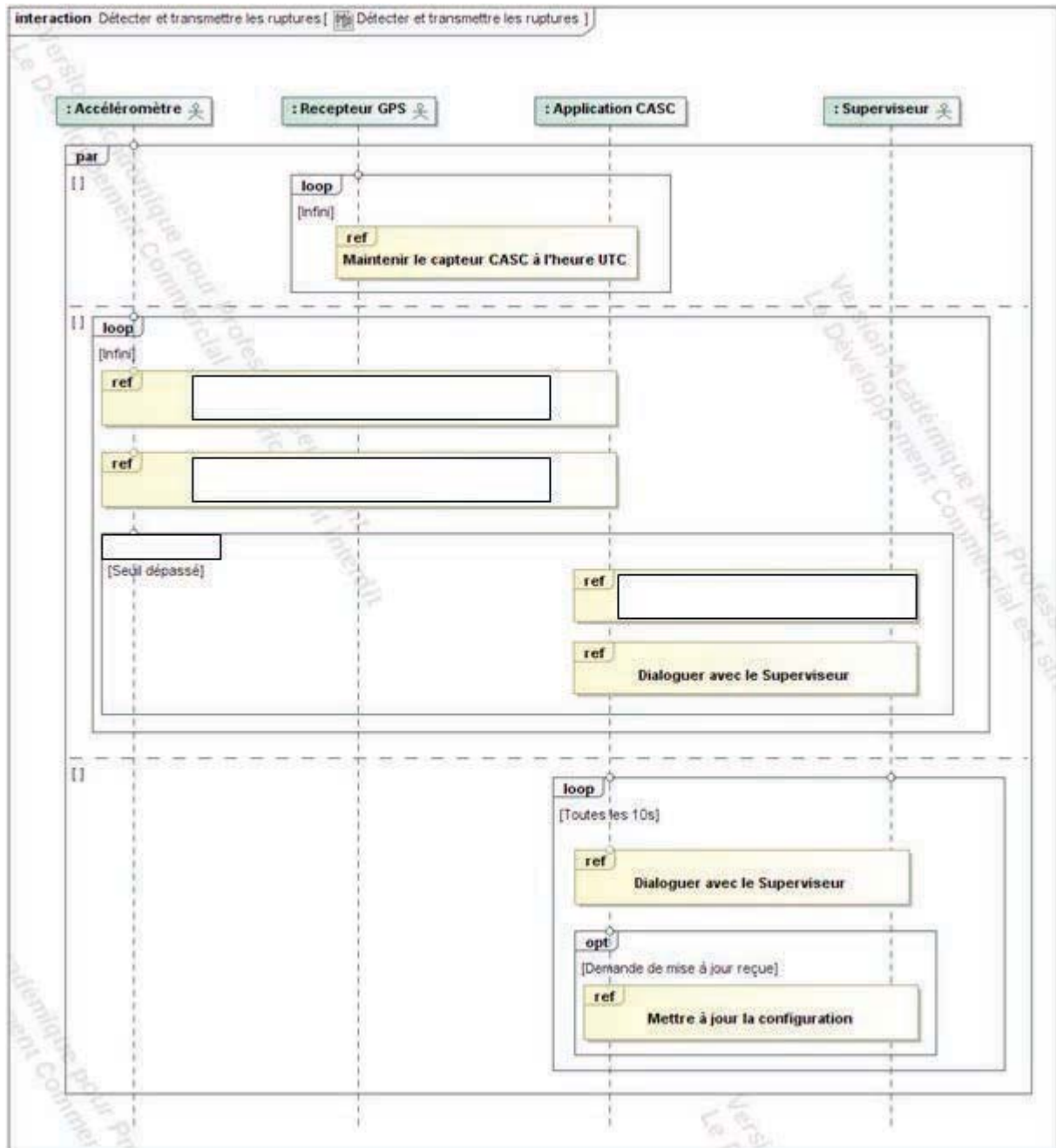
DR3 - DR4

NE RIEN ECRIRE DANS CE CADRE

Document réponse DR3

Partie 2 – Transmission des données du capteur Casc vers le superviseur

Question 8



Document réponse DR4

Partie 2 – Méthode AnalyserBufferSPI()

Question 10

On définit les constantes suivantes :

```
#define PRE_TRIGGER      250
#define POST_TRIGGER    1750
#define NOMBRE_ECHANTILLONS_TOTAL  2000

void GestionnaireSPI::AnalyserBufferSPI(short *buffer, unsigned int buffer_size)
{
    // Variables de parcours du tableau lu
    unsigned int index = 0;

    // Variables pour la détection du signal de rupture
    short buffer_signal[NOMBRE_ECHANTILLONS_TOTAL];
    int heure, minute, seconde, microseconde, gps;

    // Parcours et analyse le nouveau tableau acquis à partir de 'index'
    short seuil_haut = this->gestionnaire_trames->GetSeuilHaut();
```

Nom de famille :

(Suivi, s'il y a lieu, du nom d'usage)

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--



Prénom(s) :

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

**Numéro
Inscription :**

--	--	--	--	--	--	--	--	--	--	--

Né(e) le :

		/			/				
--	--	---	--	--	---	--	--	--	--

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

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

Concours / Examen : **Section/S 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.
- Numéroté 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.

EDE NUM 2


DR5 - DR6 - DR7

NE RIEN ECRIRE DANS CE CADRE


Document réponse DR5

Partie 2 – Transmission des données du capteur Casc vers le superviseur

Question 17



Capteur
CASC



Superviseur
CASC

Port TCP						
Adresse IP						
Adresse MAC						

Port TCP						
Adresse IP						
Adresse MAC						

Partie 3 – Chaîne de compilation : format des différents fichiers générés

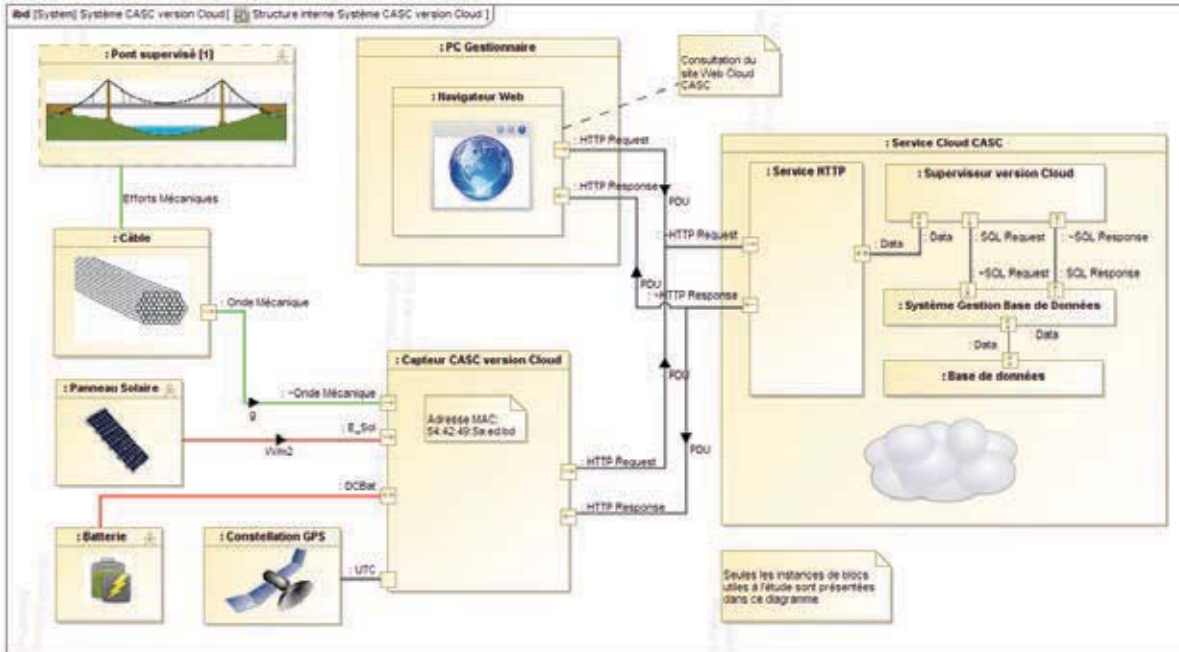
Question 24

Type de fichier	Contenu du fichier (texte : code ASCII)	Contenu du fichier (binaire : code machine)	Utilisé en entrée de la phase de compilation	Utilisé en entrée de la phase d'édition de lien
GestionnaireTrames.cpp				
GestionnaireTrames.o				
GestionnaireTrames.h				
libboost_system.so				
build/Release/bin/Casc				

Document réponse DR6

Partie 3 – Cheminement des données

Question 29



Partie 3 – Requête HTTP

Question 30

Champs	Explications
POST	
/cloudcascapp/ evenement_casc/	
HTTP/1.1	
Host: casc.ifsstar.fr	
Content- Length:16241	
timestamp=...	

Document réponse DR7

Partie 4 – Diagramme de classes : ajout du capteur magnétique

Question 44

