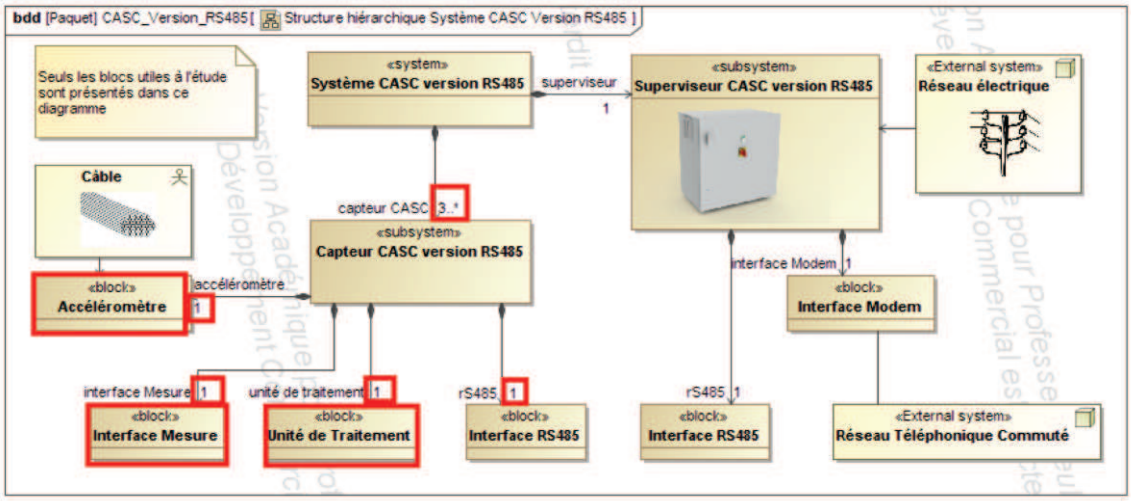
Éléments de correction de l'épreuve « Étude d'un système, d'un procédé ou d'une organisation » – option ingénierie informatique

Question 1



Question 2

Avantages RS485 : liaison filaire différentielle garantissant une bonne immunité au bruit sur une longue distance.

Inconvénients RS485 : coût de la liaison filaire (des kilomètres de fil de cuivre). Difficulté d'installation.

Avantages Wifi : liaison radio. Facilité d'installation et d'utilisation.

Inconvénients Wifi : risque de perturbation radio. Sécurité à garantir.

Justification : la transmission Wifi est moins coûteuse, plus facile à installer et à maintenir que l'installation RS485.

Question 3

La méthode CSMA/CD fait partie des méthodes non persistantes. Si l'émetteur constate que le média est occupé, il ne reste pas à l'écoute en attendant qu'il soit libre, mais réessaie ultérieurement. Dans une méthode CSMA/CD, l'interface réseau reste à l'écoute pendant l'émission de la trame, au cas où une autre trame serait émise d'un autre point du réseau alors que l'émission a commencé. Le temps d'écoute après émission doit être au moins égal à deux fois celui nécessaire pour transmettre la trame entre les points les plus éloignés du réseau. Si une autre émission est détectée, chaque émetteur arrête son émission et attend pendant une durée différente puis émet de nouveau son message.

Dans le cas d'une transmission par Wifi, l'interface ne peut écouter et émettre simultanément, d'où l'utilisation de la méthode CSMA/CA. Dans ce cas, les conflits d'accès au média sont évités avant l'émission d'un message. L'émetteur envoie un premier message indiquant qu'il va émettre des données. L'inconvénient de ces méthodes est qu'elles ne sont pas déterministes (voir question suivante).

Question 4

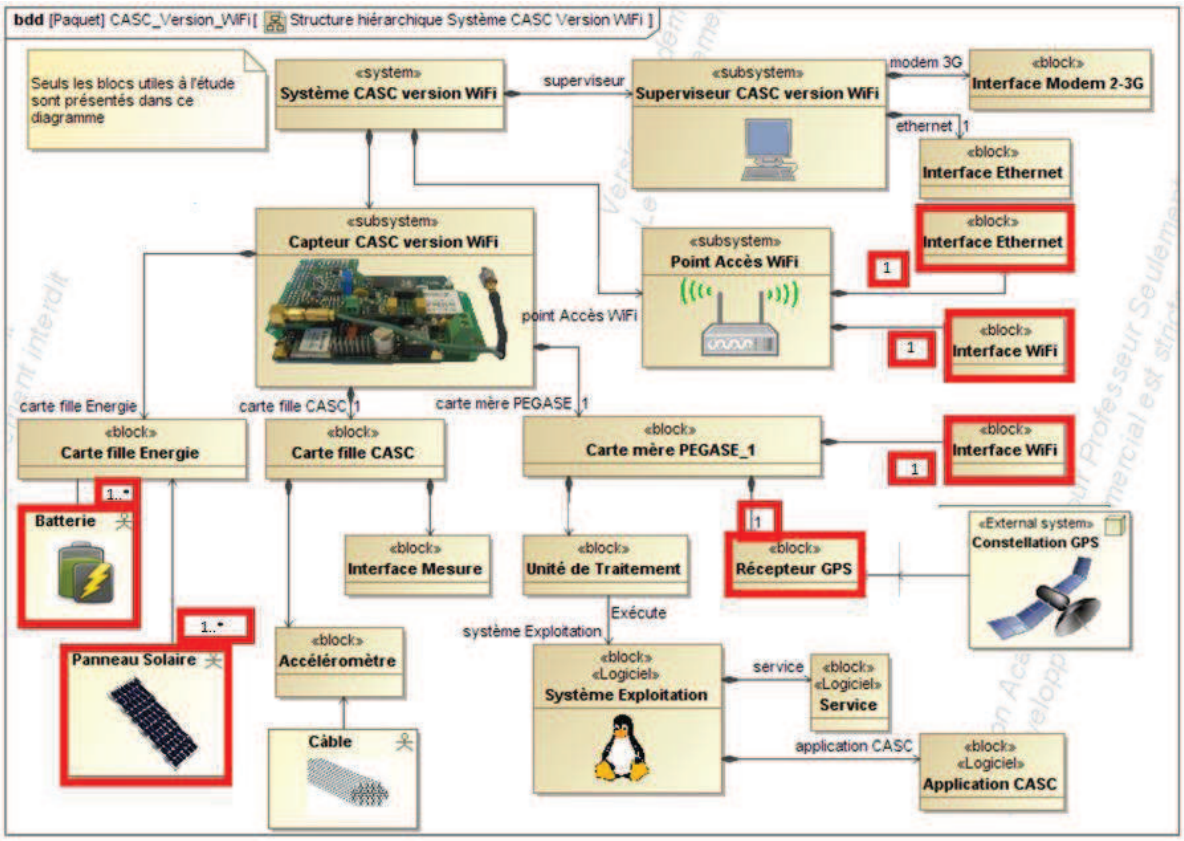
La communication par bus RS485 avec un protocole Maître-Esclave est déterministe, c'est-à-dire qu'elle permet un fonctionnement en temps réel, donc une datation du signal émis par synchronisation

de tous les capteurs sur une même horloge. La méthode CSMA/CA implique que l'émission peut être retardée aléatoirement, si bien que la datation n'est plus possible du fait que l'on ne peut pas garantir que les capteurs aient la même horloge (qui doit être précise à quelques micro-secondes dans le cas de l'application CASC).

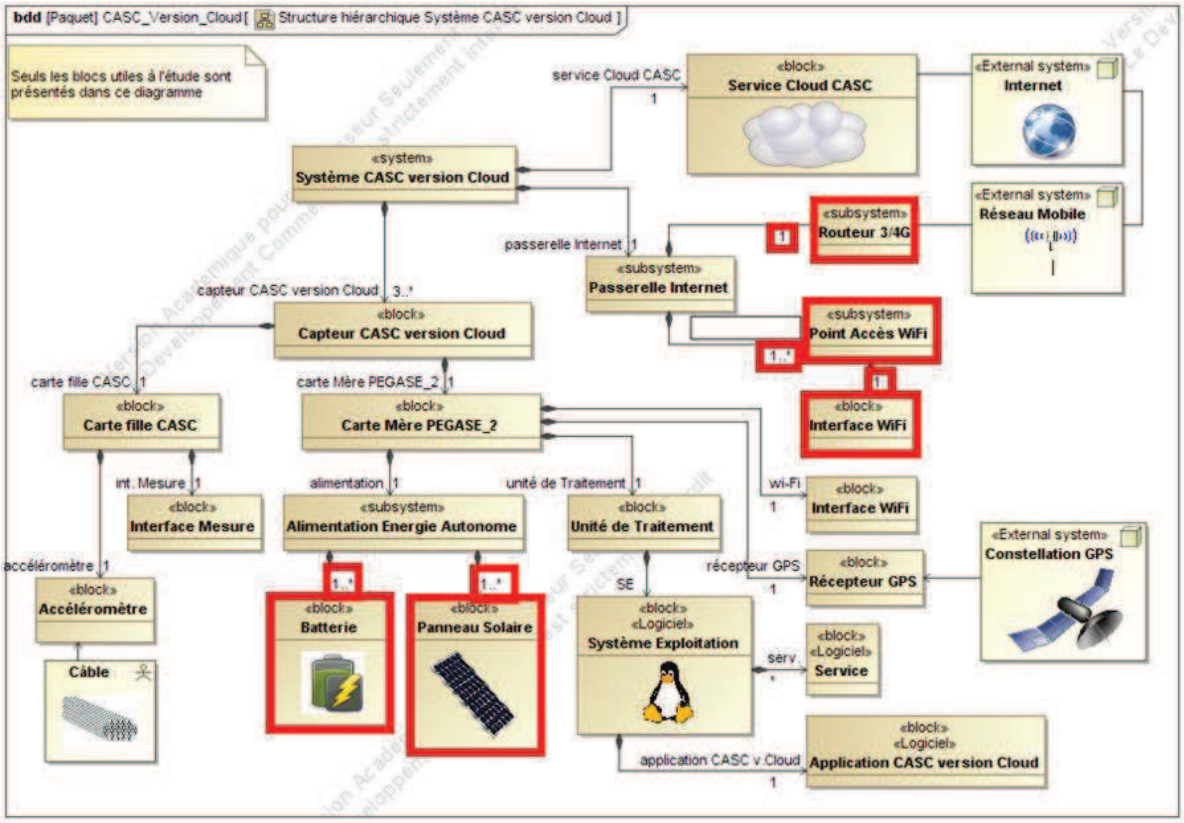
Question 5

Si la présence d'une interface GPS permet la localisation, elle permet aussi d'avoir l'heure absolue, donc la datation des événements détectés (signal PPS – Pulse Per Second – précis à +/-50 ns).

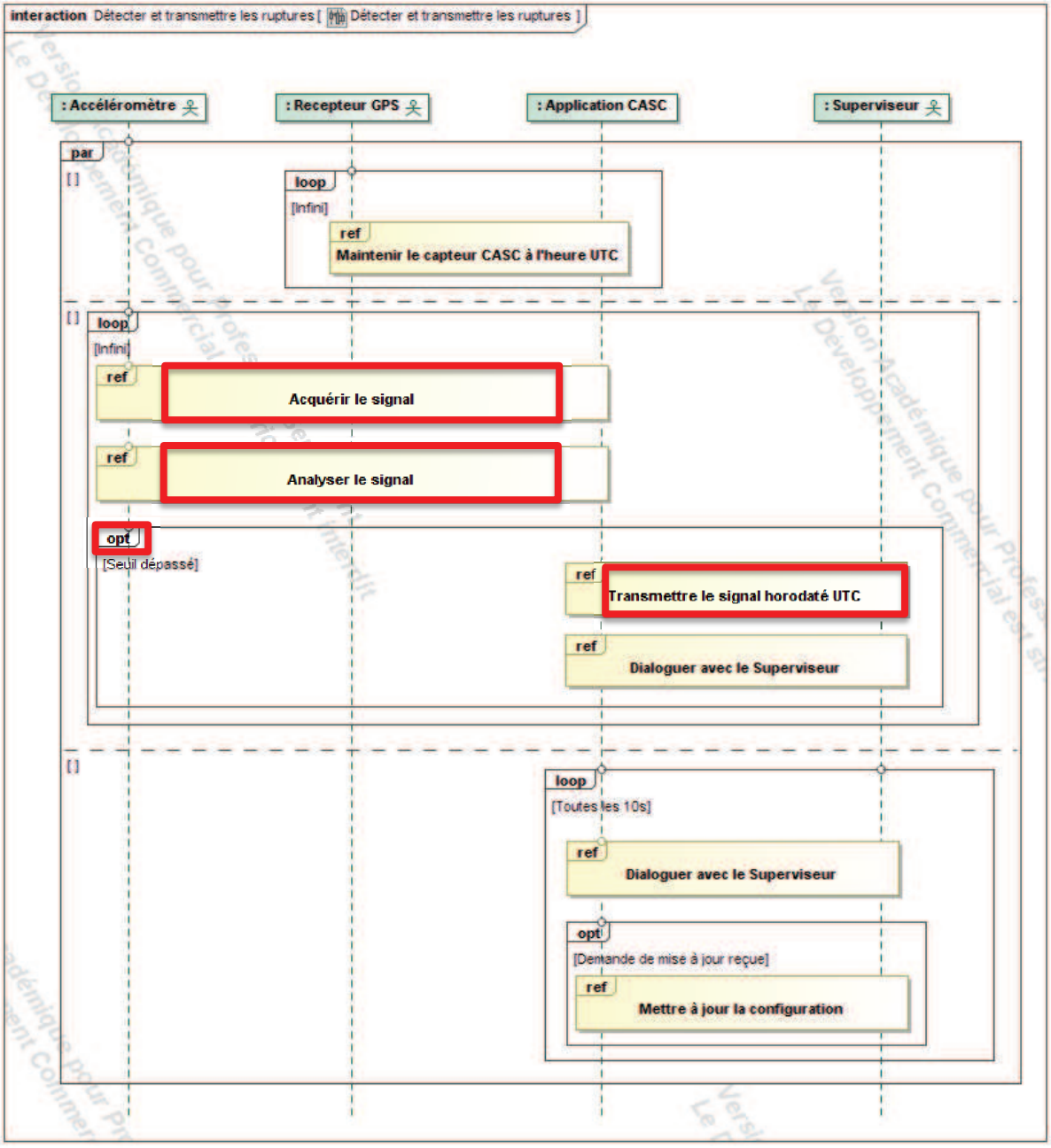
Question 6



Question 7


bdd [Paquet] CASC_Version_Cloud [Structure hiérarchique Système CASC version Cloud]
Seuls les blocs utiles à l'étude sont présentés dans ce diagramme
service Cloud CASC
1
«block»
Service Cloud CASC
«External system»
Internet
«system»
Système CASC version Cloud
«External system»
Réseau Mobile
«subsystem»
Routeur 3/4G
passerelle Internet
1
«subsystem»
Passerelle Internet
capteur CASC version Cloud
3..*
«block»
Capteur CASC version Cloud
«subsystem»
Point Accès WiFi
1..*
carte fille CASC
1
carte Mère PEGASE_2
1
«block»
Carte fille CASC
«block»
Carte Mère PEGASE_2
«block»
Interface WiFi
int. Mesure
1
alimentation
1
unité de Traitement
1
wi-Fi
1
«block»
Interface Mesure
«subsystem»
Alimentation Energie Autonome
«block»
Unité de Traitement
«block»
Interface WiFi
«External system»
Constellation GPS
accéléromètre
1
SE
récepteur GPS
1
«block»
Récepteur GPS
«block»
Accéléromètre
1..*
1..*
«block»
«Logiciel»
Système Exploitation
«block»
«Logiciel»
Service
serv.
*
Câble
«block»
Batterie
«block»
Panneau Solaire
application CASC v.Cloud
1
«block»
«Logiciel»
Application CASC version Cloud

Question 8



Question 9

Relation d'association : la classe GestionnaireSPI doit pouvoir utiliser la méthode MessageAddSignal() de la classe GestionnaireTrames.

Question 10

```
void GestionnaireSPI::AnalyserBufferSPI(short *buffer, unsigned int buffer_size)
{
        Donné dans l'énoncé…

        for (index=0; index<buffer_size; index++) {
                // Si dépassement du seuil haut
                if ((buffer[index] >= seuil_haut) || (buffer[index] <= 0 - seuil_haut)) {
                // Datation absolue du signal au dépassement de seuil
```

```
                this->CalculHeureAuDepSeuil (index, &heure, &minute, &seconde, &microseconde,
&gps);

                // Recopie bloc à bloc les données dans le signal à envoyer
                memcpy (buffer_signal, buffer + index – PRE_TRIGGER + 1, sizeof(short) *
(PRE_TRIGGER + POST_TRIGGER));

                 // Autre solution
                for (int i=0; i < NOMBRE_ECHANTILLONS_TOTAL; i++)
                 buffer_signal[i] = buffer[index – PRE_TRIGGER + 1 + i];

        // Passe le signal évalué au Gestionnaire de trames
        this->gestionnaire_trames->MessageAddSignal (heure, minute, seconde, microseconde, gps,
buffer_signal, NOMBRE_ECHANTILLONS_TOTAL * sizeof(short));

        // Avance l'index de recherche après ce signal
        index += POST_TRIGGER;
                }
        }
}
```

Question 11

MUTEX ou sémaphore booléen : lock et unlock en début et fin de construction de la trame qui est une zone critique.

Question 12

L'allocation dynamique (new et delete) se fait dans le tas.
Il peut y avoir des fuites mémoires (tas plein) entraînant un « plantage » de l'application.
Libérer chaque allocation mémoire après son utilisation (un new = un delete).

Question 13

UDP : pas de contrôle des données donc plus rapide.
TCP : contrôle des données donc plus de sécurité sur la transmission.
Choix de la fiabilité des transmissions (vies humaines en jeu).

Question 14

Message est une union des différents types de message :
MessageSignal : 4 unsigned short + 1 unsigned int + 2000 unsigned shorts soit
        4×2 + 1×4 + 2000×2 = 4012 octets
MessageASCII : 100 char soit 1×100 = 100 octets
MessageDemandeMAJ : 5 short soit 5×2 = 10 octets
MessageReceptionMAJ : 12 short soit 12×2 = 24 octets
Le compilateur réservera la taille de 3 short (id, size et sub_id) plus la taille de la structure la plus grande (MessageSignal) pour un message (union).
La taille maximale d'un message est donc de : 3*2 + 4012 = 4018 octets.

Question 15

Capteur CASC : client – Superviseur : serveur.

Question 16

Pas d'acquittement lors de la demande de synchronisation.

Le port d'écoute du superviseur est sur 4001 au lieu du port 4000 annoncé en configuration dans l'énoncé.

Question 17

Capteur CASC : Port TCP : 51220 (c814 en hexadécimal) Adresse IP : 192.168.0.11 Adresse MAC : 54 : 42 : 49 : 5a : ed : bd	Superviseur Port TCP : 4000 (0fa0 en hexadécimal) Adresse IP : 192.168.0.20 Adresse MAC : 14 : fe : b5 : c5 : b9 : 62

Question 18

Trames 2 et 3 correspondant à la recherche de l'adresse MAC du destinataire (ARP : Address Resolution Protocol) pour amorcer la communication.

Question 19

Trame 1 : demande de communication (Synchronisation SYN) de la part du client.

Trame 4 : acquittement de la demande SYN de la part du superviseur.

Trame 5 : acquittement de l'acquittement de la part du client.

Question 20

Début du message : $(0E\ 00)_{16\ \text{Little Indian}} = (00\ 0E)_{16} = (14)_{10}$. Il s'agit donc d'un message de type MessageSignal.

Question 21

Les trames 6, 7 et 8 émettent les trames contenant les données relatives à la rupture. La couche 3 (IP) s'occupe de fragmenter le paquet de données IP en 3 trames, car la couche 2 (Ethernet) ne peut émettre 1 500 octets par trame (MTU Ethernet : Maximum Transfert Unit).

Taille trame 6 : len = 1448

Taille trame 7 : len = 1448

Taille trame 8 : len = 1122

Taille totale = 4018 = taille d'un message de la question 14.

Question 22

Processeur SIMD : Single Instruction on Multiple Data « instruction unique, données multiples ». La même instruction est appliquée simultanément à plusieurs données pour produire plusieurs résultats. Mettre du parallélisme pour traiter des données de vecteurs ou de matrices. Application CASC : domaine du traitement du signal, donc SIMD adapté.

Question 23

Ce processeur possède un sous-système DSP (Imaging Video and Audio Processor). Le DSP (Digital Signal Processor) contient un jeu d'instructions spécifique pour le traitement du signal adapté à l'application CASC.

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	X		X	
GestionnaireTrames.o		X		X
GestionnaireTrames.h	X		X	
libboost_system.so		X		X
build/Release/bin/Casc		X		

Question 25

C'est la phase d'édition de lien (linkage) permettant de générer l'exécutable à partir des fichiers objets et des bibliothèques.

Question 26

Boucle « while » : le capteur CASC teste la connectivité avec le cloud à travers le réseau IP (commande ping). Si le ping aboutit, l'application CASC ainsi que quelques services sont démarrés. Au bout de 10 tentatives infructueuses, le capteur reboote.

Remarque : une erreur de transcription s'est glissée dans le script shell du DT9 : il manque la commande exit(0) après le lancement de l'application CASC (Casc &) afin d'éviter le problème du redémarrage permanent du capteur.

Question 27

Pas de droit d'exécution (absence de « x »).

Question 28

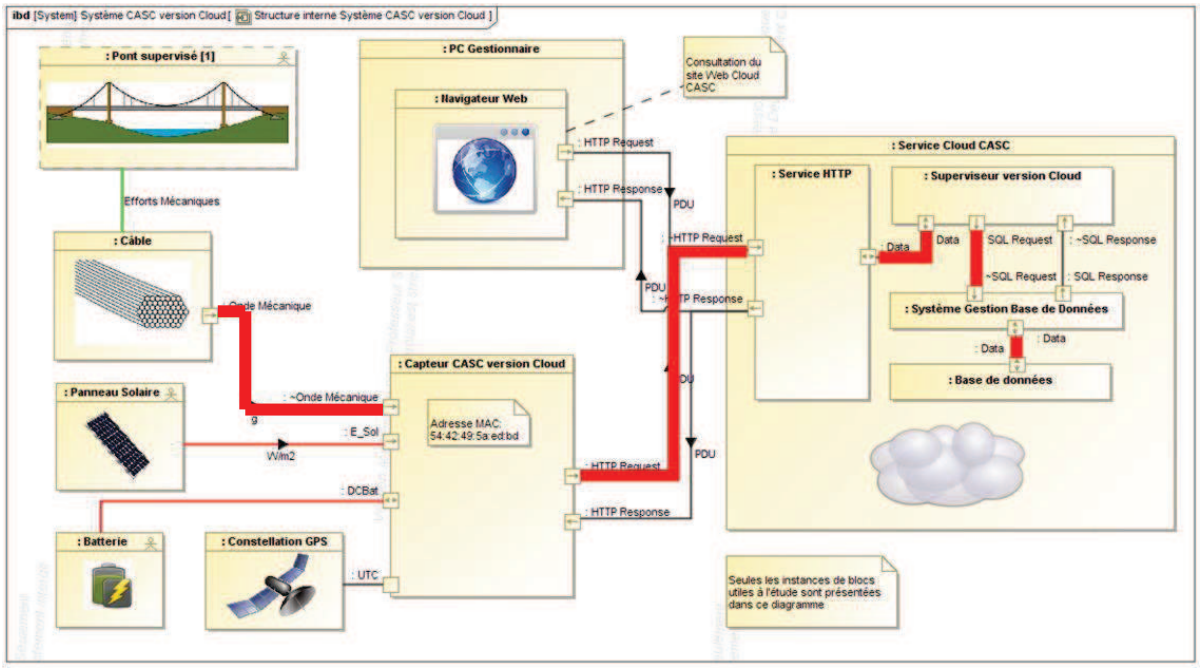
Commande modifiant les droits d'exécution :

chmod ugo+x autorun.sh

ou bien chmod 755 autorun.sh

ou bien chmod u=rwx, g=rx, o=rx autorun.sh

Question 29



Question 30

Champs	Explications
POST	requête http POST (émission de données)
/cloudcascapp/evenement_casc/	adresse URL du service cloud concerné par la demande
HTTP/1.1	protocole et version du protocole http utilisée (1.1)
Host: casc.ifsttar.fr	adresse du destinataire dans le cloud
Content-Length:16241	nombre de données transmises
timestamp=…	données transmises avec la requête

Question 31

Requête POST : émission de données vers un service web.
Requête GET : demande de ressource disponible sur un service web.

Dans le cas étudié, le capteur transmet des données vers le cloud. Le POST s'impose.

Bien qu'il soit possible de transmettre des données avec une requête GET (dans l'URL), la taille de celle-ci est limitée par le serveur à environ 2 000 caractères.

Question 32

&capteur=54:42:49:5a:ed:bd&

Question 33

201 : requête réalisée ; ressource créée (2XX = OK)
404 : ressource indisponible (4XX = erreur)

Question 34

Aucune ambiguïté puisque les deux capteurs sont sur des réseaux locaux différents (adresses IP privées gérées par deux passerelles différentes).

Question 35

NAT (*Network Address Translation*) : la passerelle fait le lien entre la partie privée (réseau local) et la partie publique (Internet). Les adresses IP des passerelles Internet sont publiques et uniques alors que les adresses IP peuvent être réutilisées dans d'autres réseaux locaux.

Question 36

Adresse IP de la passerelle par défaut : 192.168.0.254/24.

Question 37

adresse_mac : clé primaire
ligne_id : clé étrangère

Question 38

```
UPDATE cloudcascapp_capteur
SET seuil_haut = 150 ;
```

Question 39

```
INSERT INTO cloudcascapp_ligne
(nom, commentaires, filtretemporel_actif, flitretemporel, ouvrage_id)
VALUES ('Ligne RD aval', '10 juillet 2016', 1, 5000, 2) ;
```

Question 40

```
SELECT auth_user.email from auth_user
JOIN cloudcacscapp_ouvrage on gestionnaire_id = auth_user.id
WHERE cloudcacscapp_ouvrage.id = 5
```

Question 41

Relation d'agrégation

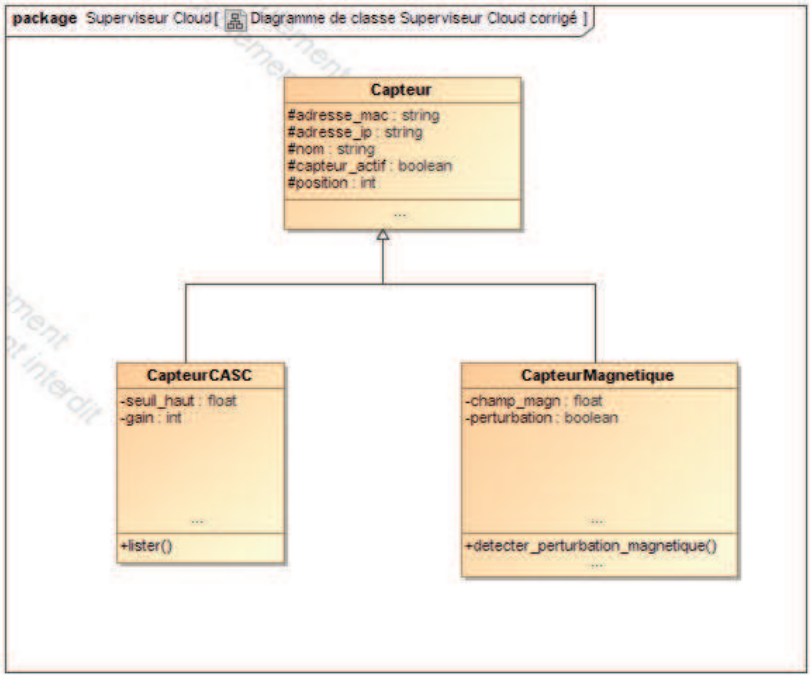
Question 42

$$\begin{cases} X = V_{onde} \times (t_i - t_0) \\ L_{ij} - X = V_{onde} \times (t_j - t_0) \end{cases} \Rightarrow X = \frac{1}{2} \times (L_{ij} - V_{onde} \times (t_j - t_i)) = \frac{1}{2} \times (L_{ij} - V_{onde} \times \Delta t_{ij})$$

Question 43

```
# la variable « liste » est initialisée par le constructeur de la classe rupture à partir de 3 événements
def calculer_position_rupture(self) :
    vonde = (self.liste[1][2]-self.liste[0][2])/(self.liste[1][1]-self.liste[0][1])
    longueur_troncon = self. liste[2][2] – self.liste[1][2]
    dt = self.liste[2][1] – self.liste[1][1]
    localisation = 0.5*(longueur_troncon - vonde * dt)
    return [self.liste[1][0], self.liste[2][0], localisation, vonde]
ou bien
def calculer_position_rupture(self) :
    vonde = calculer_vitesse_onde(self)
        # pour ceux qui ont bien regardé le diagramme de classes
    longueur_troncon =self. liste[2][2] – self.liste[1][2]
    dt = self.liste[2][1] – self.liste[1][1]
    localisation = 0.5*(longueur_troncon - vonde * dt)
    return [self.liste[1][0], self.liste[2][0], localisation, vonde]
```

Question 44



Question 45

Utilisation d'un système de gestion de version (git, subversion, etc.).
Utilisation d'un framework de tests unitaires (Test Driven Development).
Utilisation d'outils d'intégration continue (Jenkins, Wercker, etc.).
D'autres réponses sont aussi acceptées comme la documentation du code, l'utilisation de modélisation UML/SysML, etc.

Question 46

Cycle en V – Enchaînement de plusieurs phases : analyse des besoins et faisabilité, spécification fonctionnelle, conception architecturale, conception détaillée, codage, Test unitaire, test d'intégration, test de validation, test d'acceptation
Inconvénients du cycle en V : lourdeurs. Effet tunnel. Peu itératif. Tests et démonstrations tardifs.
Agile :

- viser la satisfaction client (accepter les changements) ;
- démontrer l'avancement (TODO, DOING, TO VERIFY, DONE) ;
- livrer l'attendu après des itérations courtes de 2 à 3 semaines (plus d'effet tunnel) ;
- itérer sur des fonctionnalités hiérarchisées.