BREVET DE TECHNICIEN SUPÉRIEUR
SYSTÈMES NUMÉRIQUES

Option A – Informatique et Réseaux

Épreuve E4 : ÉTUDE D'UN SYSTÈME NUMÉRIQUE ET D'INFORMATION

SESSION 2020

_____

Durée : 6 heures

Coefficient : 5

_____

L'usage de la calculatrice avec mode examen actif est autorisé.
L'usage de la calculatrice sans mémoire, « type collège » est autorisé.

Tout autre matériel est interdit.

Ce sujet comporte :

Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Chaque candidat remettra deux copies séparées : une copie « domaine professionnel » dans laquelle seront placés les documents réponses pages DR-Pro1 à 8 et une copie « Sciences Physiques » dans laquelle sera placé le document réponses page DR-SP1.

PRÉSENTATION DU SYSTÈME

Supervision de bornes de recharge en agglomération

1. Présentation du système

La transition énergétique dans le domaine du transport passe de plus en plus par l'acquisition d'un véhicule électrique, notamment pour les déplacements urbains.

L'installation de bornes de recharge (Figure 1) en nombre suffisant devient une nécessité pour beaucoup de communes en France.

Les systèmes de recharge sont basés sur un paiement après création d'un compte client. Le client possède alors une carte sans contact RFID.

Pour les clients occasionnels, un système de paiement via un QRCode et un Smartphone est aussi possible. Le QRcode permet d'identifier la borne de recharge et d'accéder à un site de paiement en ligne.

Figure 1 : installation de bornes de recharge de véhicules électriques.

2. Architecture d'une installation d'un système de recharge

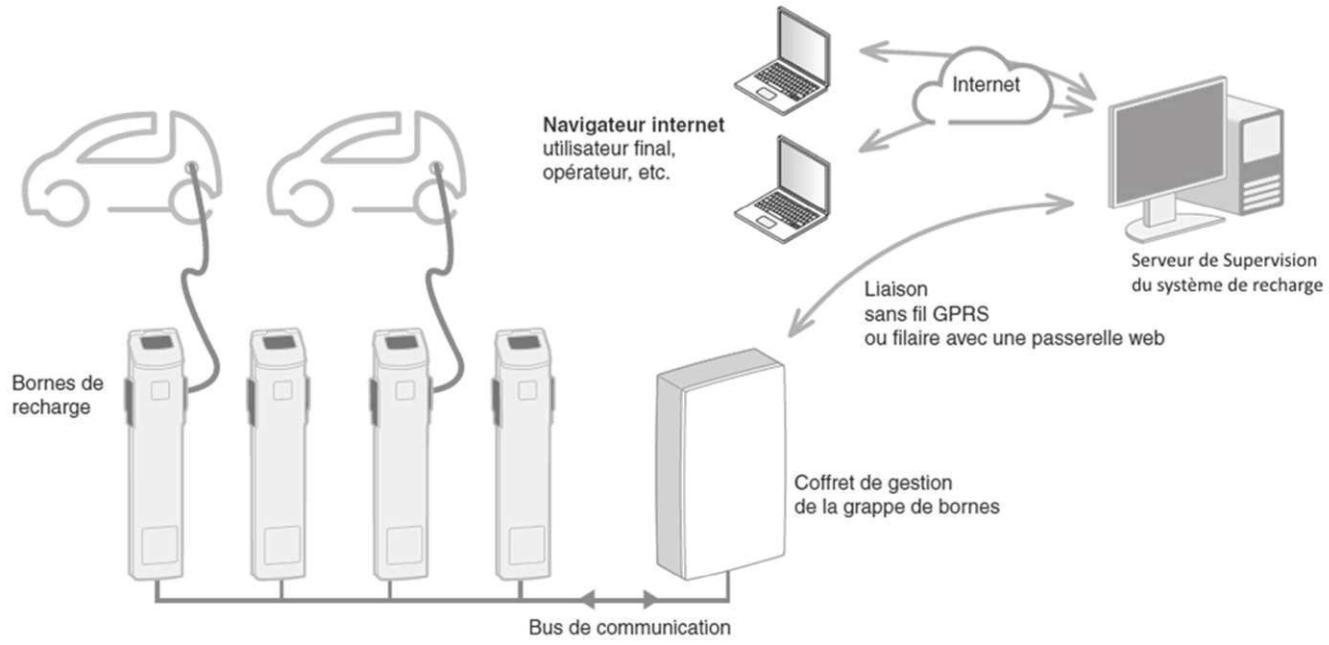


Figure 2 : schéma d'une installation de recharge

Un système de recharge (Figure 2) est composé :

- de **bornes de recharge** qui assurent la charge et une communication entre le véhicule et la borne.
- d'un coffret de gestion de la grappe de bornes qui assure :
  - la communication avec les **bornes de recharge** : elle peut être assurée par le protocole Modbus via une liaison série RS485 ou par le protocole ZigBee via une liaison sans fil,
  - la communication entre les bornes de recharge et le Serveur de Supervision.
- d'un Serveur de Supervision du système de recharge dont le rôle est d'assurer :
  - l'autorisation des recharges,
  - la facturation des clients,
  - la maintenance des bornes de recharges.

Le dialogue entre le coffret de gestion et le serveur de supervision utilise le protocole OCPP (**O**pen **C**harge **P**oint **P**rotocol). La liaison entre ces deux éléments peut-être assurée par différentes technologies (GPRS, Ethernet, ...).

3. Objectifs du sujet

L'étude proposée porte sur la communication entre une borne de recharge, le coffret de gestion et le système de supervision.

Une première partie portera sur le dialogue RFID au sein de la borne. Puis on étudiera la manière d'enregistrer les informations des clients et des bornes au sein de la supervision. Ensuite, on abordera la mise en place de Services Web gérant les autorisations nécessaires à la recharge d'un véhicule. La dernière partie nous amènera à réfléchir à l'infrastructure réseau d'une installation type.

SUJET

Option A Informatique et Réseaux

Partie 1 Domaine Professionnel

Durée 4 h coefficient 3

Partie A. Analyse et spécifications

Problématique : Il s'agit dans cette partie, de préciser les procédures des utilisateurs du système de recharge de véhicules électriques.

Les différents diagrammes SysML et/ou UML sont donnés en annexe ou sont à compléter dans les documents réponses.

Le diagramme de cas d'utilisation ne présente qu'une partie des fonctionnalités du système de recharge. Il se concentre notamment sur un mode particulier : la recharge non autonome.

Le diagramme de packages (Figure 3) présente le système dans son ensemble.

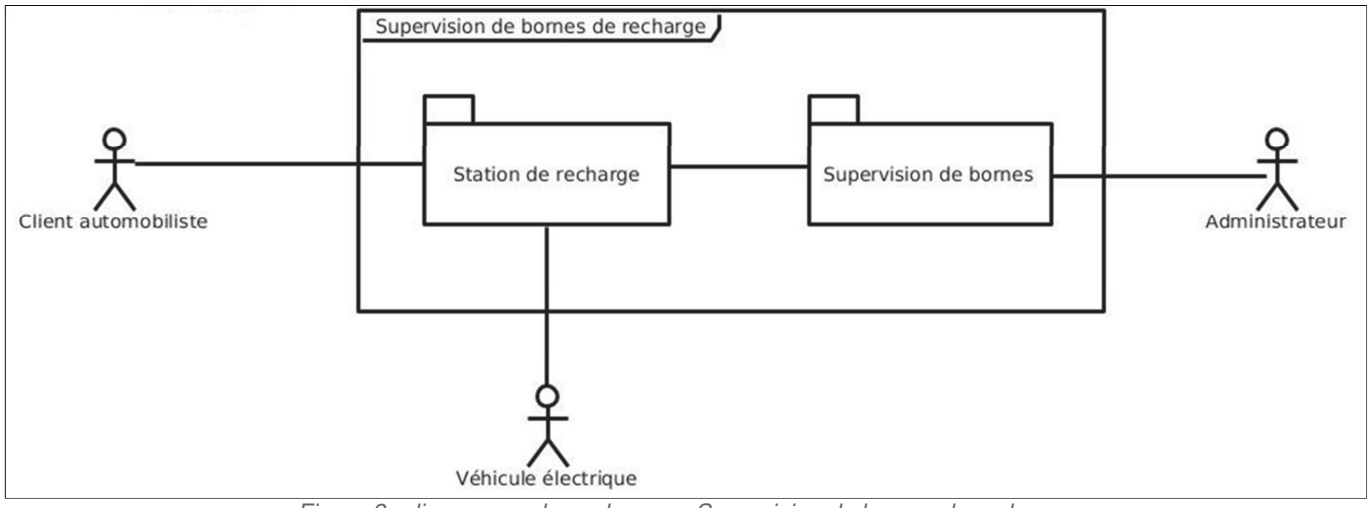


Figure 3 : diagramme de packages – Supervision de bornes de recharge

L'étude porte principalement sur le package « Station de recharge » détaillé dans le diagramme de cas d'utilisation (Figure 4).

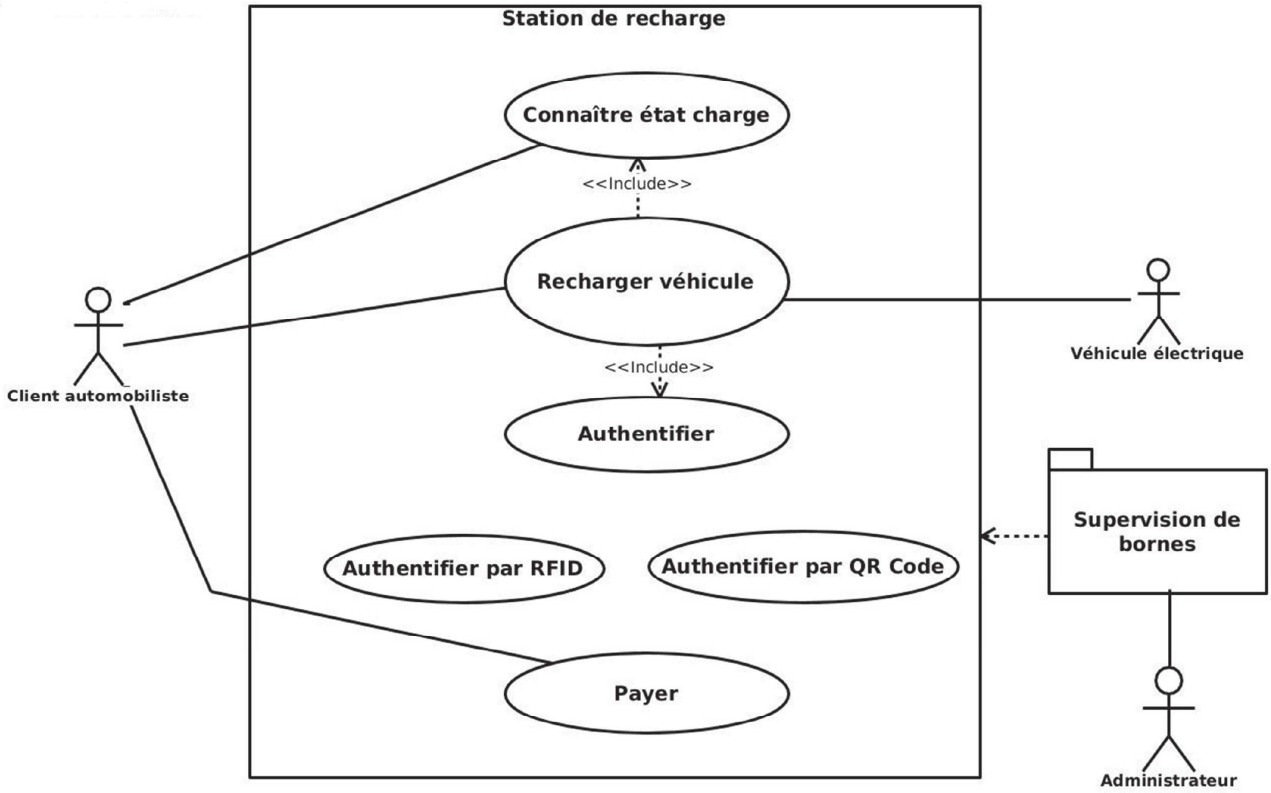


Figure 4 : diagramme de cas d'utilisation – Station de recharge

Q1. En vous aidant du diagramme des exigences *(**documentation PP1**),* citer les deux possibilités qui permettent au client d'interrompre une charge en cours.

Q2. En vous aidant du diagramme des exigences *(**documentation PP1**),* expliquer comment le client est informé de la fin de la charge de son véhicule.

Description du cas d'utilisation « Recharger véhicule » :

Le client automobiliste a la possibilité de recharger son véhicule électrique. Pour cela, il peut s'authentifier grâce à une carte RFID préalablement enregistrée auprès des services de la commune. S'il n'a pas de carte RFID, il peut malgré tout accéder à la recharge grâce au QR Code présent sur la borne. Une fois le QR Code scanné grâce à son smartphone, il a accès au site Web de la commune où il peut créer un compte ou simplement prépayer par carte bancaire.

Q3. En vous aidant du diagramme de cas d'utilisation (Figure 4), lister les acteurs qui interagissent avec le système.

Q4. Compléter le diagramme de cas d'utilisation (**document réponses**) en précisant les relations entre les cas « Authentifier », « Authentifier par RFID » et « Authentifier par QR Code ».

Le diagramme de séquences système de la figure 3 présente le fonctionnement global du système du point de vue du client automobiliste. Il se focalise sur la recharge du véhicule suite à une authentification via un badge RFID.

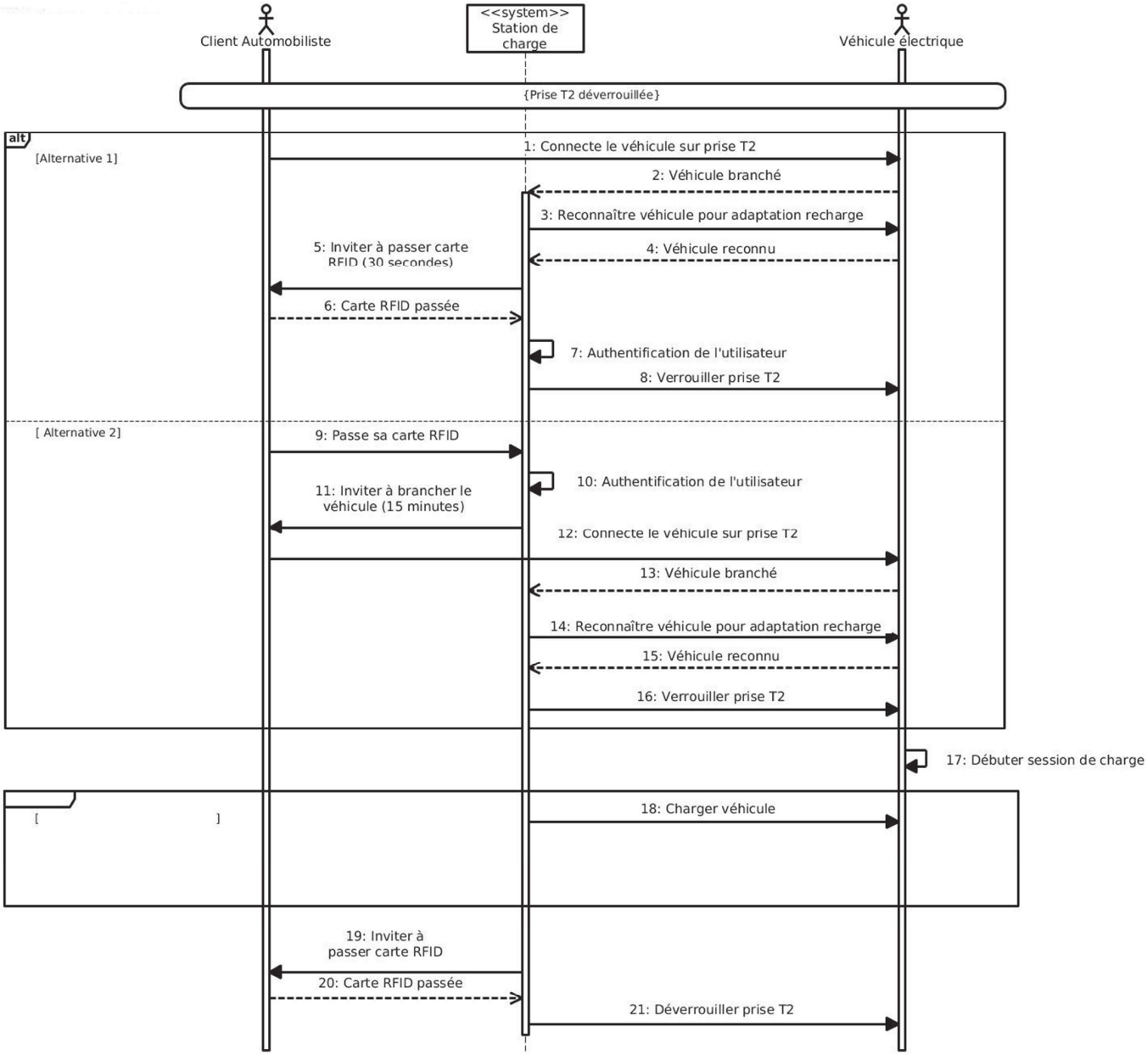


Figure 5 : diagramme de séquences système

Q5. Identifier la différence majeure entre les deux alternatives [Alternative 1] et [Alternative 2] du diagramme de séquences (Figure 5)

Dans ce scénario, tant que le véhicule n'est pas chargé totalement, la charge continue. Le véhicule informe régulièrement la station de son niveau de charge.

Q6. Compléter le second fragment du diagramme de séquences système du **document réponses** (zones 1, 2 et 3) de façon à tenir compte de la fin de charge.

Partie B. Architecture

Problématique : Il s'agit dans cette partie d'effectuer des choix technologiques de façon à répondre aux besoins des collectivités.

Dans tout ce qui suit, la borne de recharge utilisée respecte les différentes normes de communication ModBus, ISO/IEC14443, OCPP1.5 ou OCPP1.6.

Borne EVLink City

Figure 6 : une borne EVLink

La série des bornes Schneider EVLink (Figure 6) dispose de toutes les options nécessaires pour en faire un ensemble de recharge de véhicules électriques en grappe et reliée à une supervision.

*Elles permettent notamment la lecture de cartes RFID par l'**ajout** d'un module : ce dernier communique avec l'unité centrale de la borne grâce au protocole **ModBus RTU** (Remote Terminal Unit)*

Ce module est capable de lire des cartes RFID dont la fréquence porteuse est 13,56MHz.

Q7. Parmi tous les modules RFID présentés dans la **documentation PP2**, proposer ceux qui correspondent le mieux aux exigences du fabricant de la borne. Justifier votre réponse.

*En vous appuyant sur l'extrait des spécifications « ModBus over Serial Link » (**documentation PP4**),*

Q8. Déterminer le nombre maximum de bornes de recharge, qu'il est possible d'installer, par secteur avec le protocole ModBus

Certains lecteurs RFID ne disposent pas de liaison série exploitant le protocole ModBus (exemple : BALLUFF BIS M-620-068-A01-00-ST29, interface RS232) et ne peuvent donc pas être connectés directement à la borne de recharge.

*Leur utilisation nécessite l'ajout d'une passerelle ModBus, comme le composant **AnyBus Communicator** présenté dans la **documentation PP3**.*

Q9. Donner le nombre d'esclaves ModBus que peut adresser cette passerelle.

On souhaite utiliser cette passerelle pour ajouter un lecteur RFID. Voici la configuration de la passerelle pour la relier à une borne de recharge du système :

- *RS232,*
- 9600 bits/sec,
- pas de parité,
- 2 bits de stop,
- adresse ModBus/RTU : 15.

Q10. Préciser sur le **document réponses**, la position des switches de façon à ce que la passerelle décrite dans la **documentation PP3** puisse s'interfacer correctement avec le système.

Partie C. Conception et réalisation **–** C++

Problématique : Il s'agit dans cette partie de faire des choix d'implémentation logicielle.

La lecture d'une carte ou d'un badge RFID permet de récupérer un numéro unique UID souvent appelé Tag UID. Les cartes RFID peuvent être de différents types. Ici, les cartes RFID sont des badges ***Mifare 13,56 Mhz Classic****,* ***Ultralight****,* ***Plus*** *ou* ***DESFire****.*

Q11. À partir du diagramme de classes de la **documentation PP5**, compléter la déclaration en C++ de la classe CTag du **document réponses**.

Lors du passage d'un ***badge RFID*** *devant le* ***lecteur****, une succession d'échanges a lieu entre le* ***badge RFID*** *et le* ***lecteur****.*

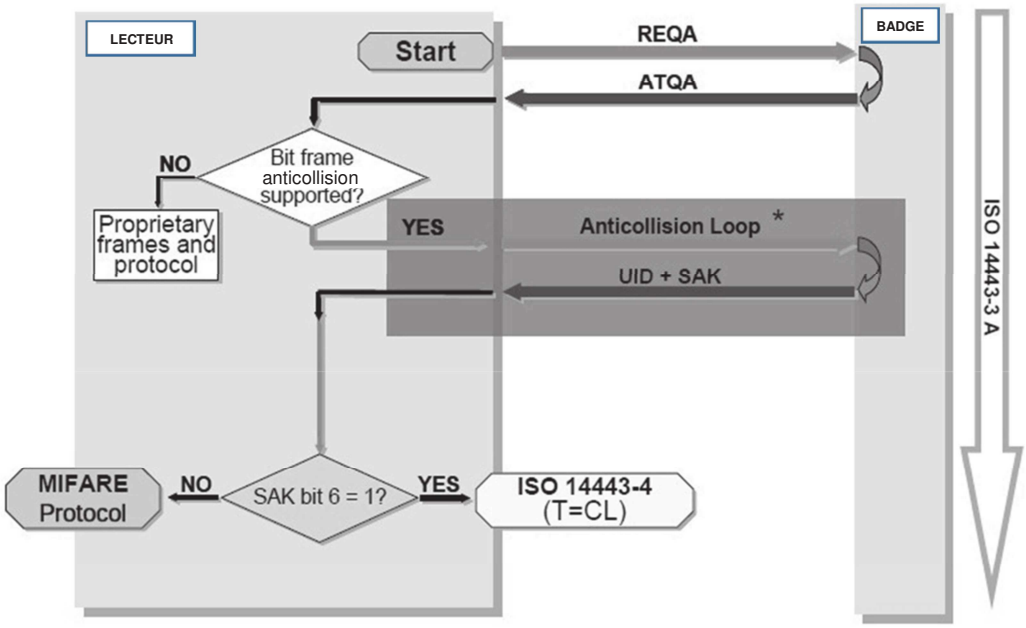


Figure 7 : dialogue entre le ***lecteur*** *et le* ***badge*** *tag RFID*

Le lecteur commence par émettre une requête REQA. Le badge RFID répond par ATQA. Puis un ensemble d'échanges a lieu permettant la récupération de l'UID (Unique IDentifier) et du SAK (Select AcKnowledge).

Remarque : on s'aperçoit que c'est le lecteur qui est à l'initiative du dialogue. Le lecteur détecte la présence d'un badge RFID car celui-ci perturbe son champ électromagnétique.

En fait, l'UID ne sert qu'à la détection (présence ou absence) d'une collision, c'est-à-dire, dans le cas où deux badges RFID (ou plus) se situent dans le champ électromagnétique du lecteur.

L'UID est une information codée sur 4 octets ou 7 octets.

Q12. Expliquer comment le lecteur peut connaître la taille de l'UID du badge RFID à partir de la **documentation PP6** (ATQA Coding of NXP Contactless Card ICs).

La valeur ATQA lue par le lecteur est un mot de 16 bits stocké dans l'attribut atqa de la classe CReader.

Q13. Sur le **document réponses**, compléter le code C++ de la méthode *determineSizeUID()* de la classe CReader, permettant d'initialiser l'attribut *sizeUID* (taille de l'UID) à partir de la valeur contenue dans l'attribut *atqa*.

Lors des échanges, certaines trames sont vérifiées à l'aide d'un calcul de BCC (Bit Count Check) codé sur un octet. Le calcul de l'octet BCC est un OU exclusif entre tous les octets de la trame.

Rappel : le OU exclusif en C/C++ est l'opérateur ^.

Lorsqu'un échange entre le lecteur et le badge a eu lieu, les attributs atqa et sak de la classe CReader se voient affecter les valeurs de l'ATQA et du SAK transmis. L'attribut tag de la classe CTag se verra affecter la valeur de l'UID du badge RFID. Le BCC est stocké dans l'attribut BCC de la classe CReader.

Q14. En vous aidant de la **documentation PP14**, compléter sur le **document réponses** le code C++ de la méthode *calculateBCC()* de la classe CReader permettant de calculer le BCC de la valeur de l'UID du badge RFID.

On décide d'ajouter les méthodes accesseur et mutateur dans la classe CTag afin d'accéder à l'attribut tag de celle-ci.

Q15. Sur le **document réponses**, compléter l'implémentation en C++ de l'*accesseur* et du *mutateur* de la classe CTag en vous aidant du diagramme de classes de la **documentation PP5**.

Partie D. Supervision de système de recharge - Base de données SQL

On s'intéresse à présent à la supervision. Elle dispose d'une base de données SQL dont la structure est représentée par le schéma Entités-Relations du ***document PP15****.*

La ***documentation PP7*** *rappelle les principales requêtes SQL.*

La base de données comporte les tables suivantes :

- user : détient les informations de l'utilisateur
- address : détient les détails de l'adresse d'un utilisateur
- charge_box : détient les informations d'une borne (modèle, numéro de série, positionnement terrestre, etc.)
- ocpp_tag : détient les informations d'un badge
- reservation : détient les informations de réservation d'une borne

Le système est dimensionné pour :

- autoriser un utilisateur à posséder plusieurs badges,
- accepter que plusieurs utilisateurs habitent à une même adresse (cas d'une famille).

Q16. Compléter le schéma Entités-Relations du **document réponses** en faisant apparaître les relations entre les tables « address », « user » et « ocpp-tag », et en respectant le symbolisme des relations (flèches en pointillés).

Q17. Élaborer la commande SQL permettant de lister les identifiants de l'ensemble des bornes utilisées (table « charge_box »).

Le logiciel de supervision propose entre autres, d'ajouter de nouveaux clients ou de modifier leurs informations. Toutes les informations concernant le client peuvent être saisies dans le logiciel de supervision.

Un client existant déclare avoir perdu son badge. Voici ses informations actuellement détenues dans la base de données :

- *Nom : Proust*
- *Prénom : Marcel*
- *Date de naissance : 10 juillet 1971*
- *Sexe : Masculin*
- *Tag : 70880485138334*
- *Rue : avenue Charles de Gaulle*
- *N°: 4*
- *Code Postal : 98100*
- *Ville : Balbec*

Un nouveau badge lui est attribué ayant le tag 70880485139000 afin de remplacer le badge perdu dont le paramètre ocpp_tag_pk vaut 71.

Q18. Élaborer la commande SQL permettant de mettre à jour ces informations.

Partie E. Communication Bornes - Supervision

*Le protocole OCPP dans sa version 1.5 met en œuvre un ensemble de requêtes et de réponses (25 en tout) entre les bornes de recharge et la supervision (**assurée par le Serveur OCPP**). Les requêtes et les réponses sont encapsulées dans des services web selon les protocoles SOAP et WSDL.*

SOAP permet donc d'installer des services au sein d'un serveur Web. Il s'agit en fait d'un échange de fichiers XML.

Q19. À partir de l'extrait de la **documentation PP8** sur SOAP, compléter le **document réponses** en faisant apparaître l'empilement des protocoles réseau IP, TCP, HTTP et SOAP.

*Plusieurs captures d'échanges entre une borne et la supervision à l'aide d'un analyseur réseau ont été effectuées : **documentation PP9**.*

Une tentative de réservation de borne a été effectuée par un client.

Q20. En analysant les captures de trame de la **documentation PP9,** donner l'identifiant de la borne ayant tenté une demande d'autorisation.

Q21. En analysant les trames des dialogues 1 et 2 de la **documentation PP9,** préciser les numéros des trames qui débutent les requêtes et les réponses SOAP.

Q22. En vous aidant de la **documentation PP10,** indiquer les possibilités de réponse de la supervision vers la borne lors d'une demande d'autorisation.

*Lors du dialogue 1 de la **documentation PP9**, le client effectue une tentative d'authentification à l'aide de sa carte RFID. On constate alors une erreur.*

Q23. Identifier l'UID du tag à l'origine de l'erreur.

Q24. Identifier le groupe de balises SOAP permettant d'informer la présence de cette erreur.

*Lors du dialogue 2 de la **documentation PP9**, la supervision a validé la demande du client. Pour des raisons de sécurité, cette acceptation est limitée dans le temps.*

Q25. Indiquer la balise SOAP qui met en évidence cette limitation.

Afin de pouvoir effectuer les requêtes et réponses conformes au protocole SOAP, il est nécessaire de créer une infrastructure (un schéma) décrite dans un document au format WSDL (description des services Web dans un fichier XML)

*Un extrait de l'infrastructure des services Web mis en place par la supervision OCPP se trouve en **documentation PP10**.*

Q26. Indiquer la taille maximale d'un tag UID (ou idToken) que peut accepter la supervision. Justifier votre réponse.

Partie F. Réseau

La suite de l'étude, porte sur l'architecture réseau des bornes de recharge installées au sein d'une ville. Un extrait de l'architecture est présenté dans la ***documentation PP11****.*

Plusieurs services interviennent dans la gestion des recharges :

- le service administratif qui s'occupe de la gestion des comptes client, de la facturation et de la gestion des contentieux,
- le service technique qui s'occupe de la gestion des bornes de recharge (maintenance), de l'administration de la supervision OCPP ainsi que de l'installation des nouvelles bornes.

Les trois routeurs font partie d'un seul et même réseau d'adresse 172.16.150.0/25

*Les bornes sont installées au sein d'un secteur. Les bornes d'un secteur reliées par une liaison Ethernet ont une adresse de réseau en 172.16.****N°secteur*** *.0/25*

Au sein d'un secteur, on peut avoir plusieurs points de recharge. À chaque point de recharge est associée une borne maître (au sens ModBus).

Certains secteurs sont à part : il s'agit des bornes installées dans une zone où Internet n'est accessible qu'à partir d'une connexion 3G. La liaison est alors assurée grâce à un modem 3G ajouté dans la borne maître.

Les machines (ordinateurs) du service administratif font partie du réseau 172.20.0.0/25. Un serveur DHCP s'occupe de fournir les adresses IP à ces machines.

Les machines (Serveur OCPP, Serveur MySQL, ordinateurs) du service technique font partie du réseau 172.16.96.0/25. Certaines machines de ce réseau ont des adresses IP fixes (serveur OCPP, serveur MySQL).

Q27. Indiquer le mode d'accès de la borne maître du point de recharge 1 du secteur 2 au accéder au Serveur OCPP ?

Q28. Compléter le **document réponses** en choisissant les adresses IP des différentes interfaces des routeurs.

Q29. Compléter le **document réponses** pour le réseau 172.16.96.0/25

Les tables de routage des trois routeurs contiennent les adresses de destination des réseaux et la correspondance avec l'interface Ethernet (ou WAN) concernée. Le routeur R2 étant un routeur avec une connexion Internet, il sera considéré comme étant la passerelle par défaut.

Q30. Compléter la table de routage du routeur R3 du **document réponses**.

Le serveur OCPP est basé sur un système d'exploitation Linux. On peut sécuriser l'accès à ce serveur grâce au pare-feu **iptables** présent sur celui-ci.

Q31. À partir de la **documentation PP12**, préciser les commandes linux à utiliser pour configurer le pare-feu de façon à ce qu'il autorise les accès (en entrée et en sortie) sur les ports 8080 et ssh.

On désire autoriser les « ping » sur la machine de supervision OCPP. On rappelle que la commande ping utilise le protocole icmp.

Q32. À partir de la **documentation PP12**, préciser les commandes Linux à utiliser pour configurer le pare-feu ***iptables*** de façon à autoriser une machine du réseau 172.16.96.0/25 à « pinger » le Serveur OCPP. Justifier votre réponse.

Un switch (Switch A) permet de relier les trois routeurs R1, R2 et R3.

*Les concepteurs de l'infrastructure décident de remplacer les routeurs R1, R2 et R3 d'ancienne génération par des routeurs CISCO de la série 890 (voir **documentation PP13**).*

Q33. Le switch A est-il toujours nécessaire si on installe les routeurs CISCO 890 ? Justifier votre réponse.

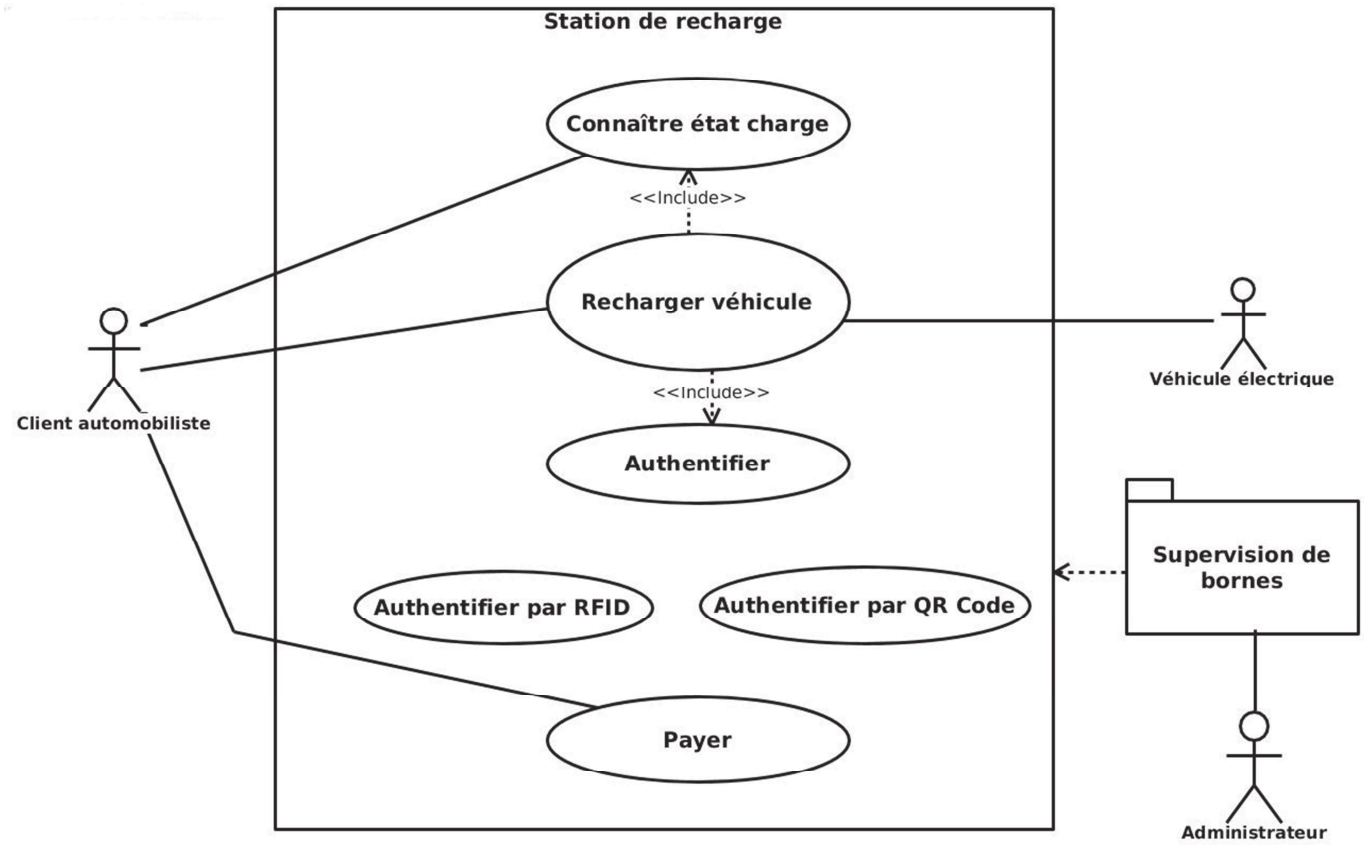
L'organisation logique de l'infrastructure réseau nécessite une évolution en créant des VLANs : meilleure gestion des réseaux, optimisation du flux, sécurisation...

Q34. Combien de VLANs pourrait-on gérer avec les routeurs CISCO de la série 890 ?

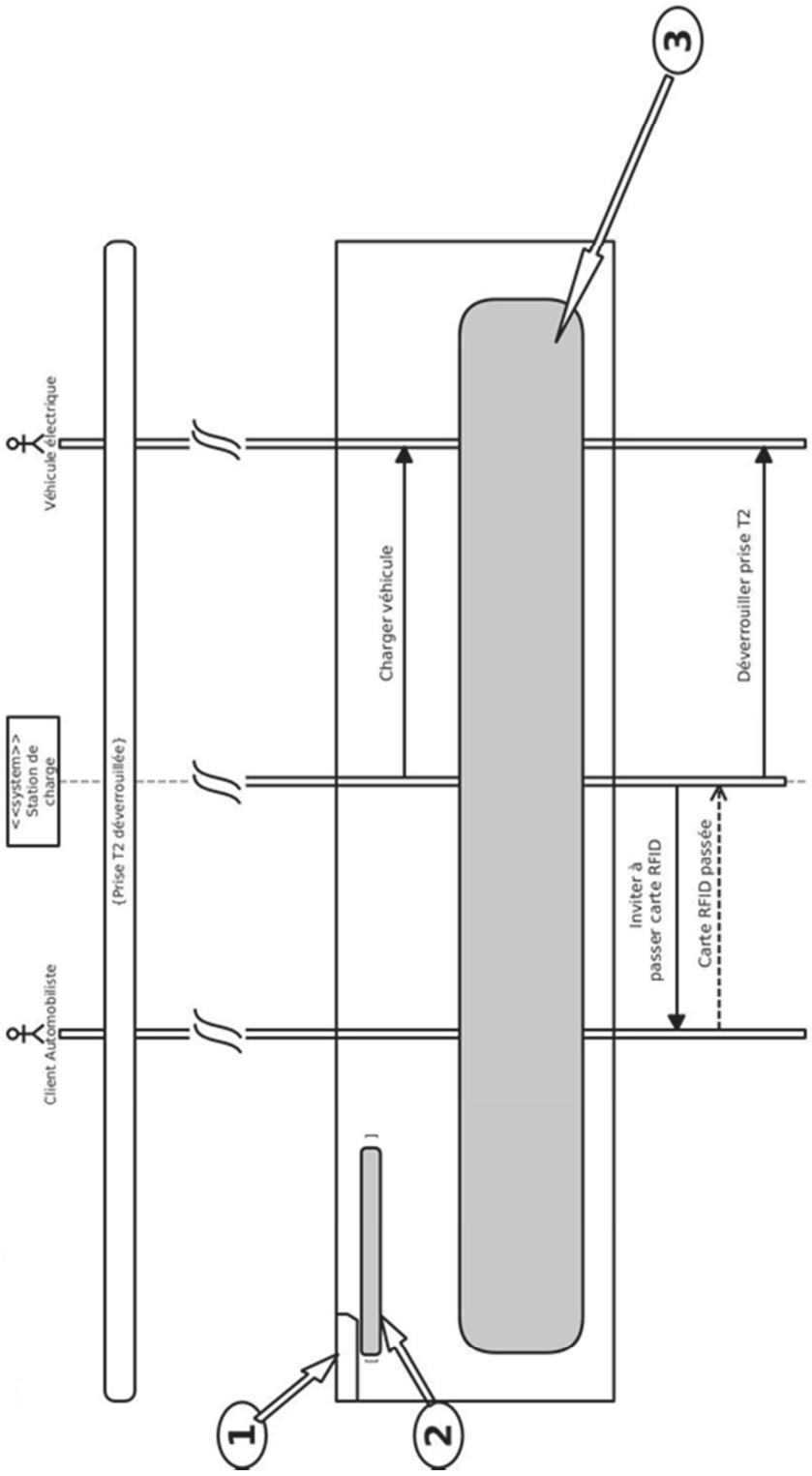
DOCUMENTS RÉPONSES – Domaine Professionnel

A RENDRE AVEC LA COPIE

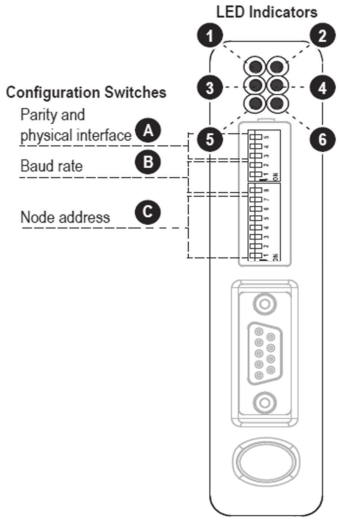
Réponse à la question Q4



Réponse à la question Q6



Réponse à la question Q10



A	Switch 3	Switch 4	Switch 5

B	Switch 8	Switch 1	Switch 2

C	Switch 1	Switch 2	Switch 3	Switch 4	Switch 5	Switch 6	Switch 7
	OFF						

Réponse à la question Q11

```
class CTag
{

  private :



  public:




};
```

Réponse à la question Q13

Code C++ de la méthode *determineSizeUID()* de CReader :

```
#include "creader.h"

void CReader::determineSizeUID()
{









}
```

Réponse à la question Q14

Code C++ de la méthode *calculateBCC()* de CReader :

```
void CReader::calculateBCC()
{

    // Récupération de l'attribut tag de l'instance de la classe CTag
    std::vector <uint8_t> valeurUID =                              ;

    // Calcul du BCC









}
```

Réponse à la question Q15

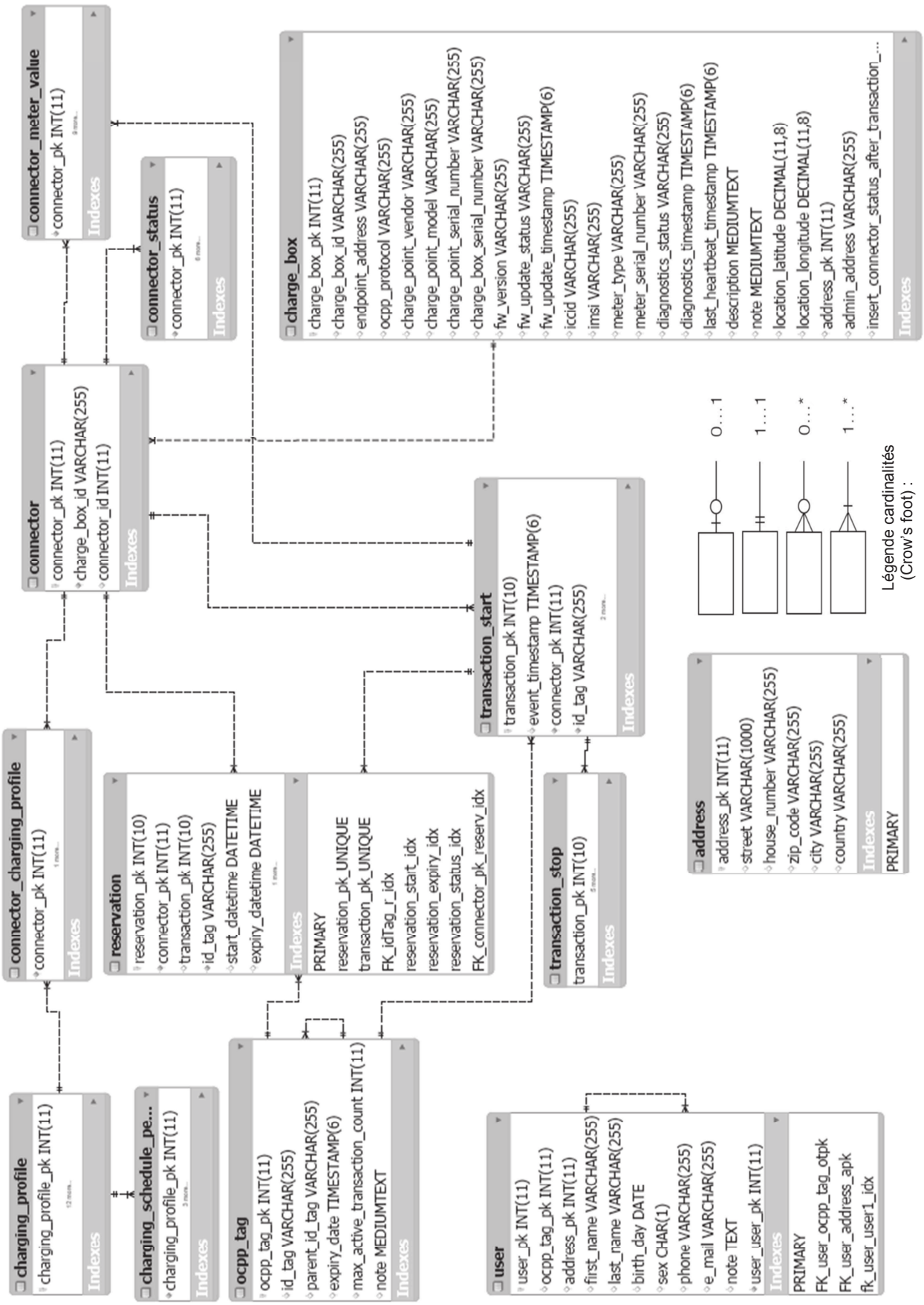
Code C++ de l'accesseur de CTag :

```
#include "ctag.h"
```

Code C++ du mutateur de CTag :

```
#include "ctag.h"
```

Réponse à la question Q16



Réponse à la question Q19

Placer dans le tableau suivant, les protocoles : TCP, HTTP, IP et SOAP

COUCHE	PROTOCOLE
Application	
Transport	
Réseau	

Réponse à la question Q28

Routeur	Eth1	Eth2	Eth3
R1			
R2			
R3			

Réponse à la question Q29

Adresse de réseau	***Masque de sous-réseau en décimal pointé***	***Adresse de broadcast***	***Nombre total de machines***	***Adresse de la première machine***	***Adresse de la dernière machine***
172.16.96.0/25					

Réponse à la question Q30

Table de routage du routeur R3 :

Nom du réseau	*Réseau de destination*	*Masque de sous-réseau en décimal pointé*	*Passerelle*	*Interface*
« backbone »				
Secteur 1	**176.16.1.0**		**@ipEth1R1**	**Eth1**
Secteur 3				
Service administratif				
Service technique				
Route par défaut				

@ipEth1R1 signifie l'adresse IP de l'interface Eth1 du routeur R1.

SUJET

Option A Informatique et Réseaux

Partie 2 Sciences physiques

Durée 2 h coefficient 2

Ce sujet est composé de deux parties indépendantes.

Partie A : communication entre les bornes de recharge de véhicules électriques

A.1 Liaison série asynchrone RS485.

A.2 Protocole Zigbee.

Partie B : contrôle de la tension délivrée par une borne de recharge.

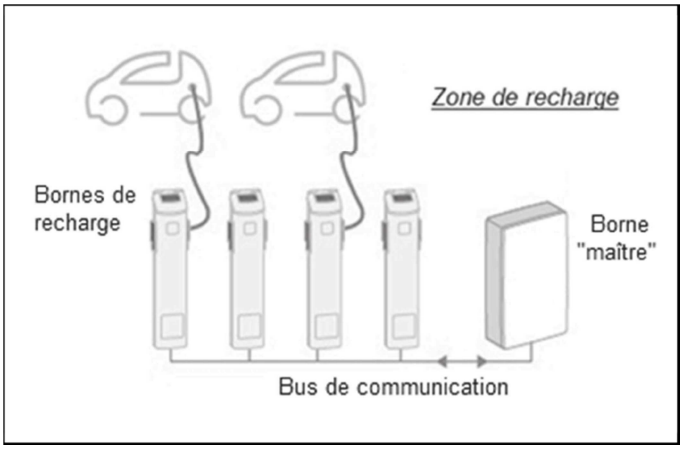
B.1 Numérisation.

B.2 Traitement numérique du signal.

Partie A. Communication entre les bornes de recharge de véhicules électriques et la borne maître

Chaque zone de recharge est constituée d'un ensemble de bornes dont l'une d'elles est une borne « maître ».

Le protocole de communication entre les bornes est soit ModBus sur RS485 soit une communication sans fil ZigBee.



A.1 – Liaison série asynchrone RS485

Les « liaisons séries » sont des moyens de transport d'informations entre divers systèmes numériques.

Elles sont appelées asynchrones lorsqu'il n'y a pas de signal d'horloge transporté avec le signal de données.

Les liaisons séries asynchrones sont rencontrées sous différentes normes dans tous les domaines du traitement de l'information : RS-232/422/485

Problématique : justification de l'emploi de la norme RS-485.

Spécifications	**RS-232**	**RS-422**	**RS-485**
Mode de fonctionnement	Non différentiel	Différentiel	Différentiel
Nombre d'émetteurs/récepteurs sur une ligne	1 émetteur 1 récepteur	1 émetteur 10 récepteurs	32 émetteurs* 32 récepteurs
Longueur maximale du câble	15 m	1 200 m	1 200 m

Tableau 1: spécifications RS-232, RS-422, RS-485

*un seul émetteur est actif à un moment donné source : http://ni.com/white-paper/11390/fr/

Q35. Justifier que la norme RS-232 ne peut pas être utilisée pour la communication entre la borne maître et les bornes de recharge, à partir des spécifications du tableau 1.

Un des principaux problèmes des liaisons séries est la sensibilité aux bruits sur les lignes de transmission du signal. L'émetteur et le récepteur comparent les tensions par rapport à une masse commune en ligne (exemple RS-232). La présence de parasites peut alors limiter la distance maximale et la vitesse de communication.

Avec la norme RS-485, il n'y a pas de masse commune comme signal de référence. La transmission est dite différentielle et nécessite l'utilisation de paires torsadées. Le récepteur fournit une tension V_{RS485} égale à une différence de potentiels telle que $V_{RS485} = Data^+ - Data^-$.

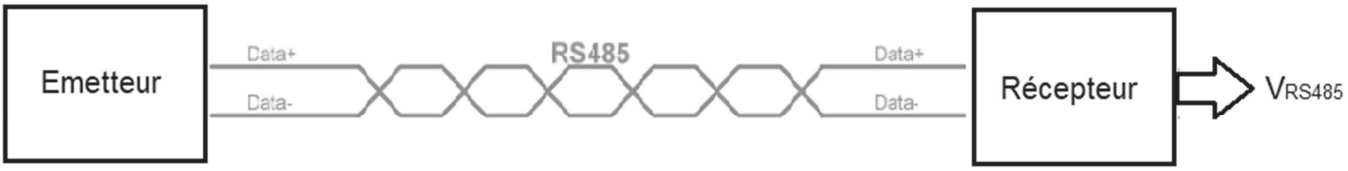


Figure 1: schéma de liaison norme RS-485

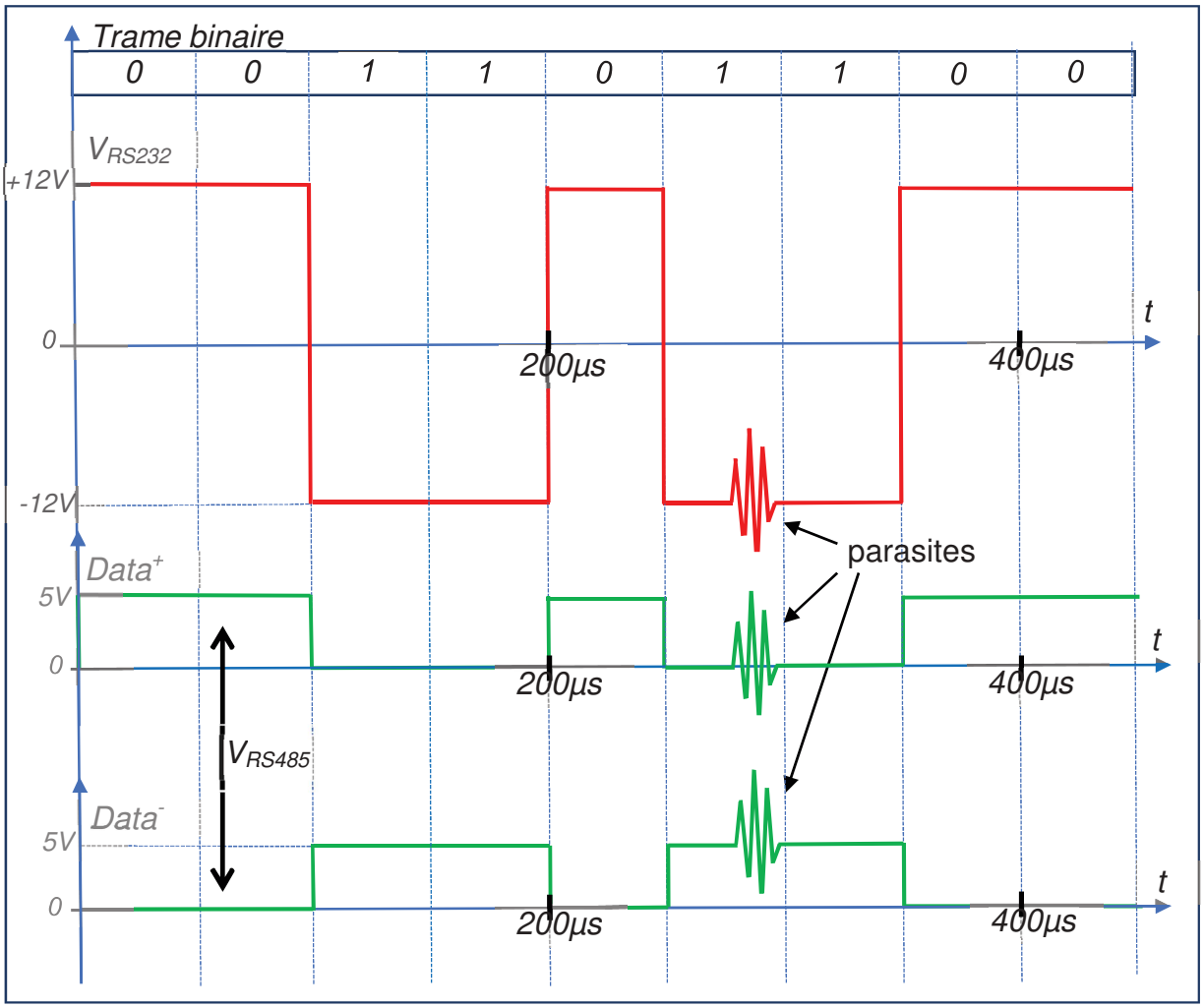


Figure 2 : exemple de chronogrammes de signaux pour une liaison de type RS-232 et RS-485

Q36. Tracer l'allure du signal V_{RS485} sur le document réponses en page DR-SP1.

Q37. Justifier, dans le cas d'une liaison série, qu'une transmission différentielle permet d'obtenir une plus faible sensibilité (meilleure immunité) aux parasites par rapport à une transmission non différentielle.

Q38. Déterminer la valeur du débit binaire en bps (bits par seconde) dans l'exemple proposé à la figure 2.

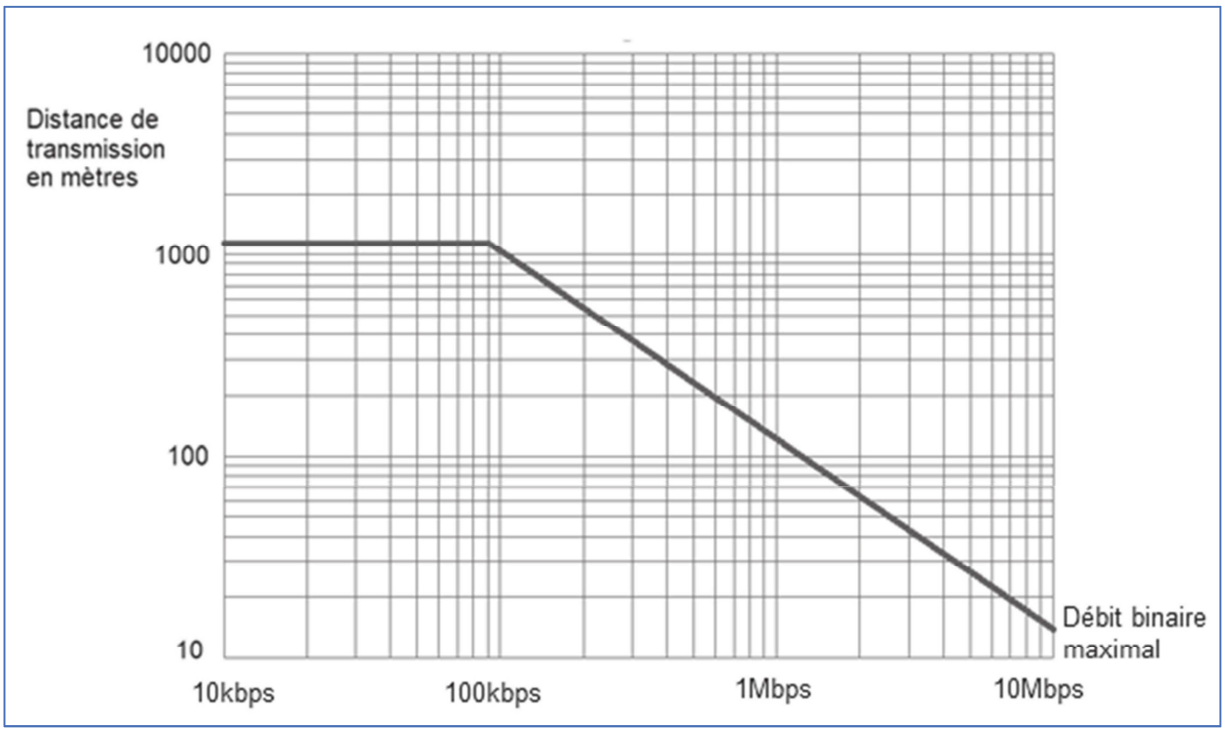


Figure 3: distance de transmission en fonction du débit maximal pour une liaison RS-485

Q39. Justifier, en s'aidant de la figure 3, que l'utilisation de la liaison RS-485 est possible sachant que la distance de transmission ne dépasse pas 100 m et que le débit binaire est de 19 200 bps.

A.2 – Protocole Zigbee

ZigBee est un protocole de réseau local (LAN) fonctionnant à 2,4 GHz. Il a été développé en tant que spécification basée sur l'IEEE 802.15.4 pour une suite de protocoles de communication de haut niveau utilisés pour créer des réseaux personnels sans fil à faible puissance.

ZigBee a été pensé pour des réseaux maillés où l'information se propage de proche en proche, par sauts successifs, jusqu'au destinataire.

Les caractéristiques du système étudié utilisant la technologie Zigbee pour la communication entre les bornes de recharge sont les suivantes :

Bande de fréquence : Bande ISM à 2,4 GHz

Modulation QPSK

Durée minimale d'un bit : T_{Bmin}= 4,0 µs

Bande passante : B = 500 kHz

Puissance électrique de l'émetteur : 0,0 dBm

Gain antenne émission / réception : 0,0 dB

Pertes à l'émission : 3,0 dB

Pertes à la réception : 3,0 dB

Sensibilité de réception : - 86 dBm

Température de fonctionnement : - 40 à 80°C

TEB (Taux d'erreur par bit) ou BER maximal : $1{,}0 \cdot 10^{-4}$

Problématique : vérifier que le protocole Zigbee est une technologie radio robuste, c'est-à-dire qu'il possède une bonne immunité aux bruits.

Un bilan de liaison permet de calculer la puissance disponible au niveau du récepteur en fonction de la puissance fournie à l'antenne d'émission.

Dans le cas de deux antennes séparées par une distance, notée d, en espace libre, l'atténuation A liée à la transmission peut être calculée par la relation suivante :

$$A = 32{,}4 + 20 \cdot \log(f) + 20 \cdot \log(d) \qquad \text{avec A en dB, f en MHz et d en km}$$

Q40. Calculer l'atténuation A en dB pour une distance d de 50 m.

Q41. Calculer le niveau de puissance P_r reçue par le récepteur en dBm pour une distance de 50 m, en tenant compte des pertes à l'émission et à la réception et de l'atténuation A liée à la transmission en espace libre. En déduire si la communication est possible sur cette distance.

Q42. Calculer la distance théorique maximale d_{max} pour une transmission en espace libre.

Q43. Calculer le nombre maximal de bits erronés par seconde pour un débit de 200 kbps à partir des caractéristiques fournies du protocole Zigbee.

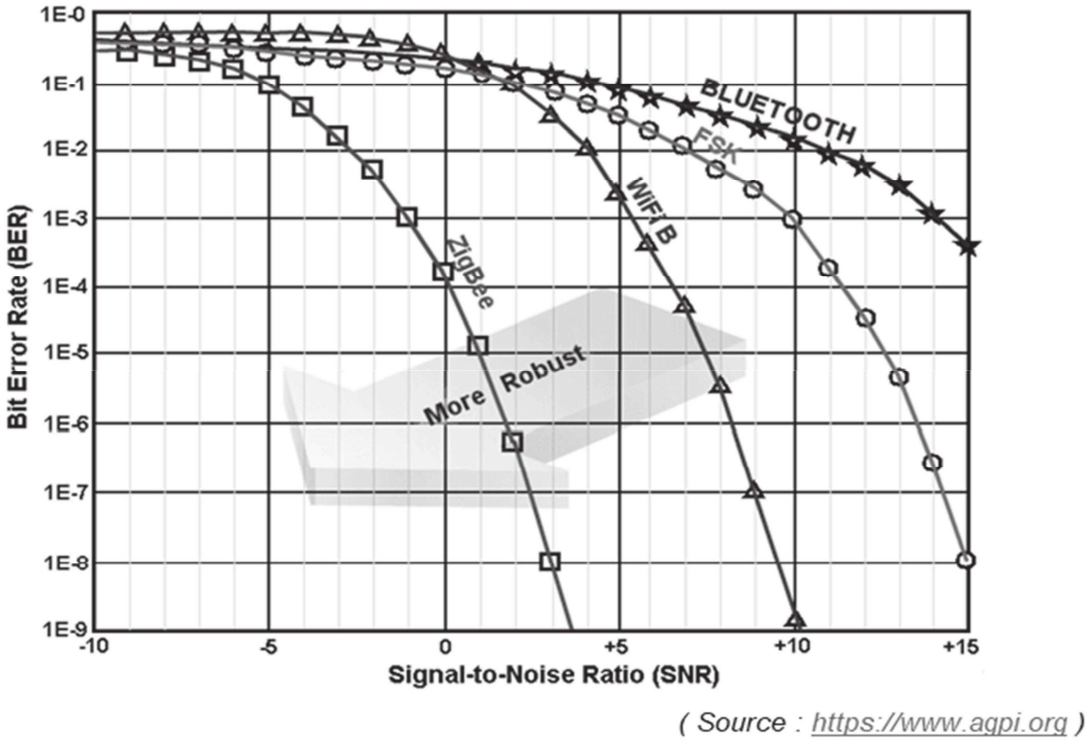


Figure 4 : variations du TEB (BER) en fonction du SNR minimal en dB

Rappel : SNR = S - B
avec : S puissance du signal en dBm ; B puissance du bruit en dBm et SNR le rapport signal sur bruit en dB.

Q44. Déterminer, en s'aidant de la figure 4, la puissance maximale du bruit pour une puissance du signal de - 84 dBm pour que la communication puisse être établie avec le protocole Zigbee.

Q45. Justifier que la communication Zigbee soit qualifiée de « robuste » par rapport au Wifi et au Bluetooth.

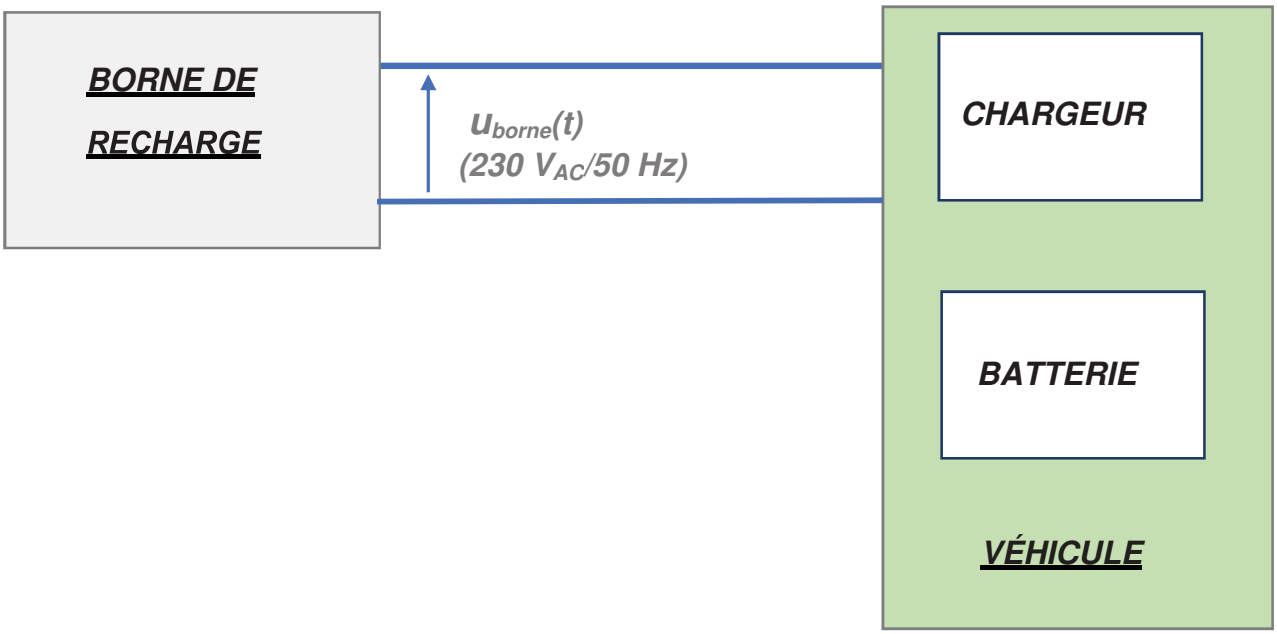
Partie B. Contrôle de la tension délivrée par une borne de recharge.

B.1- Numérisation

En cas de problème, comme des fluctuations de la tension du réseau, une surintensité, ou un courant de fuite, la borne de recharge peut interrompre la charge pour des raisons de sécurité.

Cela nécessite l'acquisition de la tension et de l'intensité du courant fournies par chaque borne ainsi que la surveillance des fuites à la terre (GFCI : Ground Fault Circuit Indicator).

La tension $u_{borne}(t)$ qui alimente le chargeur du véhicule devra être étudiée.



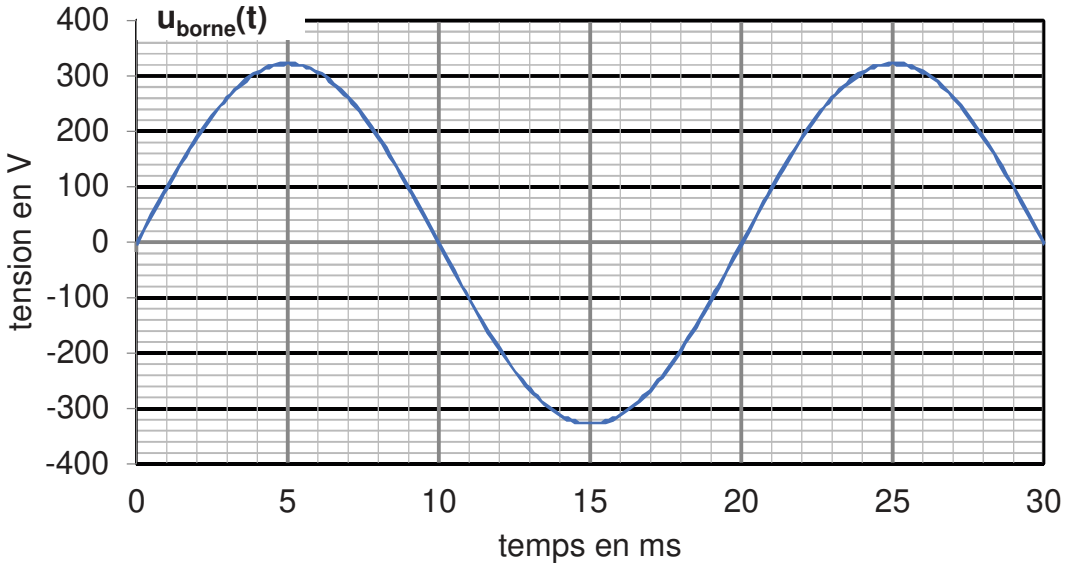
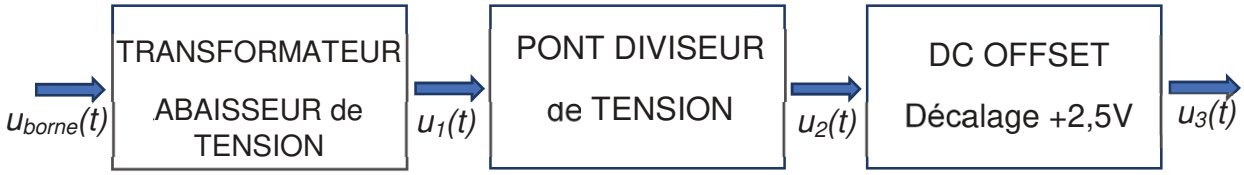


Figure 5 : chronogramme de la tension $u_{borne}(t)$ pour un fonctionnement normal

Problématique : vérifier que l'adaptation en tension avant la numérisation est compatible avec la plage d'entrée (0 à 5 V) du convertisseur analogique-numérique.

Pour effectuer une numérisation de la tension délivrée par la borne de recharge, il est indispensable de réaliser une isolation galvanique et d'adapter les niveaux de tension à l'entrée du convertisseur analogique-numérique du microcontrôleur.

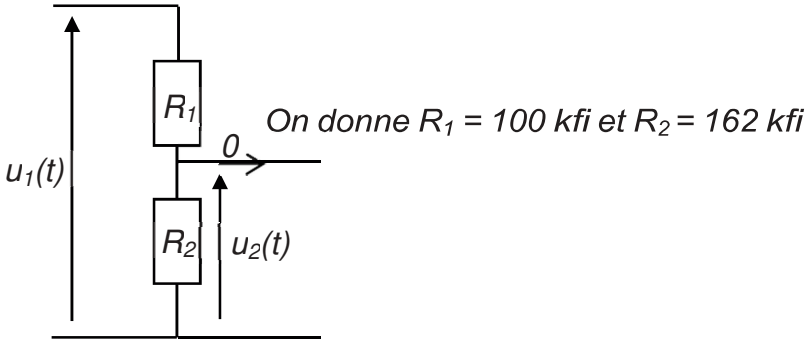
Pour cela on utilise le schéma de principe ci-dessous :



Avec : $u_1(t) = u_{borne}(t)/100$;

$u_3(t) = u_2(t) + 2{,}5$;

Schéma électrique du pont diviseur de tension :

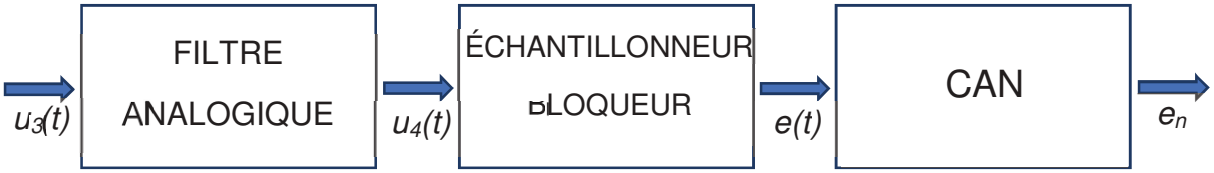


Q46. Montrer que $u_2(t) = 0{,}618 \cdot u_1(t)$ à partir du schéma électrique ci-dessus.

Q47. Déterminer, en s'aidant de la figure 5, l'amplitude $\hat{U}_{borne}$, en volts, de la tension sinusoïdale $u_{borne}(t)$.

Q48. Calculer, pour un fonctionnement normal du réseau, les valeurs minimale u_{3min} et maximale u_{3max} de $u_3(t)$.

La tension $u_3(t)$ est ensuite appliquée à l'entrée d'un filtre analogique qui est placé en amont d'un échantillonneur-bloqueur et du convertisseur analogique-numérique (CAN). La tension $u_3(t)$ peut, dans certaines situations anormales, comporter des harmoniques. Dans ce cas, les amplitudes des harmoniques dont la fréquence est en dehors de la bande passante du filtre analogique, seront considérées comme nulles.



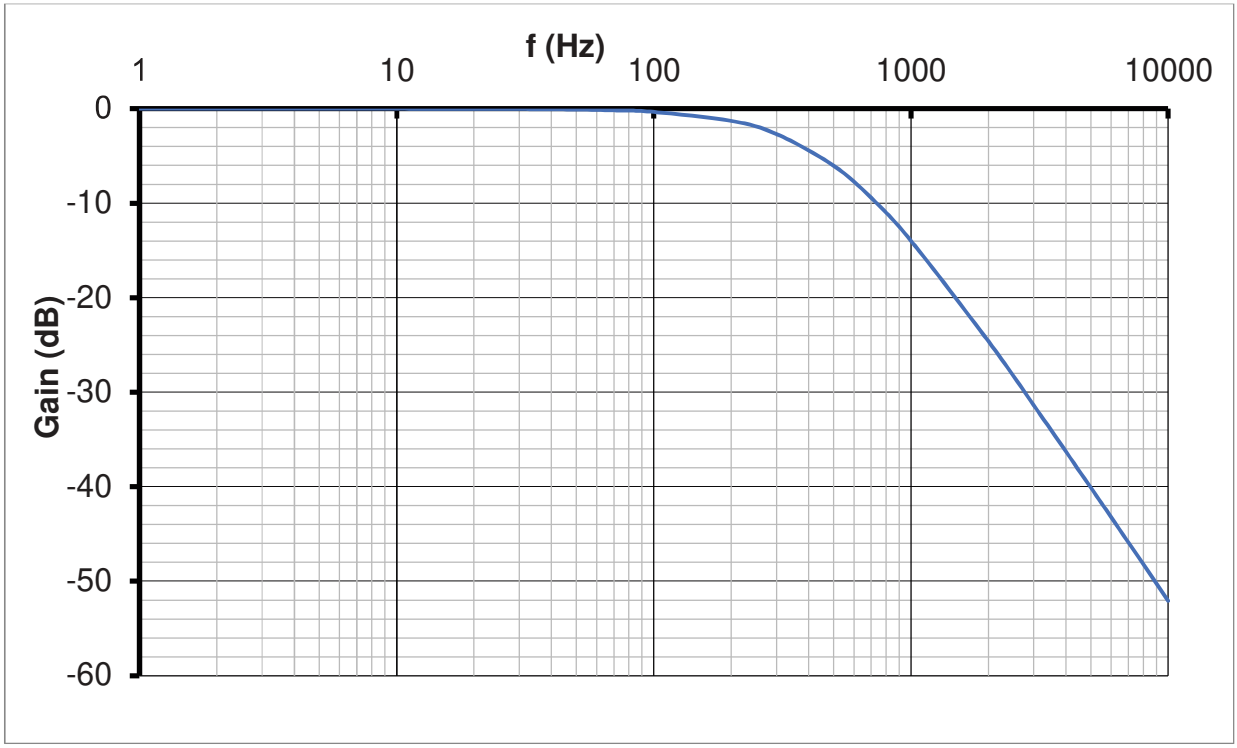


Figure 6 : diagramme du gain du filtre analogique en fonction de la fréquence.

Q49. Indiquer la nature du filtre analogique (passe-bas, passe-haut ou passe-bande). Préciser l'intérêt de ce filtre avant de numériser un signal analogique en général.

Q50. Relever la valeur du gain à la fréquence 50 Hz et en déduire que la plage de variation de $u_4(t)$ est bien compatible avec la plage d'entrée (0 V à 5 V) du convertisseur analogique numérique.

Problématique : vérifier que le CAN choisi permet la détection d'une variation de tension d'au moins 1 V de la tension $u_{borne}(t)$.

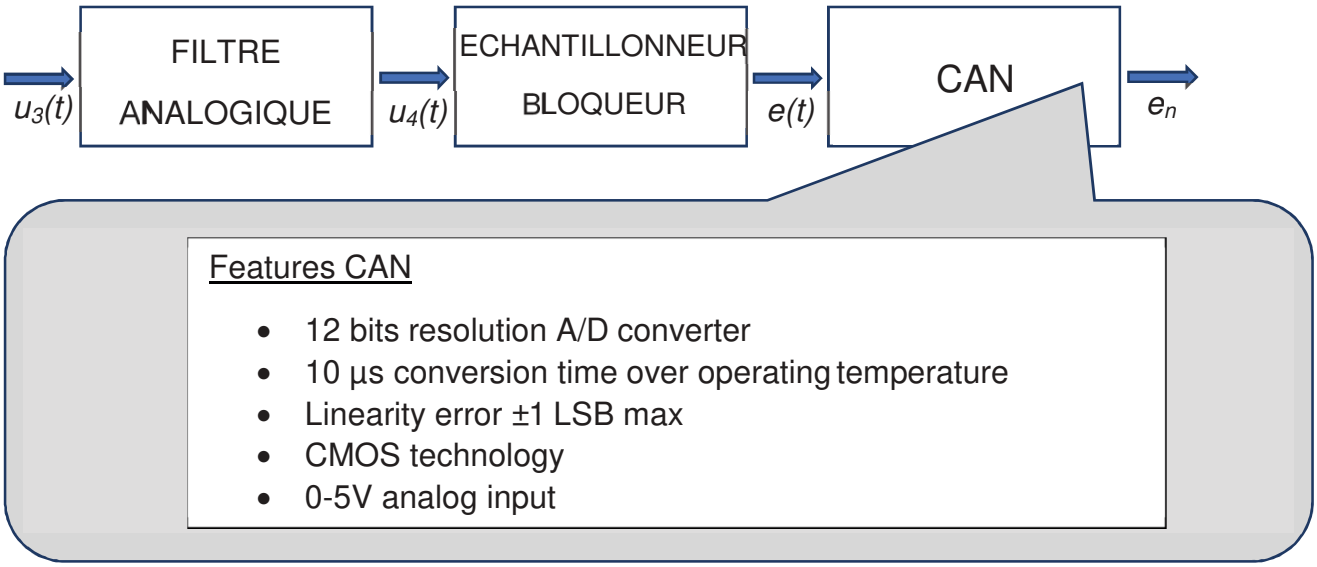


Figure 7 : extrait de la fiche technique du convertisseur analogique-numérique

Q51. Calculer le pas de quantification q, exprimé en volts, du CAN pour une plage de tension d'entrée de 0 V à 5 V.

Q52. Déterminer, en s'aidant de la figure 7, la résolution numérique minimale (nombre de bits) nécessaire au CAN pour détecter une variation de la tension $u_{borne}(t)$ de 1 V. En déduire si le nombre de bits du convertisseur est suffisant.

B.2- Traitement numérique du signal.

Le traitement numérique vise à isoler le fondamental du signal, afin de déterminer la valeur maximale de la composante sinusoïdale de fréquence 50 Hz.

Le filtre numérique a pour équation de récurrence :

$$s_n = a_0 \cdot e_n + a_2 \cdot e_{n-2} + b_1 \cdot s_{n-1} + b_2 \cdot s_{n-2}$$

Les échantillons e_n sont à l'entrée du filtre numérique et s_n à la sortie.

Problématique : vérification que le filtre numérique permet de sélectionner la composante sinusoïdale de fréquence 50 Hz.

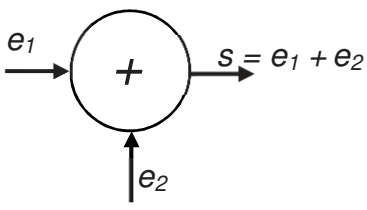
Q53. Proposer, en s'aidant de la figure 7 et en justifiant la réponse, un encadrement de la fréquence d'échantillonnage de la chaîne d'acquisition en tenant compte de :

- la fréquence de coupure du filtre (figure 6),
- la condition de Shannon,
- la durée de conversion du CAN.

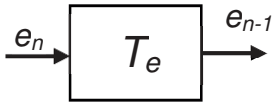
La fréquence d'échantillonnage fe est finalement fixée à 1,0 kHz.

Q54. Donner, en justifiant la réponse, la récursivité ou non de ce filtre.

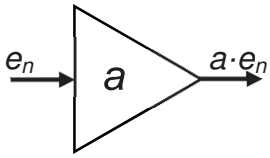
Q55. Dessiner une représentation structurelle de cet algorithme à l'aide des symboles suivants :



Addition



Retard d'une période d'échantillonnage T_e.



Multiplication de valeur a

Sachant que :

E(z) est la transformée en z de la séquence d'entrée $\{e_n\}$
S(z) est la transformée en z de la séquence de sortie $\{s_n\}$

Q56.Montrer que la fonction de transfert en z de ce filtre peut se mettre sous la forme :

$$H(z) = \frac{S(z)}{E(z)} = \frac{a_0 \cdot z^2 + a_2}{z^2 - b_1 \cdot z - b_2}$$

Les valeurs des coefficients de l'équation de récurrence sont les suivantes :

a_0 = 0,05945 ; a_2 = - 0,05945 ; b_1 = 1,793 et b_2 = - 0,8816

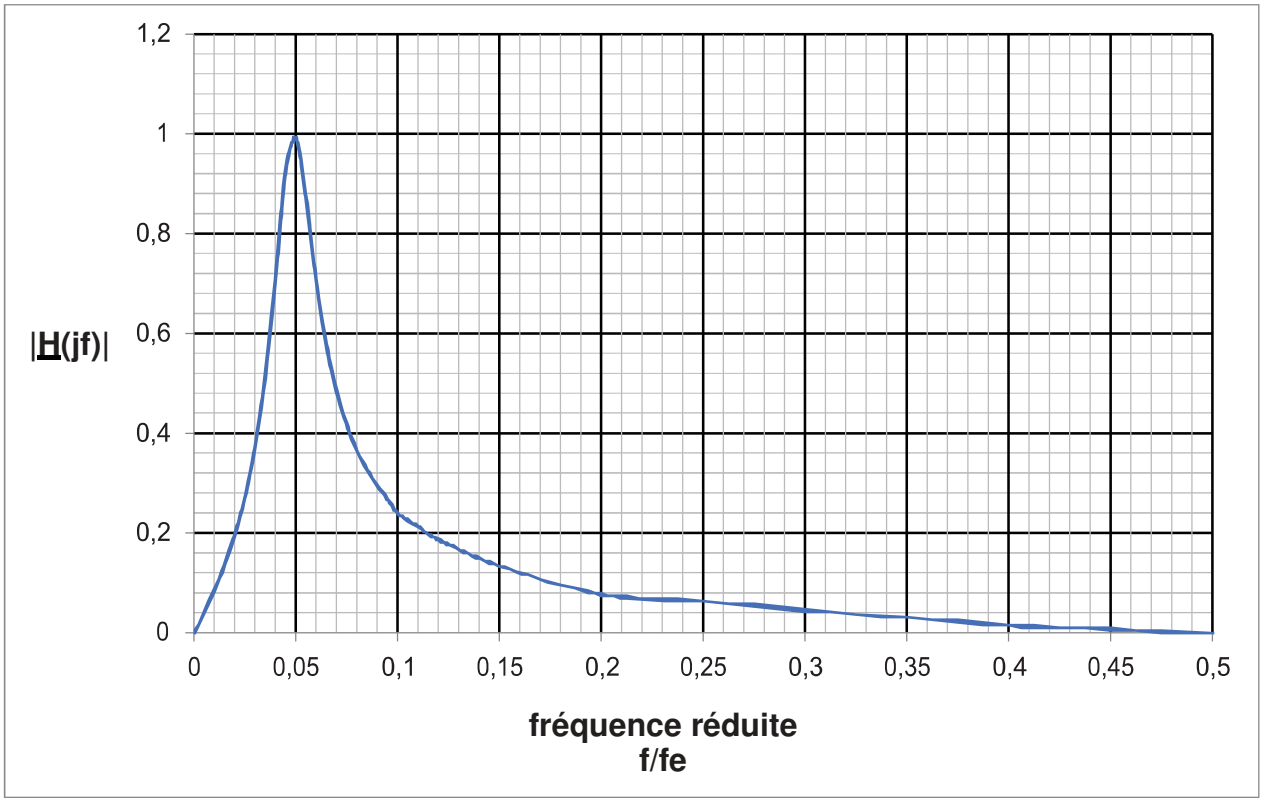


Figure 8 : module de la fonction de transfert H(jf) en fonction de la fréquence réduite f/fe.

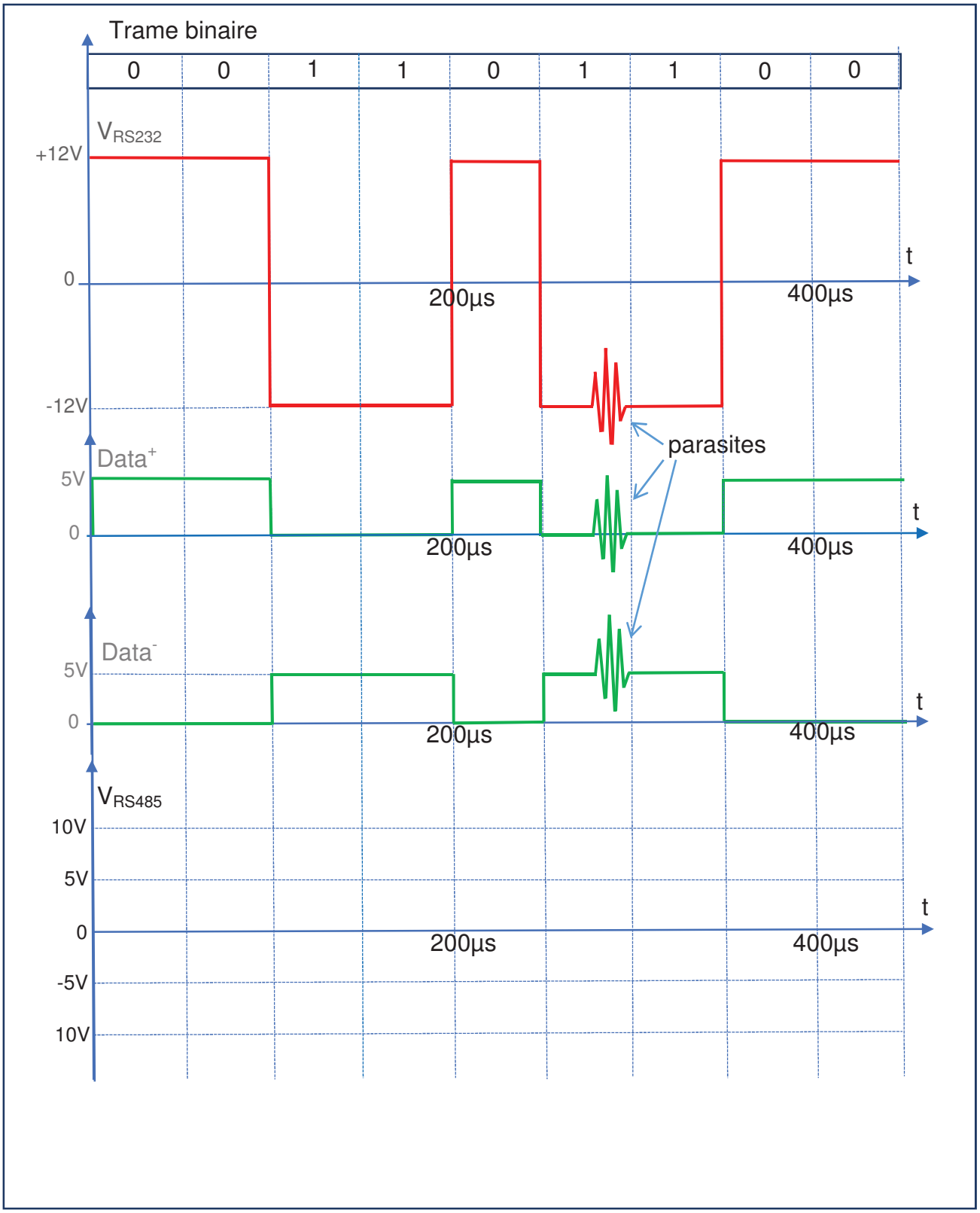
Q57. Préciser la nature du filtre.

Q58. Déterminer la fréquence de résonance f_o.

Q59. Justifier que le filtre numérique sélectionne bien le fondamenta l du signal d'entrée.

DOCUMENT RÉPONSES - Sciences Physiques

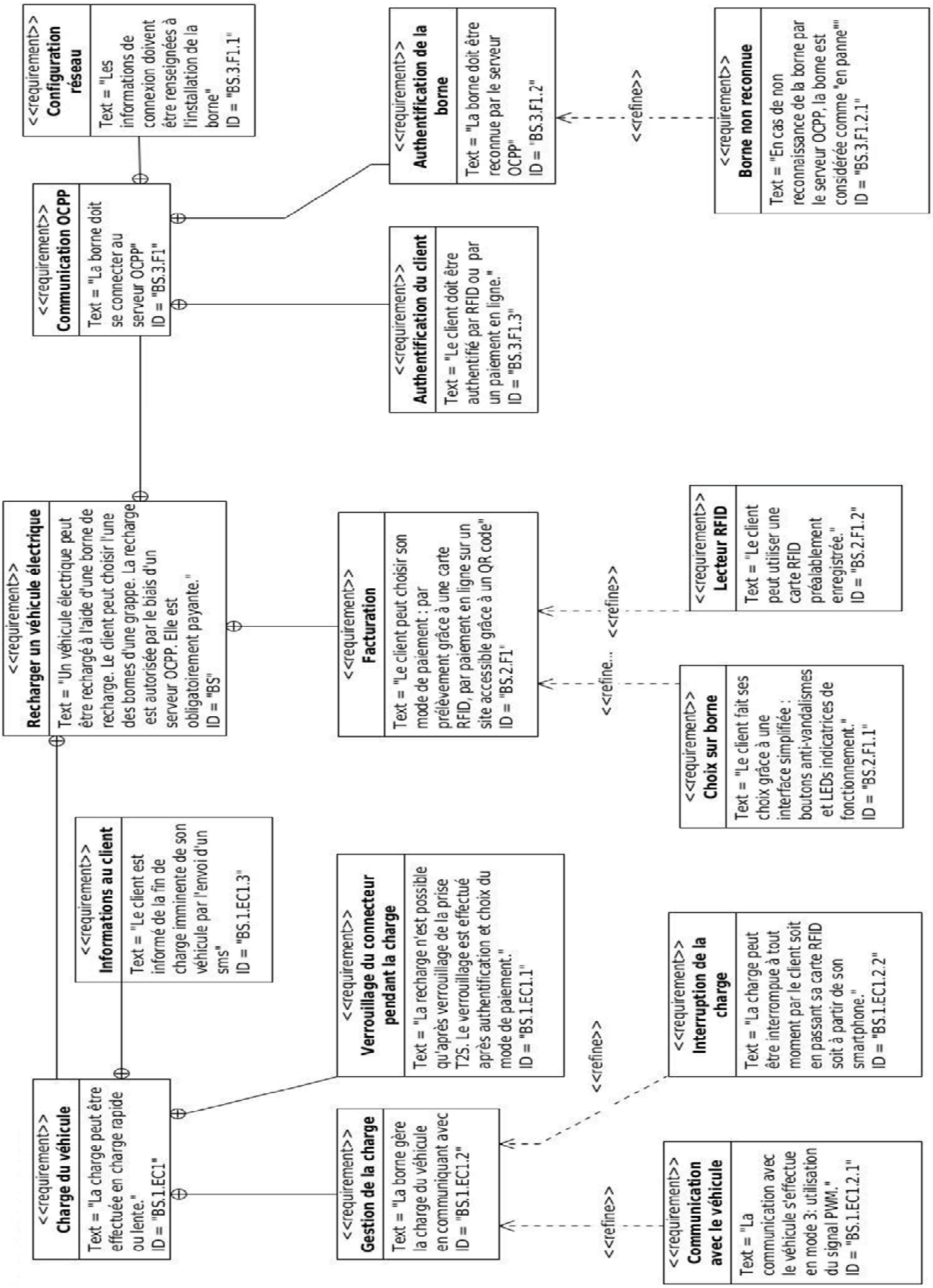
À RENDRE AVEC LA COPIE



Session 2020	BTS Systèmes Numériques Option A Informatique et Réseaux Épreuve E4	Page DR-SP1 sur 1
20SN4SNIR1	Sciences Physiques - Document réponses	

DOCUMENTATION

DOCUMENTATION PP1 : Diagramme des exigences



INVEO

RFID IND Modbus-Mif

General features

The reader is equipped with an RS485 port supporting Modbus RTU protocol and a USB port used for configuration and testing of the module.
The device has two relay outputs and two inputs.

Technical data:
Supply voltage:12-24VDC
Power supply: 40mA (12V)

Transponders:
Supported transponder standard: Mifare
Carrier frequency: 13,56 MHz
Reading distance to 10cm (depending on the type of transponder used)

Communication:
1 RS485 port – modbus RTU
1 USB port to configuration

Inputs/Outputs
2 relay outputs 1A@30VDC
2 inputs

Enclosure:
IP Rating: IP65

PEPPERL+FUCHS
IPH-FP-V1

Technical data

General specifications		
Operating frequency		125 kHz
Transfer rate		2 kBit/s
Sensing range		
Read distance		0 ... 100 mm
Write distance		0 ... 80 mm
Width		max. 80 mm
Operating distance		maximum: 100 mm
UL File Number		E87056
Functional safety related parameters		
$MTTF_d$		710 a
Mission Time (T_M)		10 a
Diagnostic Coverage (DC)		0 %
Indicators/operating means		
LED green/yellow		green: power on green flashing: read/write attempt performed yellow: data carrier detected
Electrical specifications		
Power consumption	P_0	≤ 1.2 W
Supply		from the IDENTControl
Ambient conditions		
Ambient temperature		-25 ... 70 °C (-13 ... 158 °F)
Storage temperature		-40 ... 85 °C (-40 ... 185 °F)
Mechanical specifications		
Degree of protection		IP67
Connection		M12 x 1 connector
Material		
Housing		PBT
Base		diecast aluminum
Encapsulation compound		Fermadur
Installation		
Distance between two heads		Multiplex on: ≥ 100 mm Multiplex off: ≥ 550 mm
Mass		approx. 380 g
Compliance with standards and directives		
Directive conformity		
R&TTE Directive 1995/5/EC		EN 301489-1 V1.8.1 (2008-04), EN 301489-3 V1.4.1 (2002-08), EN 300330-2 V1.3.1 (2006-04), EN 60950-1:2006

Telemecanique OsiSense XG series Smart Antenna

Characteristics		XGCS4901201 - format 40	XGCS8901201 - format 80	XGCS490B201	XGCS49LB201
Temperature	Operation	-25…+70°C (-13…158°F)		-40…+70°C (-40…158°F)	
	Storage	-40…+85 °C (-40…+185°F)		-40…+85 °C (-40…+185°F)	
Degree of protection		IP65 in accordance with IEC60529			IP69K on front faceIP65 on back part
Vibration resistance EN 60068.2.27 EN 60068.2.6		2 mm (0.078 in) from 5 to 29.5 Hz / 7 g (7 gn) from 29.5 to 150 Hz 30 g (30 gn) / 11 ms			
Resistance to mechanical shocks		IK04 according to EN 50102		IK04 according to EN 50102	
Standards / Certifications		CE, cULus, EN 300330-1/2, EN 301489-01/03, FCC Part 15 IC			
Immunity to disturbances		Resistance to electrostatic discharges, radiated electromagnetic fields, fast transients, electrical surges, conducted and induced interference and power frequency magnetic field according to IEC 61000/EN 55022.			
Unit dimensions		40x40x15 mm (1.57x1.57x0.59 in)	80x80x26 mm (3.15x3.15x1.02 in)	40x40x15 mm (1.57x1.57x0.59 in) Mounting ø22 mm (0.87 in)	
RFID frequency		13.56 MHz			
Type of associated tags		Standardized ISO 15693 and ISO 14443 tags Automatic detection of the tag type			
Nominal sensing distance (according to the associated tag)		18…70 mm (0.70…2.75 in)	20…100 mm (0.78…3.94 in)	10…70 mm (0.39…2.75 in)	
Nominal power supply		24 Vdc PELV			
Power supply voltage limits		19.2…29 V ripple included			
Power consumption		< 60 mA			
Serial links	Type	RS485			
	Protocol	Modbus RTU			
	Speed	9600…115 200 Bauds: Automatic detection			
Display		1 dual color LED for network communication 1 dual color LED for RFID communication (Tag present, Smart Antenna/tag dialog)			
Lights		-			2 Multicolor lights (7 colors)
Conne		5-way male M12 connector for connection to the communication network and power supply			
Tightening torque for the mounting		< 1 Nm (8.85 lbf-in)	< 3 Nm (26.55 lbf-in)	< 2.2 Nm (19.5 lbf-in)	

PEPPERL+FUCHS IPH-FP-V1

Technical data

General specifications		
Operating frequency		13.56 MHz
Transfer rate		26 kBit/s
Sensing range		
Read distance		0 ... 130 mm
Write distance		0 ... 130 mm
Width		max. 100 mm
UL File Number		E87056
Functional safety related parameters		
$MTTF_d$		680 a
Mission Time (T_M)		10 a
Diagnostic Coverage (DC)		0 %
Indicators/operating means		
LED red/green		Green: power on Flashing green: IO-Link communication Flashing red/green: IO-Link communication interrupted
LED blue/yellow		Blue: Write/read attempt performed Yellow: Read/write tag detected
Electrical specifications		
Rated operating voltage	U_e	20 ... 30 V DC , ripple 10 $\%_{SS}$
Power consumption	P_0	≤ 2 W
Interface		
Interface type		IO-Link
Protocol		IO-Link V1.1
Cycle time		min. 4 ms
Mode		COM 3 (230.4 kBaud)
Process data witdh		32 Byte
SIO mode support		no
Directive conformity		
Electromagnetic compatibility		
Directive 2014/30/EU		EN 61000-6-2:2005 EN 61000-6-4:2007
Radio and telecommunication terminal equipment		
Directive 2014/53/EU		EN 301489-1 V1.9.2:2011 EN 301489-3 V1.6.1:2013 EN 300330 V2.1.1:2017 EN 62368-1:2014+AC:2015 EN 50364:2010

BALLUFF

BIS M-620-068-A01-00-ST29

HF (13.56 MHz)

Display/Operation

(BB) Ready	Green LED
RF	LED yellow

Electrical connection

Connection (COM 1)	X1 (RS232/supply voltage): M12x1-Male, 8-pole
Connection slots	RCA-Female X2 (IN/OUT): M12x1-Female, 8-pole

Electrical data

Control input	1 (optocoupler isolated) PNP/NPN
Control output	2 (optocoupler isolated)
Current consumption max. at 24 V DC	500 mA
Input current max. at 24 V	28 mA
Operating voltage Ub	19.2...28.8 VDC
Operating voltage, output Vs	6...30 V DC
Output current max.	500 mA (500 mA ext. supply) 100 mA (int. supply)
Residual ripple max.	10 %
Voltage control	6...30 VDC

Environmental conditions

Ambient temperature	-20...50 °C
Continuous shock load	yes
EN 60068-2-27, Shock	yes
EN 60068-2-32 Free fall	yes
EN 60068-2-6, Vibration	yes
IP rating	IP65 with connector
Storage temperature	-20...70 °C

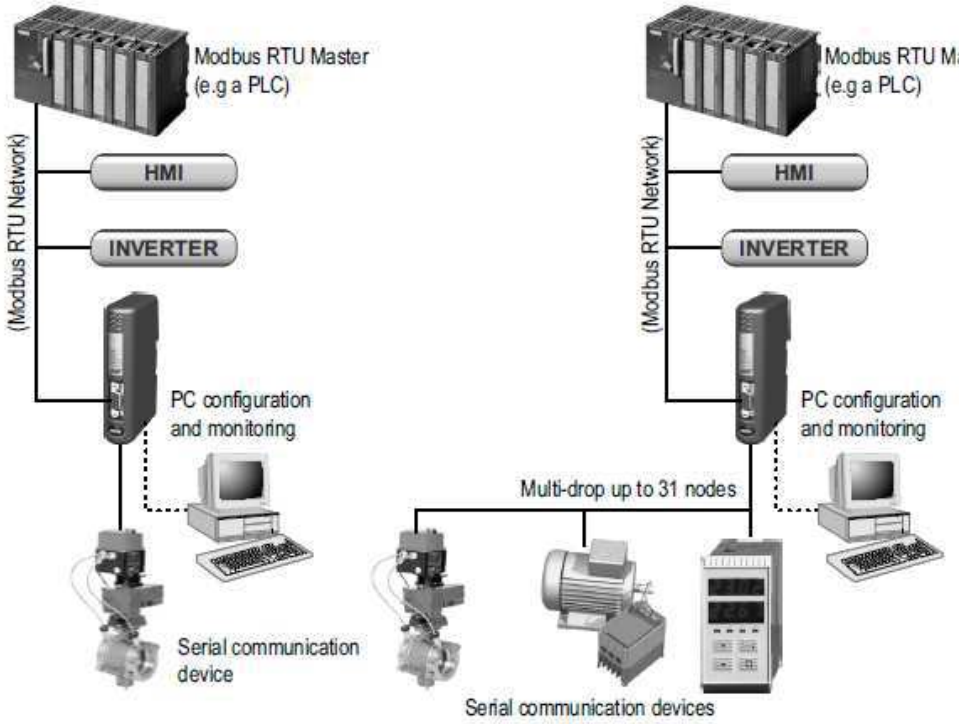
Output/Interface

Interface	RS232

DOCUMENTATION PP3 : AnyBus Communicator

About the Anybus Communicator for Modbus RTU

The Anybus Communicator for Modbus RTU acts as a gateway between virtually any serial application protocol and a Modbus RTU-based network. Integration of industrial devices is enabled without loss of functionality, control and reliability, both when retro-fitting to existing equipment as well as when setting up new installations.



Single-Node Serial Sub Network **Multi-Node Serial Sub Network**

Subnetwork

The gateway can address up to 31 nodes, and supports the following physical standards:

- RS-232
- RS-422
- RS-485

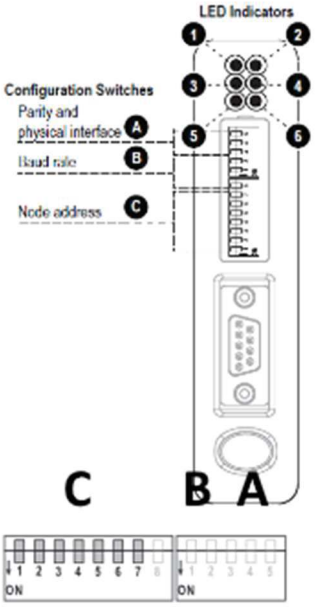
Modbus RTU Interface

Modbus RTU connectivity is provided through patented Anybus technology, a proven industrial communication solution used all over the world by leading manufacturers of industrial automation products.

- Galvanically isolated bus interface
- Coil and Register access
- RS-232 or RS-485 operation
- On-board configuration switches
- 1200... 57600bps operation

Passerelle ModBus

Configuration Switches



The configuration switches determines the basic communication settings for the Modbus interface. Normally, these switches are covered by a plastic hatch. When removing the hatch, avoid touching the circuit boards and components. If tools are used to open the hatch, use caution.

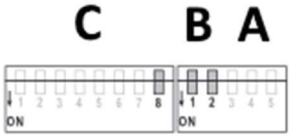
Note that these settings cannot be changed during runtime, i.e. the gateway must be restarted in order for any changes to have effect.

Node Address

Node Address	Sw. 1	Sw. 2	Sw. 3	Sw. 4	Sw. 5	Sw. 6	Sw. 7
(reserved)	OFF	OFF	OFF	OFF	OFF	OFF	OFF
1	OFF	OFF	OFF	OFF	OFF	OFF	ON
2	OFF	OFF	OFF	OFF	OFF	ON	OF
...	...	...	...	...	...	...	...
126	ON	ON	ON	ON	ON	ON	OFF
127	ON	ON	ON	ON	ON	ON	ON

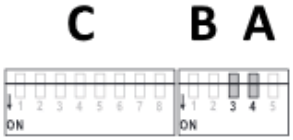
Baudrate Configuration

Baudrate	Sw. 8	Sw. 1	Sw. 2
(reserved)	OFF	OFF	OFF
1200 bps	OFF	OFF	ON
2400 bps	OFF	ON	OFF
4800 bps	OFF	ON	ON
9600 bps	ON	OFF	OFF
19200 bps (standard)	ON	OFF	ON
38400 bps	ON	ON	OFF
57600 bps	ON	ON	ON



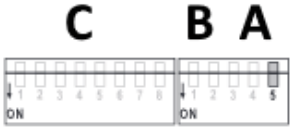
Parity & Stop Bits

Parity	Sw. 3	Sw. 4
(reserved)	OFF	OFF
No parity, 2 stop bits	OFF	ON
Even parity, 1 stop bit	ON	OFF
Odd parity, 1 stop bit	ON	ON



Physical Interface

Interface Type	Sw. 5
RS-485	OFF
RS-232	ON



DOCUMENTATION PP4 : « *ModBus over Serial link* »

MODBUS Data Link Layer

MODBUS Master / Slaves protocol principle

The MODBUS Serial Line protocol is a Master-Slaves protocol. Only one master (at the same time) is connected to the bus, and one or several (247 maximum number) slaves nodes are also connected to the same serial bus. A MODBUS communication is always initiated by the master. The slave nodes will never transmit data without receiving a request from the master node. The slave nodes will never communicate with each other. The master node initiates only one MODBUS transaction at the same time.

The master node issues a MODBUS request to the slave nodes in two modes :

➔ In **unicast mode**, the master addresses an individual slave. After receiving and processing the request, the slave returns a message (a 'reply') to the master.

In that mode, a MODBUS transaction consists of 2 messages : a request from the master, and a reply from the slave.

Each slave must have an unique address (from 1 to 247) so that it can be addressed independently from other nodes.

➔ In **broadcast mode**, the master can send a request to all slaves.

No response is returned to broadcast requests sent by the master. The broadcast requests are necessarily writing commands. **All devices must accept the broadcast for writing function**. The address 0 is reserved to identify a broadcast exchange.

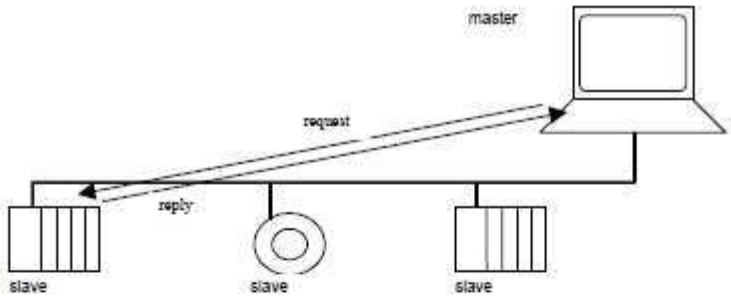


Figure 3: Unicast mode

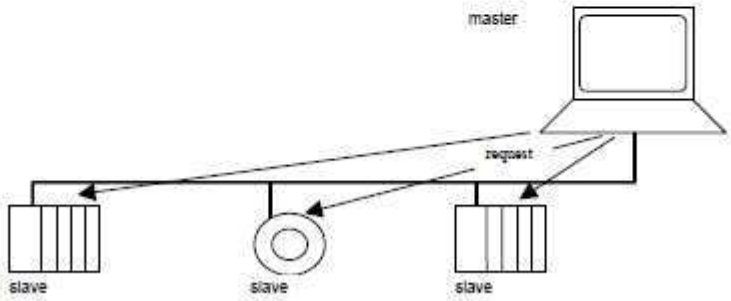
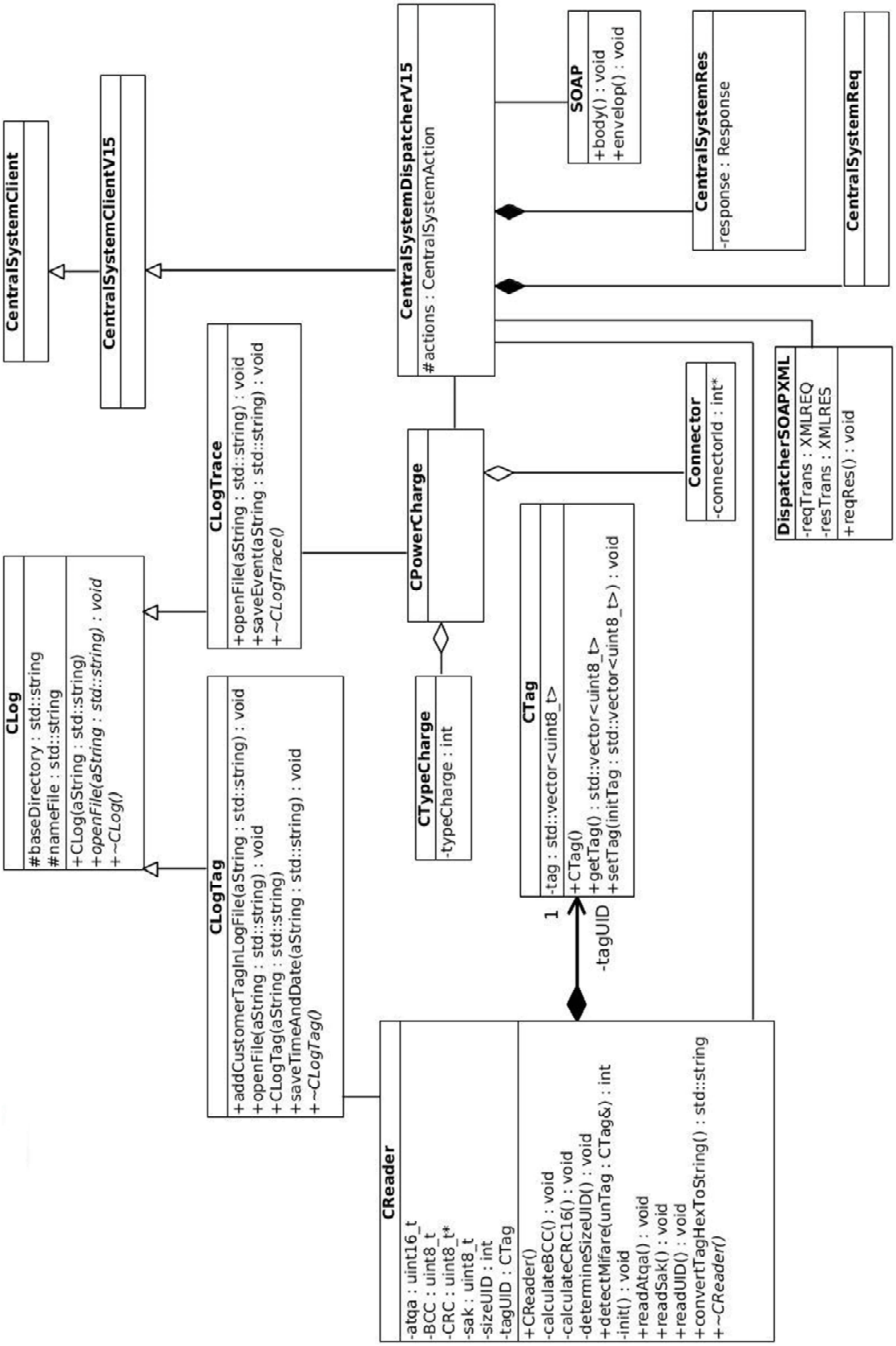


Figure 4: Broadcast mode

DOCUMENTATION PP5 : Diagramme de classes (Borne)



DOCUMENTATION PP6 : ATQA Coding of NXP Contactless Card ICs

Table ATQA Coding of NXP Contactless Card ICs

X: depends on the COS

Bit number	Hex Value	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
ISO/IEC 14443-3		RFU				Proprietary				UID size		RFU	Bit Frame Anti-collision				
MIFARE Plus 2K (4 Byte UID or 4 Byte RID)	00 04	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
MIFARE Plus EV1 2K (4 Byte UID or 4 Byte RID)	00 04	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
MIFARE Plus 4K (4 Byte UID or 4 Byte RID)	00 02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
MIFARE Plus EV1 4K (4 Byte UID or 4 Byte RID)	00 02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
MIFARE Plus 2K (7 Byte UID)	00 44	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
MIFARE Plus EV1 2K (7 Byte UID)	00 44	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
MIFARE Plus 4K (7 Byte UID)	00 42	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
MIFARE Plus EV1 4K (7 Byte UID)	00 42	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
MIFARE DESFire	03 44	0	0	0	0	0	0	1	1	0	1	0	0	0	1	0	0
MIFARE DESFire EV1	03 44	0	0	0	0	0	0	1	1	0	1	0	0	0	1	0	0
P3SR008	00 44	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

2. [2] The 7 byte UID MIFARE Mini has bit 7 = 1, even if the 4 byte NUID mapping is enabled.
3. [3] The 7 byte UID MIFARE Classic 1K has bit 7 = 1, even if the 4 byte NUID mapping is enabled.
4. [4] The 7 byte UID MIFARE Classic 4K has bit 7 = 1, even if the 4 byte NUID mapping is enabled.

DOCUMENTATION PP7 : Principales requêtes SQL

Utiliser (rendre active) une base de données existante :	`USE nom_de_la_base;`
Créer une base de données :	`CREATE DATABASE nom_de_la_base;`
Supprimer une base de données	`DROP DATABASE nom_de_la_base;`
Créer une table dans la base de données active:	`CREATE TABLE nomTable (id INT NOT NULL AUTO_INCREMENT, champ1 DOUBLE, champ2 VARCHAR, champ3 TIMESTAMP NOT NULL, …, PRIMARY KEY(id)) ;`
Lister la structure d'une table :	`DESCRIBE nomTable;`
Sélectionner toutes les informations de la table :	`SELECT * FROM nomTable ;`
Sélectionner seulement les informations d'un champ	`SELECT nomChamp FROM nomTable ;`
Sélectionner tous les champs de la table nomTable correspondant à deux critères.	`SELECT * FROM nomTable WHERE nomChamp1 = 'poste' AND nomChamp3 < 12 ;`

Sélectionner sur plusieurs tables (jointure)
`nomTable1.nomChamp1 est clé primaire.`
`nomTable2.nomChamp4 est une clé étrangère vers nomTable1.`
	`SELECT * FROM nomTable1, nomTable2 WHERE nom_table1.nomChamp1 = nom_table2.nomChamp4 ;`

Écrire une nouvelle entrée dans une table de BDD	`INSERT INTO nomTable( champ1, champ2) VALUES( 'valeur1', 'valeur2') ;`
Modifier les informations de l'entrée dont le champ id = 51	`UPDATE nomTable SET nomChamp1=10, valeur2=32 WHERE id=51 ;`
Ajouter des nouveaux champs (colonnes) dans une table	`ALTER TABLE nomTable ADD champ1 DOUBLE, ADD champ2 BOOLEAN DEFAULT FALSE ;`

Les mots en gras dans la colonne de droite sont des mots réservés par le langage SQL.

DOCUMENTATION PP8 : SOAP

SOAP (*from Wikipedia*)

SOAP (abbreviation for **Simple Object Access Protocol**) is a messaging protocol specification for exchanging structured information in the implementation of web services in computer networks. Its purpose is to provide extensibility, neutrality and independence. It uses XML Information Set for its message format, and relies on application layer protocols, most often Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

SOAP allows processes running on disparate operating systems (such as Windows and Linux) to communicate using Extensible Markup Language (XML). Since Web protocols like HTTP are installed and running on all operating systems, SOAP allows clients to invoke web services and receive responses independent of language and platforms.

Characteristics

SOAP provides the Messaging Protocol layer of a web services protocol stack for web services. It is an XML-based protocol consisting of three parts:

- an envelope, which defines the message structure and how to process it
- a set of encoding rules for expressing instances of application-defined datatypes
- a convention for representing procedure calls and responses

SOAP has three major characteristics:

1. *extensibility* (security and WS-Addressing are among the extensions under development)
2. *neutrality* (SOAP can operate over any protocol such as HTTP, SMTP, TCP, UDP, or JMS)
3. *independence* (SOAP allows for any programming model)

As an example of what SOAP procedures can do, an application can send a SOAP request to a server that has web services enabled—such as a real-estate price database—with the parameters for a search. The server then returns a SOAP response (an XML-formatted document with the resulting data), e.g., prices, location, features. Since the generated data comes in a standardized machine-parsable format, the requesting application can then integrate it directly.

The SOAP architecture consists of several layers of specifications for:

- message format
- Message Exchange Patterns (MEP)
- underlying transport protocol bindings
- message processing models
- protocol extensibility

SOAP evolved as a successor of XML-RPC, though it borrows its transport and interaction neutrality from Web Service Addressing and the envelope/header/body from elsewhere (probably from WDDX).

SOAP terminology

SOAP specification can be broadly defined to be consisting of the following 3 conceptual components: protocol concepts, encapsulation concepts and network concepts.

Data encapsulation concepts

- **SOAP message**: Represents the information being exchanged between 2 SOAP nodes.
- **SOAP envelope** : As per its name, it is the enclosing element of an XML message identifying it as a SOAP message.
- **SOAP header block**: A SOAP header can contain more than one of these blocks, each being a discrete computational block within the header. In general, the SOAP *role* information is used to target nodes on the path. A header block is said to be targeted at a SOAP node if the SOAP role for

the header block is the name of a role in which the SOAP node operates. (ex: A SOAP header block with role attribute as *ultimateReceiver* is targeted only at the destination node which has this role. A header with a role attribute as *next* is targeted at each intermediary as well as the destination node.)

- **SOAP header** : A collection of one or more header blocks targeted at each SOAP receiver.
- **SOAP body** : Contains the body of the message intended for the SOAP receiver. The interpretation and processing of SOAP body is defined by header blocks.
- **SOAP fault**: In case a SOAP node fails to process a SOAP message, it adds the fault information to the SOAP fault element. This element is contained within the SOAP body as a child element.

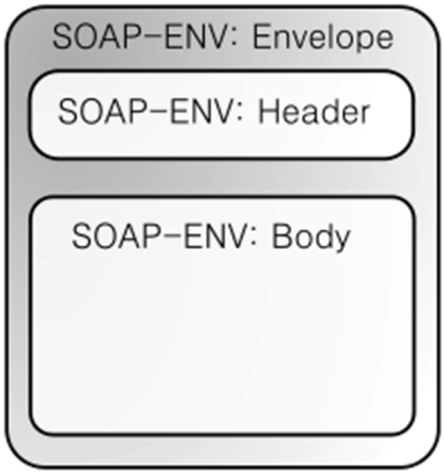
Message sender and receiver concepts

- **SOAP sender**: The node that transmits a SOAP message.
- **SOAP receiver** : The node receiving a SOAP message. (Could be an intermediary or the destination node.)
- **SOAP message path** : The path consisting of all the nodes that the SOAP message traversed to reach the destination node.
- **Initial SOAP sender**: This is the node which originated the SOAP message to be transmitted. This is the root of the SOAP message path.
- **SOAP intermediary**: All the nodes in between the SOAP originator and the intended SOAP destination. It processes the SOAP header blocks targeted at it and acts to forward a SOAP message towards an ultimate SOAP receiver.
- **Ultimate SOAP receiver**: The destination receiver of the SOAP message. This node is responsible for processing the message body and any header blocks targeted at it.

Specification

The SOAP specification defines the messaging framework, which consists of:

- The ***SOAP processing model*** defining the rules for processing a SOAP message
- The ***SOAP extensibility model*** defining the concepts of SOAP features and SOAP modules
- The ***SOAP underlying protocol binding*** framework describing the rules for defining a binding to an underlying protocol that can be used for exchanging SOAP messages between SOAP nodes
- The ***SOAP message construct*** defining the structure of a SOAP message



1 SOAP Structure

SOAP building blocks

A SOAP message is an ordinary XML document containing the following elements:

Element	Description	Required
Envelope	Identifies the XML document as a SOAP message.	Yes
Header	Contains header information.	No
Body	Contains call, and response information.	Yes
Fault	Provides information about errors that occurred while processing the message.	No

Transport methods

Both SMTP and HTTP are valid application layer protocols used as transport for SOAP, but HTTP has gained wider acceptance as it works well with today's internet infrastructure; specifically, HTTP works well with network firewalls. SOAP may also be used over HTTPS (which is the same protocol as HTTP at the application level, but uses an encrypted transport protocol underneath) with either simple or mutual authentication; this is the advocated WS-I method to provide web service security as stated in the WS-I Basic Profile

This is a major advantage over other distributed protocols like GIOP/IIOP or DCOM, which are normally filtered by firewalls. SOAP over AMQP is yet another possibility that some implementations support. SOAP also has an advantage over DCOM that it is unaffected by security rights configured on the machines that require knowledge of both transmitting and receiving nodes. This lets SOAP be loosely coupled in a way that is not possible with DCOM. There is also the SOAP-over-UDP OASIS standard.

Example message (encapsulated in HTTP)

```
  POST /InStock HTTP/1.1
  Host: www.example.org
  Content-Type: application/soap+xml; charset=utf-8
  Content-Length: 299
  SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

  <?xml version="1.0"?>
  <soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:m="http://www.example.org">
    <soap:Header>
    </soap:Header>
    <soap:Body>
      <m:GetStockPrice>
        <m:StockName>GOOG</m:StockName>
      </m:GetStockPrice>
    </soap:Body>
  </soap:Envelope>
```

Dialogue 1 : Borne -> Supervision

No.	Time	Source	Destination	Protocol	Length	Info
8	2.122297	192.168.0.241	192.168.0.102	TCP	66	8080 → 44112 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
9	2.122717	192.168.0.102	192.168.0.241	TCP	60	44112 → 8080 [ACK] Seq=1 Ack=1 Win=29200 Len=0
10	2.128202	192.168.0.102	192.168.0.241	HTTP/XML	1401	POST /steve/services/CentralSystemService HTTP/1.1
11	2.159196	192.168.0.241	192.168.0.102	TCP	800	8080 → 44112 [PSH, ACK] Seq=1 Ack=1348 Win=1051136 Len=746 [TCP segment of a reassembled
12	2.159415	192.168.0.241	192.168.0.102	HTTP/XML	54	HTTP/1.1 200 OK
13	2.160003	192.168.0.102	192.168.0.241	TCP	60	44112 → 8080 [ACK] Seq=1348 Ack=747 Win=32120 Len=0
14	2.163539	192.168.0.102	192.168.0.241	TCP	60	44112 → 8080 [FIN, ACK] Seq=1348 Ack=748 Win=32120 Len=0

```
Frame 10: 1401 bytes on wire (11208 bits), 1401 bytes captured (11208 bits) on interface 0
Ethernet II, Src: Telemech_42:86:06 (00:80:f4:42:86:06), Dst: LcfcHefe_be:9c:76 (c8:5b:76:be:9c:76)
Internet Protocol Version 4, Src: 192.168.0.102, Dst: 192.168.0.241
Transmission Control Protocol, Src Port: 44112, Dst Port: 8080, Seq: 1, Ack: 1, Len: 1347
Hypertext Transfer Protocol
eXtensible Markup Language
<?xml
    version="1.0"
    encoding="UTF-8"
    ?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
    xmlns:SOAP-ENC="http://www.w3.org/2003/05/soap-encoding"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:cp="urn://Ocpp/Cp/2012/06/"
    xmlns:chan="http://schemas.microsoft.com/ws/2005/02/duplex"
    xmlns:wsa5="http://www.w3.org/2005/08/addressing"
    xmlns:cs="urn://Ocpp/Cs/2012/06/">
  <SOAP-ENV:Header>
    <cs:chargeBoxIdentity>
      EV1234
      </cs:chargeBoxIdentity>
    <wsa5:MessageID>
      urn:uuid:d7870b21-ab67-47ba-81a8-eb745829e117
      </wsa5:MessageID>
    <wsa5:From>
      <wsa5:Address>
        http://192.168.0.102:8080/
        </wsa5:Address>
      </wsa5:From>
      <wsa5:ReplyTo>
        <wsa5:Address>
          http://www.w3.org/2005/08/addressing/anonymous
          </wsa5:Address>
        </wsa5:ReplyTo>
      <wsa5:To
        SOAP-ENV:mustUnderstand="true">
        http://192.168.0.241:8080/steve/services/CentralSystemService
        </wsa5:To>
    <wsa5:Action
        SOAP-ENV:mustUnderstand="true">
        /Authorize
        </wsa5:Action>
      </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <cs:authorizeRequest>
      <cs:idTag>
        0780BA2305625D
        </cs:idTag>
      </cs:authorizeRequest>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Dialogue 1 : Supervision -> Borne

```
> Frame 12: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: LcfcHefe_be:9c:76 (c8:5b:76:be:9c:76), Dst: Telemech_42:86:06 (00:80:f4:42:86:06)
> Internet Protocol Version 4, Src: 192.168.0.241, Dst: 192.168.0.102
> Transmission Control Protocol, Src Port: 8080, Dst Port: 44112, Seq: 747, Ack: 1348, Len: 0
> [2 Reassembled TCP Segments (746 bytes): #11(746), #12(0)]
> Hypertext Transfer Protocol
v eXtensible Markup Language
  v <soap:Envelope
      xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
    v <soap:Header>
      v <Action
          xmlns="http://www.w3.org/2005/08/addressing">
          /AuthorizeResponse
          </Action>
      v <MessageID
          xmlns="http://www.w3.org/2005/08/addressing">
          urn:uuid:a5f79918-bf4e-4018-aa3f-dad2f8d8ba19
          </MessageID>
      v <To
          xmlns="http://www.w3.org/2005/08/addressing">
          http://www.w3.org/2005/08/addressing/anonymous
          </To>
      v <RelatesTo
          xmlns="http://www.w3.org/2005/08/addressing">
          urn:uuid:d7870b21-ab67-47ba-81a8-eb745829e117
          </RelatesTo>
        </soap:Header>
    v <soap:Body>
      v <authorizeResponse
          xmlns="urn://Ocpp/Cs/2012/06/">
        v <idTagInfo>
          v <status>
              Invalid
              </status>
            </idTagInfo>
          </authorizeResponse>
        </soap:Body>
    </soap:Envelope>
```

Dialogue 2 : Borne -> Supervision

No.	Time	Source	Destination	Protocol	Length	Info
51	7.522515	192.168.0.241	192.168.0.102	TCP	66	8080 → 44118 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=14
52	7.522884	192.168.0.102	192.168.0.241	TCP	60	44118 → 8080 [ACK] Seq=1 Ack=1 Win=29200 Len=0
53	7.525993	192.168.0.102	192.168.0.241	HTTP/XML	1401	POST /steve/services/CentralSystemService HTTP/1.1
54	7.561072	192.168.0.241	192.168.0.102	TCP	850	8080 → 44118 [PSH, ACK] Seq=1 Ack=1348 Win=1051136 Len=796
55	7.561372	192.168.0.241	192.168.0.102	HTTP/XML	54	HTTP/1.1 200 OK

```
> Frame 53: 1401 bytes on wire (11208 bits), 1401 bytes captured (11208 bits) on interface 0
> Ethernet II, Src: Telemech_42:86:06 (00:80:f4:42:86:06), Dst: LcfcHefe_be:9c:76 (c8:5b:76:be:9c:76)
> Internet Protocol Version 4, Src: 192.168.0.102, Dst: 192.168.0.241
> Transmission Control Protocol, Src Port: 44118, Dst Port: 8080, Seq: 1, Ack: 1, Len: 1347
> Hypertext Transfer Protocol
v eXtensible Markup Language
  v <?xml
      version="1.0"
      encoding="UTF-8"
      ?>
  v <SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
      xmlns:SOAP-ENC="http://www.w3.org/2003/05/soap-encoding"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:cp="urn://Ocpp/Cp/2012/06/"
      xmlns:chan="http://schemas.microsoft.com/ws/2005/02/duplex"
      xmlns:wsa5="http://www.w3.org/2005/08/addressing"
      xmlns:cs="urn://Ocpp/Cs/2012/06/">
    v <SOAP-ENV:Header>
      v <cs:chargeBoxIdentity>
          EV1234
          </cs:chargeBoxIdentity>
      v <wsa5:MessageID>
          urn:uuid:79c19bb1-af01-48c1-9981-43dc34a055fa
          </wsa5:MessageID>
      v <wsa5:From>
        v <wsa5:Address>
            http://192.168.0.102:8080/
            </wsa5:Address>
          </wsa5:From>
      v <wsa5:ReplyTo>
        v <wsa5:Address>
            http://www.w3.org/2005/08/addressing/anonymous
            </wsa5:Address>
          </wsa5:ReplyTo>
      v <wsa5:To
          SOAP-ENV:mustUnderstand="true">
          http://192.168.0.241:8080/steve/services/CentralSystemService
          </wsa5:To>
      v <wsa5:Action
          SOAP-ENV:mustUnderstand="true">
          /Authorize
          </wsa5:Action>
        </SOAP-ENV:Header>
    v <SOAP-ENV:Body>
      v <cs:authorizeRequest>
        v <cs:idTag>
            0780BA23056776
            </cs:idTag>
          </cs:authorizeRequest>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Dialogue 2 : Supervision -> Borne

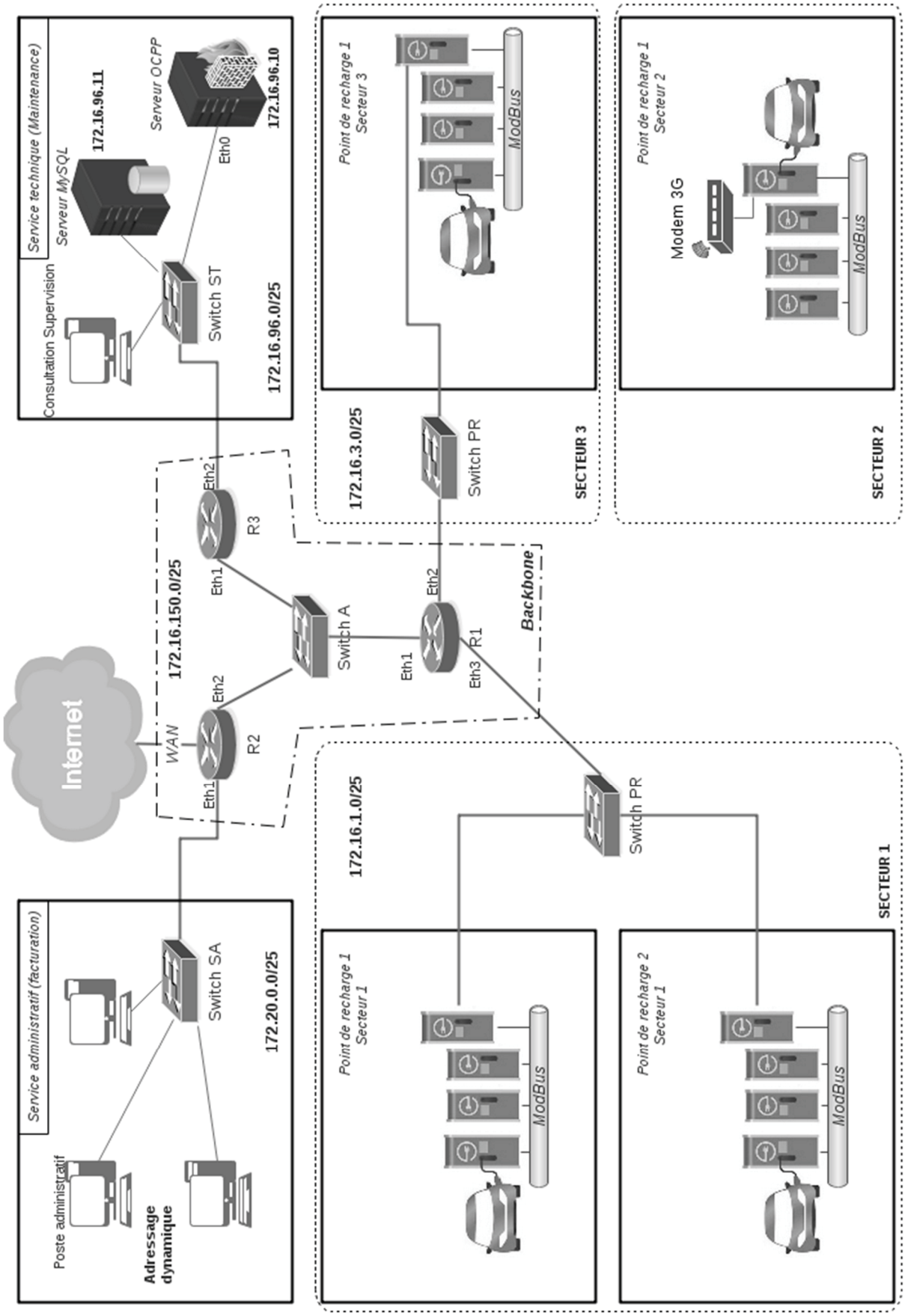
```
> Frame 55: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: LcfcHefe_be:9c:76 (c8:5b:76:be:9c:76), Dst: Telemech_42:86:06 (00:80:f4:42:86:06)
> Internet Protocol Version 4, Src: 192.168.0.241, Dst: 192.168.0.102
> Transmission Control Protocol, Src Port: 8080, Dst Port: 44118, Seq: 797, Ack: 1348, Len: 0
> [2 Reassembled TCP Segments (796 bytes): #54(796), #55(0)]
> Hypertext Transfer Protocol
v eXtensible Markup Language
  v <soap:Envelope
      xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
    v <soap:Header>
      v <Action
          xmlns="http://www.w3.org/2005/08/addressing">
          /AuthorizeResponse
          </Action>
      v <MessageID
          xmlns="http://www.w3.org/2005/08/addressing">
          urn:uuid:4d4500bf-8234-4e73-b3eb-070db29fed7a
          </MessageID>
      v <To
          xmlns="http://www.w3.org/2005/08/addressing">
          http://www.w3.org/2005/08/addressing/anonymous
          </To>
      v <RelatesTo
          xmlns="http://www.w3.org/2005/08/addressing">
          urn:uuid:79c19bb1-af01-48c1-9981-43dc34a055fa
          </RelatesTo>
        </soap:Header>
    v <soap:Body>
      v <authorizeResponse
          xmlns="urn://Ocpp/Cs/2012/06/">
        v <idTagInfo>
          v <status>
              Accepted
              </status>
          v <expiryDate>
              2019-08-13T16:53:05.178Z
              </expiryDate>
            </idTagInfo>
          </authorizeResponse>
        </soap:Body>
</soap:Envelope>
```

DOCUMENTATION PP10 : Extrait du fichier WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  xmlns:s="http://www.w3.org/2001/XMLSchema"
                  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
                  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                  xmlns:tns="urn://Ocpp/Cs/2012/06/"
                  targetNamespace="urn://Ocpp/Cs/2012/06/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
                  xmlns:wsa="http://www.w3.org/2005/08/addressing"
                  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
                  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

    <wsdl:types>
    <s:schema targetNamespace="urn://Ocpp/Cs/2012/06/" elementFormDefault="qualified">

      <!-- Begin of types shared with ChargePointService -->
      <s:simpleType name="IdToken">
        <s:annotation>
          <s:documentation>Type of string defining identification token, e.g. RFID or credit card number. To be treated as case insensitive.</s:documentation>
        </s:annotation>
        <s:restriction base="s:string">
          <s:maxLength value="20"/>
        </s:restriction>
      </s:simpleType>
        <s:simpleType name="AuthorizationStatus">
        <s:annotation>
          <s:documentation>Defines the authorization-status-value</s:documentation>
        </s:annotation>
        <s:restriction base="s:string">
          <s:enumeration value="Accepted"/>
          <s:enumeration value="Blocked"/>
          <s:enumeration value="Expired"/>
          <s:enumeration value="Invalid"/>
          <s:enumeration value="ConcurrentTx"/>
        </s:restriction>
    </s:simpleType>

    <s:complexType name="IdTagInfo">
      <s:sequence>
        <s:element name="status" type="tns:AuthorizationStatus" minOccurs="1" maxOccurs="1"/>
        <s:element name="expiryDate" type="s:dateTime" minOccurs="0" maxOccurs="1"/>
        <s:element name="parentIdTag" type="tns:IdToken" minOccurs="0" maxOccurs="1"/>
      </s:sequence>
    </s:complexType>
      <!-- End of types shared with ChargePointService -->

    <s:simpleType name="ChargeBoxSerialNumber">
      <s:annotation>
        <s:documentation>String type of max 25 chars that is to be treated as case insensitive.</s:documentation>
      </s:annotation>
      <s:restriction base="s:string">
        <s:maxLength value="25"/>
      </s:restriction>
    </s:simpleType>

    <s:simpleType name="ChargePointModel">
      <s:annotation>
        <s:documentation>String type of max 20 chars that is to be treated as case insensitive.</s:documentation>
      </s:annotation>
```

```
   <s:restriction base="s:string">
     <s:maxLength value="20"/>
   </s:restriction>
 </s:simpleType>


<s:simpleType name="ChargePointSerialNumber">
  <s:annotation>
    <s:documentation>String type of max 25 chars that is to be treated as case insensitive.</s:documentation>
  </s:annotation>
  <s:restriction base="s:string">
    <s:maxLength value="25"/>
  </s:restriction>
</s:simpleType>

<s:simpleType name="ChargePointVendor">
  <s:annotation>
    <s:documentation>String type of max 20 chars that is to be treated as case insensitive.</s:documentation>
  </s:annotation>
  <s:restriction base="s:string">
    <s:maxLength value="20"/>
  </s:restriction>
</s:simpleType>

<s:simpleType name="FirmwareVersion">
  <s:annotation>
    <s:documentation>String type of max 50 chars that is to be treated as case insensitive.</s:documentation>
  </s:annotation>
  <s:restriction base="s:string">
    <s:maxLength value="50"/>
  </s:restriction>
</s:simpleType>

<s:simpleType name="IccidString">
  <s:annotation>
    <s:documentation>String type of max 20 chars that is to be treated as case insensitive.</s:documentation>
  </s:annotation>
  <s:restriction base="s:string">
    <s:maxLength value="20"/>
  </s:restriction>
</s:simpleType>

<s:simpleType name="ImsiString">
  <s:annotation>
    <s:documentation>String type of max 20 chars that is to be treated as case insensitive.</s:documentation>
  </s:annotation>
  <s:restriction base="s:string">
    <s:maxLength value="20"/>
  </s:restriction>
</s:simpleType>

<s:simpleType name="MeterSerialNumber">
  <s:annotation>
    <s:documentation>String type of max 25 chars that is to be treated as case insensitive.</s:documentation>
  </s:annotation>
  <s:restriction base="s:string">
    <s:maxLength value="25"/>
  </s:restriction>
</s:simpleType>

<s:simpleType name="MeterType">
  <s:annotation>
    <s:documentation>String type of max 25 chars that is to be treated as case insensitive.</s:documentation>
  </s:annotation>
  <s:restriction base="s:string">
    <s:maxLength value="25"/>
  </s:restriction>
</s:simpleType>
```

DOCUMENTATION PP11 : Infrastructure réseau



DOCUMENTATION PP12 : iptables

NOM

iptables - outil d'administration pour le filtrage de paquets IPv4 et le NAT

DESCRIPTION

iptables est utilisé pour mettre en place, maintenir et inspecter les tables des règles de filtrage des paquets IP du noyau Linux. Différentes tables peuvent être définies. Chaque table contient plusieurs chaînes prédéfinies et peut aussi contenir des chaînes définies par l'utilisateur.

Chaque chaîne est une liste de règles que peuvent vérifier un ensemble de paquets ; dans ce cas, on dit qu'on cherche à établir une correspondance avec la règle. Chaque règle détermine ce qui doit être fait avec un paquet qui correspond. Cette action est appelée une «cible», qui peut être un saut vers une chaîne définie par l'utilisateur dans la même table.

filter :

C'est la table par défaut (si l'option -t est omise). Elle contient les chaînes prédéfinies **INPUT** (pour les paquets entrants dans la machine), **FORWARD** (pour les paquets routés à travers la machine) et **OUTPUT** (pour les paquets générés localement).

COMMANDES

Ces options précisent une action particulière à accomplir. Une seule option peut être indiquée sur la ligne de commande, sauf indication contraire. Pour tous les noms en version longue des commandes et des options, vous avez le droit d'utiliser un nombre restreint de lettres du moment qu' **iptables** peut identifier chaque commande sans ambiguïté.

-A, --append *chaîne règle*

Ajoute une ou plusieurs règles à la fin de la chaîne sélectionnée. Lorsque les noms source et/ou destination désignent plus d'une adresse, une règle sera ajoutée pour chaque combinaison d'adresses possible.

-D, --delete *chaîne règle*

Les chaînes standards :

FORWARD : chaine désignant les paquets désirant traverser le pare-feu
INPUT : chaine désignant les paquets s'adressant au pare-feu lui-même
OUTPUT : chaine désignant les paquets expédiés par le pare-feu lui-même
PREROUTING : chaine désignant les paquets attendant d'être routés
POSTROUTING : chaine désignant les paquets venant d'être routés

PARAMÈTRES

Les paramètres suivants composent une spécification de règle (quand ils sont utilisés dans les commandes **add**, **delete**, **insert**, **replace** et **append**).

-p, --protocol [!] *protocole*

Protocole de la règle ou du paquet à vérifier. Le protocole spécifié est l'un des suivants : *tcp*, *udp*, *icmp, ftp, ssh* ou *all*, ou bien sous forme d'une valeur numérique, représentant un de ces protocoles ou un protocole différent. Un nom de protocole issu du fichier /etc/protocols est aussi autorisé. Un «!» avant le protocole inverse le test. La valeur zéro est équivalente à *all*. Le protocole *all* correspond à tous les protocoles ; c'est aussi la valeur par défaut lorsque cette option est omise.

-s, --source [!] *adresse*[/*masque*]

Spécification de la source. L'*adresse* peut être un nom de réseau, un nom d'hôte (attention : spécifier un nom à résoudre avec une requête distante de type DNS est vraiment une mauvaise idée), une adresse de réseau IP (avec /masque) ou une simple adresse IP. Le *masque* peut être un masque de réseau ou un nombre entier spécifiant le nombre de bits égaux à 1 dans la partie gauche du masque de réseau (bits de poids fort). Par conséquent, un masque de *24* est équivalent à *255.255.255.0*. Un «!» avant la spécification d'adresse inverse la sélection d'adresse. L'option **--src** est un synonyme de --source.

-d, --destination [!] *adresse*[/*masque*]

Spécification de la destination. Voir la description du paramètre **-s** (source) pour une description détaillée de la syntaxe. L'option **--dst** est un synonyme de --destination.

-j, --jump *cible*

Ceci détermine la cible de la règle ; c'est-à-dire ce qu'il faut faire si le paquet correspond à la règle. La cible peut être une chaîne définie par l'utilisateur (autre que celle dans laquelle se situe cette règle), une des cibles prédéfinies qui décide immédiatement du sort du paquet, ou une extension (voir **EXTENSIONS** ci-dessous). Si cette option est omise dans une règle, la correspondance d'un paquet avec la règle n'aura aucun effet sur le sort du paquet, mais les compteurs seront incrémentés.

-i, --in-interface [!] [*nom*]

Nom de l'interface qui reçoit les paquets (seulement pour les paquets passant par les chaînes **INPUT**, **FORWARD** et **PREROUTING**). Lorsqu'un «!» est utilisé avant le nom d'interface, la sélection est inversée. Si le nom de l'interface se termine par un «+», il désigne toutes les interfaces commençant par ce nom. Si cette option est omise, toutes les interfaces réseau sont désignées.

-o, --out-interface [!] [*nom*]

Nom de l'interface qui envoie les paquets (seulement pour les paquets passant par les chaînes **FORWARD**, **OUTPUT** et **POSTROUTING**). Lorsqu'un «!» est utilisé avant le nom d'interface, la sélection est inversée. Si le nom de l'interface se termine par un «+», il désigne toutes les interfaces commençant par ce nom. Si cette option est omise, toutes les interfaces réseau sont désignées.

CIBLES

Une règle de pare-feu spécifie des critères de correspondance pour un paquet et une cible. Si le paquet correspond, la règle suivante est déterminée par la valeur de la cible, qui peut être une des valeurs spéciales suivantes : *ACCEPT, DROP,…*

ACCEPT signifie que le paquet est autorisé à passer

DROP signifie que le paquet est rejeté ou détruit.

Exemples :

iptables -A FORWARD -i eth0 -o eth1 -p ftp -j ACCEPT

ajoute une règle qui autorise les paquets ftp à traverser la machine s'il rentre par l'interface réseau eth0 et sort par l'interface réseau eth1.

iptables -A INPUT -s 192.168.0.0/24 -i eth0 -j DROP

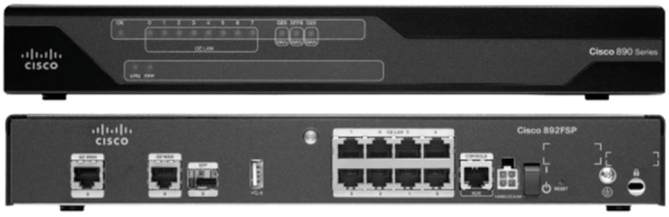
ajoute une règle qui rejette tous les paquets provenant des machines du réseau 192.168.0.0/24 entrants par l'interface réseau eth0 et destinés à cette machine.

iptables -A INPUT -i eth0 -p tcp -j ACCEPT

ajoute une règle qui autorise tous les paquets tcp entrants par l'interface eth0 et destinés à cette machine.

DOCUMENTATION PP13 : CISCO 890 series

Cisco 890 Series Integrated Services Routers



Cisco 890 Series ISRs come with an 8-port managed switch, providing LAN ports to connect multiple devices. An optional Power-over-Ethernet (PoE) capability can also supply power to IP phones and other devices. Eleven Cisco 890 Series models are available: Figure 1 shows the front and back of one, the Cisco 892FSP.

Product Specifications

Table 3 shows Cisco IOS Software features, WLAN features, and general system specifications for the 890 Series ISRs.

Table 3. 890 Series IOS Software Features, WLAN Features, and System Specifications

Feature	Specification
Cisco IOS Software: Advanced IP Features Set (Default)	
IP and IP services	• Routing Information Protocol Versions 1 and 2 (RIPv1 and RIPv2) • Generic Routing Encapsulation (GRE) and Multipoint GRE (MGRE) • Cisco Express Forwarding • Standard 802.1d Spanning Tree Protocol • Layer 2 Tunneling Protocol (L2TP) • Layer 2 Tunneling Protocol Version 3 (L2TPv3) • Network Address Translation (NAT) • Dynamic Host Configuration Protocol (DHCP) server, relay, and client • Dynamic Domain Name System (DNS) • DNS Proxy • DNS Spoofing • Access control Lists (ACLs) • IPv4 and IPv6 Multicast • Open Shortest Path First (OSPF) • Border Gateway Protocol (BGP) • Performance Routing (PfR) • Enhanced Interior Gateway Routing Protocol (EIGRP) • Virtual Route Forwarding (VRF) Lite • Next Hop Resolution Protocol (NHRP) • Bidirectional Forwarding Detection (BFD)
xDSL	• True Multimode VDSL2 and ADSL2+ over Annex A, B, J, and M including traditional G.DMT and T1.413 • World-class interoperability with industry-standard DSL access multiplexer (DSLAM) chipsets • Highest field reliability with Impulse Noise Protection over REIN/SHINE, Extended INP-Delay, G.INP, Physical Layer Retransmission, SRA, and Bitswap • VDSL2 Persistent Storage Device (PSD) profiles up to 17a/b with support for Spectral Shaping • VDSL2 Vectoring to offer blazing fiber speeds over copper • Support for 4-pair multimode G.SHDSL; that is, ATM and EFM • Remote management with TR069 and CWMP • Investment protection with GE and SFP for future fiber that could replace xDSL deployment
Switch features	• Auto Media Device In/Media Device Cross Over (MDI-MDX) • 25 802.1QVLANs • MAC filtering • Four-port 802.3af and Cisco compliant PoE • Switched Port Analyzer (SPAN) • Storm Control • Smart ports • Secure MAC address • Internet Group Management Protocol Version 3 (IGMPv3) snooping • 802.1x

Table 4. Product Part Numbers and Software Images

Product Part Number	Product Description
Integrated Services Routers	
C892FSP-K9	Cisco 892FSP Gigabit Ethernet security router with SFP
C896VA-K9	Cisco 896VA Gigabit Ethernet security router with SFP and VDSL/ADSL2+ Annex B
C897VA-K9	Cisco 897VA Gigabit Ethernet security router with SFP and VDSL/ADSL2+ Annex A
C897VAW-A-K9	Cisco 897VA Gigabit Ethernet security router with SFP and VDSL/ADSL2+ Annex A with Wireless
C897VAW-E-K9	Cisco 897VA Gigabit Ethernet security router with SFP and VDSL/ADSL2+ Annex A with Wireless
C897VA-M-K9	Cisco 897VA Gigabit Ethernet security router with SFP and VDSL/ADSL2+ Annex M
C897VAM-W-E-K9	Cisco 897VA Gigabit Ethernet security router with SFP and VDSL/ADSL2+ Annex M with Wireless
C897VAB-K9	Cisco 897VA Gigabit Ethernet security router with SFP and VDSL2/ADSL2+ Bonding over POTS
C898EA-K9	Cisco 898EA Gigabit Ethernet security router with SFP and 4 channel multimode G.SHDSL (EFM/ATM)
C891F-K9	Cisco 891F Gigabit Ethernet security router with SFP
C891-24X/K9	Cisco 891 Gigabit Ethernet security router with SFP and 24-ports Ethernet Switch
C891FW-A-K9	Cisco 891F Gigabit Ethernet security router with SFP and Dual Radio 802.11n Wifi for FCC -A domain
C891FW-E-K9	Cisco 891F Gigabit Ethernet security router with SFP and Dual Radio 802.11n Wifi for ETSI -E domain
Cisco 892FSP is supported only on Cisco IOS Software Release 15.2(4)M and later Cisco 896, 897, 898EA is supported only on Cisco IOS Software Release 15.2(4)M1 and later Cisco 891F is supported only on Cisco IOS Software Release 15.3(3)M2, 15.4(1)T and later C897VAB is supported only on Cisco IOS Software Release 15.4(3)M1 and later C891-24X is supported only on Cisco IOS Software Release 15.5(1)T and later	

DOCUMENTATION PP14 : std::vector

std::vector

template < class T, class Alloc = allocator<T> > class vector; // generic template

Vectors are sequence containers representing arrays that can change in size.

Just like arrays, vectors use contiguous storage locations for their elements, which means that their elements can also be accessed using offsets on regular pointers to its elements, and just as efficiently as in arrays. But unlike arrays, their size can change dynamically, with their storage being handled automatically by the container.

Internally, vectors use a dynamically allocated array to store their elements. This array may need to be reallocated in order to grow in size when new elements are inserted, which implies allocating a new array and moving all elements to it. This is a relatively expensive task in terms of processing time, and thus, vectors do not reallocate each time an element is added to the container.

operator [ ]

const_reference operator[ ] (size_type **n**) const;

Access element **n** in the vector container.

A similar member function, vector::**at**, has the same behavior as this operator function, except that vector::**at** is bound-checked and signals if the requested position is out of range by throwing an out_of_range exception.

Portable programs should never call this function with an argument n that is out of range, since this causes undefined behavior.

Parameters

n

Position of an element in the container.

Notice that the first element has a position of 0 (not 1).

Member type size_type is an unsigned integral type.

return value

The element at the specified position in the vector.

Example

```
#include <iostream>
#include <vector>
int main ( )
{
std::vector<int> myVector ( 10 );   // 10 non-initialized integers
int tailleVector = myVector.size();
      for ( unsigned indice = 0; indice < tailleVector; indice++ )
            std::cout << '  ' << myVector[ indice ];
      std::cout << '\n';
      return 0;
}
```

DOCUMENTATION PP15 : Schéma entités-relations incomplet de la BDD de la supervision