

SESSION 2023

CAPET
CONCOURS EXTERNE ET CAFEP CORRESPONDANT
ET TROISIEME CONCOURS

Section : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR

Option : INGÉNIERIE INFORMATIQUE

ÉPREUVE ÉCRITE DISCIPLINAIRE

Durée : 5 heures

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

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

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

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

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

INFORMATION AUX CANDIDATS

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

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

► **Concours externe du CAPET de l'enseignement public :**

| Concours | Section/option | Epreuve | Matière |
|----------|----------------|---------|---------|
| EDE | 1413E | 101 | 9311 |

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

| Concours | Section/option | Epreuve | Matière |
|----------|----------------|---------|---------|
| EDF | 1413E | 101 | 9311 |

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

| Concours | Section/option | Epreuve | Matière |
|----------|----------------|---------|---------|
| EDV | 1413E | 101 | 9311 |

COMPOSITION DU SUJET

SUJET :

- Partie 1 : Description générale Page 1
- Partie 2 : Étude du déplacement dans l'environnement Page 3
- Partie 3 : Détection d'obstacles Page 7
- Partie 4 : Étude la collecte et de l'analyse des évènements Page 12
- Partie 5 : Mise à disposition des informations utilisateurs Page 14
- Partie 6 : Conclusion Page 15

DOCUMENTS :

- Dossier technique (DT1 à DT 17) Page 16
- Documents réponses (DR1 à DR 8)
- DR1 à DR4 Page 42
- DR5 et DR6 Page 43
- DR7 et DR8 page 44

Etude d'un moyen de déplacement interne industriel

Partie 1 - Description générale

Objectif : *s'approprier le contexte de l'étude et l'intérêt de la mise en place d'une flotte de minipORTEURS.*

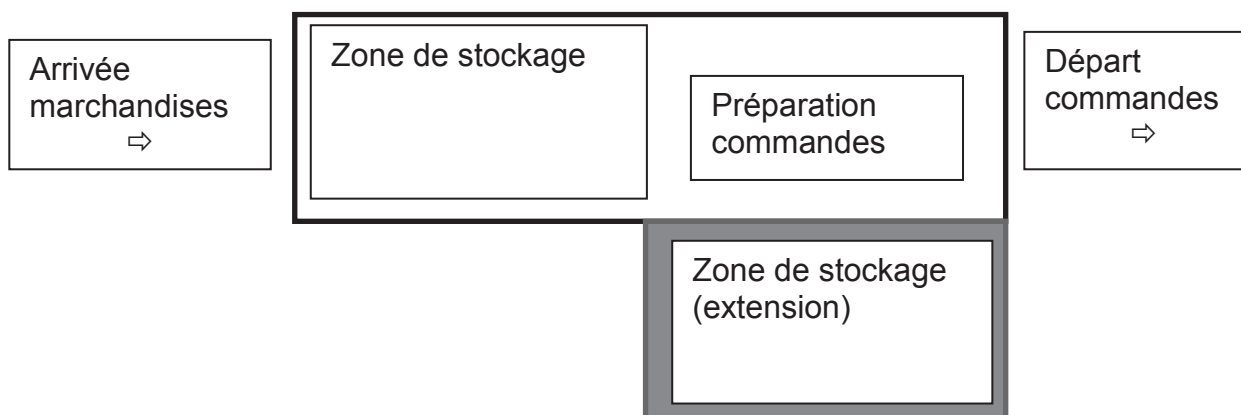
Conditions de travail liées à une activité de stockage

Avec le développement des activités de vente en ligne, les besoins en stockage ont considérablement évolué (plus de 1200 plateformes en France couvrent plus de 20000 m²). Une entreprise de logistique vient d'agrandir ses entrepôts en passant d'une surface de 10000 m² à 15000 m².

Les employés de l'entreprise rencontrent un certain nombre de difficultés dans leurs conditions de travail : temps de déplacements entre les différents espaces pour les chefs d'équipe, pénibilité du transport des charges pour les agents de maintenance et d'inventaire, insécurité des déplacements dans des entrepôts partagés entre les piétons et les véhicules.

L'espace est conçu pour optimiser le flux des marchandises (zonage des produits).

Entrepôt d'origine : 10000 m² (200 x 50)



Extension : 5000 m² (100 x 50)

L'entreprise a mené une campagne interne sur les troubles TMS (les troubles musculosquelettiques représentent 95 % des maladies professionnelles dans le secteur du Transport et de la Logistique et 15 % des accidents du travail sont liés au mal de dos d'après l'Assurance Maladie). Elle souhaite accompagner son extension d'une amélioration des conditions de travail pour les métiers impactés, notamment sur la durée des déplacements et le transport de charges.

Un chef d'équipe parcourt pour ses activités environ 8 km par jour pour couvrir l'entrepôt avant l'agrandissement.

Question 1 : Avec une activité similaire, calculer le temps quotidien nécessaire à cet employé pour couvrir l'entrepôt agrandi. On prendra comme hypothèse une vitesse de marche de $4,5 \text{ km}\cdot\text{h}^{-1}$ pour un piéton.

Une caisse à outils de maintenance peut peser de 10 à 20 kg, et une servante d'inventaire peut rapidement servir à déplacer 100 kg de marchandises. Pour assister au déplacement de ces charges, l'entreprise souhaite arrêter d'utiliser les véhicules type chariot de manutention, surdimensionnés (ils sont prévus pour 1 tonne et l'essentiel de l'énergie sert à leur propre déplacement) et peu maniables pour des déplacements rapides.

Afin améliorer efficacement les conditions de travail, l'entreprise a choisi de se doter de miniporteurs.

Miniporteur Hublex

La société Hublex conçoit des outils de co-botique d'assistance à l'homme. Le miniporteur professionnel Hublex est un outil de déplacement et de transport de petites charges.

Il permet à un employé de se déplacer sans fatigue sur de longues distances, et de transporter avec lui son matériel ou des quantités limitées de marchandises.



L'entreprise en a acquis trois, afin d'assurer une disponibilité permanente entre les différentes personnes amenées à s'en servir, dans un fonctionnement en horaire de travail en 2 x 8.

La présente étude porte sur l'intérêt de l'acquisition de ces miniporteurs dans le cadre d'une amélioration efficace des conditions de travail.



Le document DT 1 détaille le diagramme des exigences partiel du système étudié.

Question 2 : A partir de ce document, quel serait le gain maximal en temps pour le chef d'équipe évoqué ci-dessus ? Donnez l'exigence qui répond au besoin spécifique d'un agent d'inventaire.

Question 3 : Justifier l'exigence 1.3.1 au regard du besoin d'améliorer les conditions de travail avec efficacité.

Question 4 : A l'aide des deux diagrammes de définition de blocs (BDD) du DT2, expliquer sommairement la différence de fonctionnement dans les deux cas exposés.

Partie 2 - Etude du déplacement dans l'environnement

Objectif : déplacement autonome vers la zone de charge.

Afin de minimiser le temps et les efforts pour les utilisateurs, ceux-ci laissent le miniporteur après utilisation dans une zone de dépose identifiée par des marquages au sol (rectangle vert sur la figure 1)



Figure 1 : zone de dépose d'un miniporteur

La zone est reliée à une ligne guide bicolore (rouge et bleue). Un système autonome d'analyse de l'environnement et de guidage ramène alors le miniporteur dans sa zone de parking et de recharge de la batterie.

La caméra OpenMV Cam H7 embarquée sur le miniporteur permet d'acquérir les images nécessaires à l'analyse de l'environnement, à l'aide d'un logiciel en MicroPython qui implémente une version allégée de Python 3.4 et des modules spécifiques à la caméra.

Analyse de l'environnement

Pour identifier le type de zone où il se trouve, le miniporteur utilise un système de repères visuels appelé « AprilTag ». Ce système se compose de cibles carrées à imprimer et d'un logiciel de détection sous licence libre.

Question 5 : Indiquer ce que permet de faire ce type de licence.

Le document DT3 AprilTag présente les principes du système de repérage et le DT4 donne les cibles utilisées par le miniporteur.

Question 6 : Justifier l'intérêt d'utiliser ce système et donner la signification de la cible en photo sur la figure 1

Le module `cam_analysing.py` réalise entre autres la détection des tags April et des blobs (ensemble de pixels connectés par des seuils de couleur, soit ici la ligne guide bicolore). Il effectue notamment les importations suivantes :

```
import sensor
from image import Image, TAG16H5
```

Question 7 : Donner la famille AprilTag utilisée par le programme et ses caractéristiques.

Le miniporteur déposé entame son retour autonome vers la zone de charge grâce au guidage de de ligne. Le prochain tag rencontré est le tag ci-contre.



Question 8 : Sachant qu'un zéro est codé en noir, expliquer le principe d'encodage de l'information dans ce type de tag, et en déduire la valeur hexadécimale que le programme va utiliser pour reconnaître le tag.

Une distance de Hamming d entre deux mots binaires signifie qu'au minimum d bits varient entre les deux mots. Le motif ci-contre entre dans le champ de vision de la caméra.



Question 9 : Expliquer si ce tag sera détecté comme valide.

Le document DT5 présente l'organisation logicielle partielle simplifiée de l'analyse d'image .

Question 10 : Donner le nom du schéma UML présenté dans le document technique DT5, ainsi que le type d'association entre les rectangles « Image » et « AprilTag ».

La méthode `find_apriltags()` est utilisée pour identifier les tags qui se situent dans l'image. Elle retourne une liste d'objets de la classe `apriltag`, identifiés dans la zone d'intérêt de l'image (`roi`) passée en argument. Cette zone est identifiée par un index qui spécifie si c'est la zone gauche (`roi_april_index` vaut 0) ou droite (`roi_april_index` vaut 1) qui est analysée.

Extrait du module `cam_analysing.py` :

```
roi_april_index = 0
tags_on_CAN = []
img = Image(160,120, sensor.RGB565, sensor.data)

def april_tag_analysing():
    """Read april tag"""
    global roi_april_index
    global tags_on_CAN
    tags = img.find_apriltags(roi=roi_april[roi_april_index], families=TAG16H5)
    for tag in tags:
        # Append tag to the CAN buffer
        tags_on_CAN.append([roi_april_index, tag.id(), tag.cy(), tag.cx(),
                           tag.rotation()*180/3.1415])
    if(roi_april_index is 1):
        roi_april_index = 0
    else:
        roi_april_index = 1
```

Question 11 : Expliquer le rôle du mot-clé « global » et indiquer pourquoi il n'est pas appliqué à la variable `img`.

Question 12 : A l'aide du DT5, proposer une implémentation en python du constructeur de la classe `Image`.

Sur chacune des vues ci-dessous, un rectangle vert indique la zone d'intérêt et un rectangle rouge indique un tag April détecté par le programme lors de l'appel à `april_tag_analysing()`.

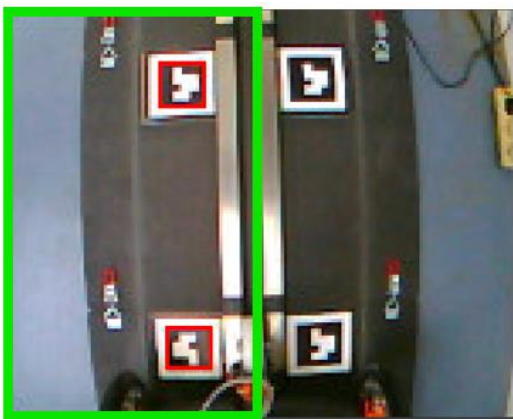
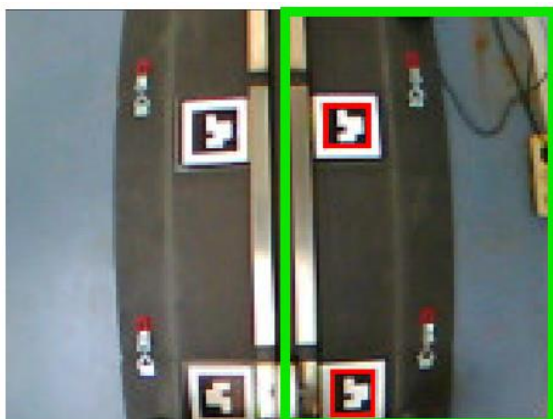


Figure 2 : analyse A



analyse B

Question 13 : Déterminer ce que renvoie `len(tags)` après l'analyse A. Expliquer précisément ce que la variable `tags` contient.

La figure 3 présente une nouvelle image capturée ainsi que l'échelle de mesure associée :

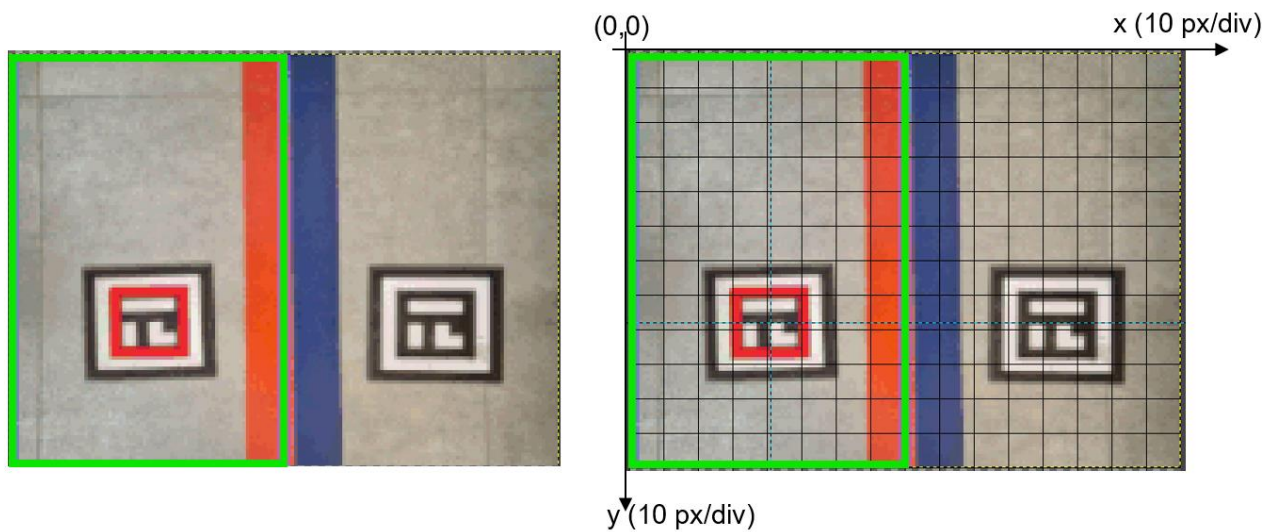


Figure 3 : image analysée et grille de mesure

Question 14 : Dans le cas de la figure 3, donner le contenu de `tags_on_CAN` après l'appel de `april_tag_analysing()`.

On souhaite faire dessiner sur l'image chaque rectangle rouge autour des Tags April détectés.

Question 15 : Proposer une instruction pour l'implémenter et indiquer son emplacement dans le code de `cam_analysing.py`

Dans le code du programme principal, on trouve les lignes suivantes :

```
import cam_analysing
import can

def run():

    cam_analysing.get_image()
    cam_analysing.april_tag_analysing()
    while(len(cam_analysing.tags_on_CAN) > 0):
        tag = cam_analysing.tags_on_CAN[0]
        can.send_tag(tag[0], tag[1], tag[2], tag[3], tag[4])
        ...
```

`cam_analysing.getimage()` récupère une image de la caméra.
`can.send_tag()` envoie un message dans le format prévu sur la liaison CAN entre la carte du guidon et la carte mère du miniporteur.

Question 16 : Expliquer et déterminer la condition de la boucle `while`. Rédiger la dernière ligne de la fonction `run()` sur le DR1.

La fonction `run()` est appelée en boucle par le programme d'acquisition du miniporteur. On redonne les dernières lignes de `april_tag_analysing()` :

```
if(roi_april_index is 1):  
    roi_april_index = 0  
else:  
    roi_april_index = 1
```

Question 17 : Expliquer l'objectif de ces lignes

Question 18 : Conclure sur la réalisation de l'exigence 1.3.1.1

Partie 3 - Détection d'obstacles

Objectif : étudier la solution retenue pour la gestion des obstacles lors d'un retour en autonomie du miniporteur.

Afin de détecter de potentiels obstacles lors d'un déplacement, le miniporteur est muni de deux capteurs VL53L0X, un pointant vers l'avant, l'autre vers le bas.

Etude du capteur

La documentation technique partielle du VL53L0X est donnée dans le DT6.

Question 19 : Donner la technologie employée par ces capteurs et son mode de fonctionnement.

Question 20 : Donner le type de signal lumineux employé. Déterminer sa longueur d'onde. A l'aide du DT7, définir la gamme de lumière dans laquelle il situe et s'il est visible par l'œil humain. Déterminer sa dangerosité pour l'œil humain.

Question 21 : Quels sont les trois types de mémoires présents sur le capteur ? Donner la signification des acronymes. Dans quelle(s) mémoire(s) pourrait être stockée l'adresse du composant ? Pourquoi ?

Mise en œuvre des capteurs

La transmission des données provenant des capteurs vers la carte caméra s'effectue par le protocole I²C.

L'adresse usine d'un capteur est 0x52.

Question 22 : L'adresse usine correspond à une écriture. Déterminer quelle serait alors l'adresse en lecture. Justifier la réponse.

La vitesse de transmission sur le bus I²C utilisé est définie ainsi :
i2c.init(I2C.MASTER, baudrate=400000)

Question 23 : Calculer dans ce cas la durée de transmission d'un bit. Calculer le temps de transmission d'un octet.

La gestion des capteurs s'effectue à l'aide d'un programme développé sous Python nommé « obstacle.py ».
Son header est donné ci-dessous :

```
# obstacle.py
1... import gc
2... from utime import sleep_ms
3... from pyb import I2C, Pin
4... gc.collect()
5... from vl53l0x import VL53L0X
6... from can import offset_distance_front
```

Question 24 : Commenter les lignes n°1, 3 et 4. Expliquer l'intérêt de préciser les classes au lieu de faire import *.

Les capteurs sont connectés au microcontrôleur par l'intermédiaire de connecteurs suivant le schéma du DT8. Le capteur bas est relié au connecteur P5 et le capteur avant au connecteur P4.

Leur utilisation nécessite de préciser la connexion des broches XSHUT et leur adresse de départ afin d'éviter tout problème d'adressage dans le cas de la réutilisation de capteurs d'un autre miniporteur.

Le programme de gestion des capteurs utilise entre autres la classe VL53L0X donnée partiellement en DT9.

Question 25 : D'après la définition de la classe VL53L0X, définir l'adresse d'initialisation par défaut des capteurs si celle-ci n'est pas précisée spécifiquement.

Question 26 : Sur le document DR2, compléter les informations manquantes :

- afin d'expliciter les broches utilisées pour le signal XSHUT de chaque capteur ;
- afin d'initialiser les deux capteurs à une même adresse connue, l'adresse dite « usine ».

Le miniporteur utilisant deux capteurs VL53L0X, il est nécessaire d'attribuer une adresse différente à chacun d'eux.

Pour ce faire, il faut désactiver un des capteurs afin d'initialiser l'autre. Cette désactivation s'effectue à l'aide de la broche XSHUT qui permet d'éteindre complètement le VL53L0X.

Le capteur bas est le premier à être initialisé selon la procédure suivante :

```
XSHUT_DOWN.value(0)
sleep_ms(10)
XSHUT_DOWN.value(1)
```

Question 27 : A l'aide du DT10, expliciter chacune des 3 lignes précédentes.

Question 28 : Donner la commande permettant de désactiver le capteur avant.

On souhaite modifier l'adresse du capteur bas en 0x30.
Le registre (index) d'adressage du capteur est le suivant :
I2C_SLAVE_DEVICE_ADDRESS = 0x8A

Question 29 : A l'aide du DT9 et des initialisations présentes sur le DR2, donner la commande permettant cette modification. Décrire les différents éléments de cette commande.

Question 30 : A l'aide du DT6, déterminer les commandes en hexadécimales à envoyer sur la liaison I²C du capteur bas.

Question 31 : Compléter le document DR3 en nommant les deux signaux et en représentant la trame de modification d'adresse du capteur bas.

Transmission des informations

Les informations récupérées par le microcontrôleur sont ensuite transmises par bus CAN à une carte intermédiaire gérant notamment le servomoteur pour la direction, puis en CANOpen jusqu'à la carte mère gérant les moteurs du miniporteur.

Question 32 : Le bus CAN comporte deux signaux, CANH et CANL, dont la lecture s'effectue en mode différentiel et la transmission par paire torsadée. Expliquer l'intérêt de ces technologies.

Question 33 : A l'aide du DT11, donner la taille d'un identificateur en mode standard ainsi que le nombre maximum de valeurs d'identificateurs utilisables.

Question 34 : Déterminer la taille maximum que peut prendre le champ de données.

Différents nœuds peuvent accéder simultanément au bus, il y a donc nécessité d'un arbitrage.

L'arbitrage consiste à déterminer la priorité d'une trame par rapport à une autre. Le champ pendant lequel s'effectue l'arbitrage est constitué des bits de l'identifiant ainsi que du bit RTR. Pendant le champ d'arbitrage, l'interface CAN compare les bits transmis et reçus. Le nœud qui émet un bit dominant ('0') gagne l'arbitrage face à un bit récessif ('1') et continue à émettre alors que l'autre nœud s'arrête.

Question 35 : Etant donné le fonctionnement du bit RTR, quelle trame sera prioritaire entre une trame de requête et une trame de données ?

Les trois identifiants suivants sont initialisés pour la communication par bus CAN des informations provenant de la caméra :

```
#IDs
_COMMANDS_CANID = const(0x606)
_DATA1_CANID = const(0x186)
_TAG_CANID = const(0x286)
```

Le tableau ci-dessous présente les données transférées.

| Id | Nom | DLC | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 |
|-------|-----------|-----|-------|------|------|------------|------|-----------|----|--------|
| 0x186 | TPDO1 | 8 | flags | line | led | dist front | | dist down | | status |
| 0x286 | Data tags | 7 | ROI | id | cy | cx | qual | rotation | | |
| 0x606 | Commands | 3 | flags | led1 | led2 | | | | | |

Question 36 : Ces 3 nœuds émettent une trame de données au même moment. Compléter le DR4 en traçant la trame résultante finalement présente sur le bus. Quelle trame est prioritaire ? Cela semble-t-il cohérent ?

Question 37 : Commenter la valeur du champ d'arbitrage de la trame prioritaire par rapport aux deux autres. En déduire comment les transmissions peuvent être priorisées.

Le bus CAN étant une liaison asynchrone, il faut qu'un front soit régulièrement présent pour resynchroniser le récepteur.

L'ajout systématique d'un bit de niveau logique inverse au bout de 5 bits de même niveau permet de résoudre le problème.

L'émetteur ajoute donc ces bits dits de « stuffing », le récepteur ne les considèrera pas comme de l'information utile.

Question 38 : Expliciter le terme asynchrone.

Question 39 : Le récepteur se synchronisant sur les fronts descendants, déterminer le nombre maximum de bits entre chaque resynchronisation.

Les contrôleurs moteur Hublex disposent d'une interface CAN implémentant le protocole CANOpen présenté succinctement dans le DT12.

Pour une commande moteur, le CANOpen id est 0x2000.

Cette commande change de signification selon le mode de contrôle actif au moment de la réception.

Les éléments de la trame sont les suivants :

Sous-index : *Non applicable*

Argument selon le mode actif :

- *Mode= Boucle Ouverte : commande en tension +/-100% Ubus, soit [-32768 ; +32767] sur 16 bits signés ;*
- *Mode = Vitesse : consigne vitesse en rpm [-200 ; +200] sur 16 bits signés ;*
- *Mode = Position : consigne position encodeur sur 32 bits signés ;*
- *Mode = Courant : consigne courant (couple) +/-13N.m ⇔ [-23920 ; +23920] sur 16 bits signés ;*

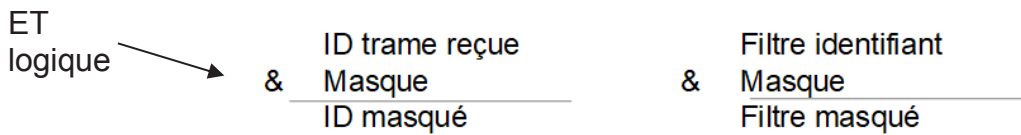
Réponse : *Aucune*

Lorsque qu'un des capteurs VL53L0X envoie une valeur indiquant une distance inférieure à 50 cm, une commande de freinage est envoyée. Elle correspond à une commande « reverse » pendant 150 ms puis rien pendant 100 ms.

Le mode « reverse » est l'envoi d'une consigne en vitesse de - 100 rpm sur un message SDO de commande.

Question 40 : Compléter la trame du DR5 pour une telle consigne sur le nœud 6.

Chaque message possède un identificateur qui n'indique pas la destination du message mais la signification des données du message. Ainsi tous les nœuds reçoivent le message, et chacun est capable de savoir, grâce à un système de filtrage et de masquage, si ce dernier lui est destiné ou non.



Lors de la réception d'une trame, si l'ID (header) masqué est identique au filtre masqué, le contrôleur accepte la trame.

Exemple : Filtre identifiant : 0x2CA = 010 1100 1010₍₂₎
 & Masque : 0x7CF = 111 1100 1111₍₂₎
 Filtre masqué : 0x2CA = 010 1100 1010₍₂₎

Les trames acceptées seront 0x2CA, 0x2DA, 0x2EA et 0x2FA.

Dans le cas du miniporteur, le filtre masqué pour le nœud 6 est 0x006.

Les 3 trames rattachées au nœud 6 sont celles présentées précédemment : 0x186, 0x286, 0x606.

Question 41 : Déterminer si le masque 0x00F permet l'acceptation de toutes ces trames.

Déterminer quelle partie du masque permet de s'assurer que le nœud 6 n'accepte que les trames lui étant destinées mais pas celles destinées aux autres nœuds.

Les trames de données CAN et les trames distantes contiennent une protection basée sur un polynôme CRC : l'émetteur calcule une somme de contrôle à partir des bits transmis et fournit le résultat dans la trame dans le champ CRC. Les récepteurs utilisent le même polynôme pour calculer la somme de contrôle à partir des bits vus sur les lignes de bus. La somme de contrôle auto-calculée est comparée à la somme reçue. Si elle correspond, la trame est considérée comme correctement reçue et le nœud récepteur transmet un état dominant dans le bit du slot ACK, écrasant l'état récessif de l'émetteur. En cas de non-concordance, le nœud récepteur envoie une trame d'erreur après le délimiteur ACK. Le document DT 13 présente la manière dont est calculé le CRC.

Question 42 : Recopier puis compléter l'algorithme suivant permettant ce calcul :

```
CRC_RG = 0; // initialize shift register
REPEAT
CRCNXT = NXTBIT EXOR CRC_RG(14);
..... // shift left by
..... // 1 position
IF CRCNXT THEN
.....
ENDIF
UNTIL (CRC SEQUENCE starts or there is an ERROR condition)
```

Question 43 : Conclure sur la réalisation de l'exigence 1.3.1.2 concernant la détection et l'évitement des obstacles.

Partie 4 - Etude de la collecte et de l'analyse des événements

Objectif : *Traitement des événements en prévention des pannes et accidents.*

Analyse de la solution existante

Les miniporteurs enregistrent lors de leur fonctionnement un certain nombre d'évènements, qui sont utilisés à titre de supervision et de maintenance.

Ces données sont stockées dans des fichiers de type csv, comme par exemple la liste des alarmes déclenchées sur un miniporteur pendant une période d'utilisation.

Ces fichiers sont ensuite récoltés sur chaque équipement pour une visualisation et un traitement des données via un tableur.

Question 44 : Expliquer ce que sont ces fichiers : donner leur structure et leur usage en général.

Passage à une base de données

Le fournisseur de miniporteurs propose un service de maintenance basé sur ces données, et souhaite passer à une structure de stockage sous forme de base de données relationnelle des informations consolidées de tous les miniporteurs, et d'une gestion de cette base.

Question 45 : Expliquer les avantages à passer dans un modèle de type base de données relationnelle avec un système de gestion associé.

La figure 4 ci-dessous donne le modèle association-entité simplifié de la base. En particulier, un trajet est défini par une utilisation sans interruption d'un miniporteur.

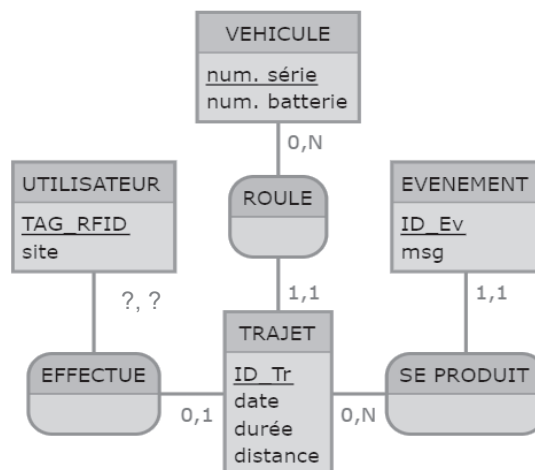


Figure 4 : modèle association-entité

Question 46 : Expliquer ce que signifie l'association EFFECTUE, et justifier la cardinalité (0,1) proche de l'entité trajet.

Question 47 : Proposer et justifier une cardinalité proche de l'entité utilisateur.

La translation du modèle conceptuel en modèle relationnel est donnée partiellement ci-dessous :

VEHICULE (num. série : int, num. batterie : int)
UTILISATEUR (TAG_RFID : string, site : string)
EVENEMENT (ID_Ev : int, msg : string, #ID_Tr : string)

Question 48 : Expliquer ce que représente ID_Tr dans la relation Evenement.

Question 49 : Rédiger la dernière relation de la translation du modèle conceptuel en modèle relationnel.

La base de données est une base interrogeable en langage SQL. Pour analyser l'utilisation des miniporteurs, on souhaite extraire un certain nombre de statistiques.

Question 50 : Donner l'instruction SQL pour afficher la distance moyenne des trajets sur l'ensemble des résultats.

Le résultat que cette requête donne sur tous les enregistrements de la base de données est de 270 m.

On souhaite savoir si certains trajets n'ont pas pu être effectués en raison de problèmes matériels.

Question 51 : Proposer une requête qui liste les numéros et date des trajets qui n'ont pas abouti à un déplacement.

Question 52 : Donner le résultat de cette requête à partir des tables du DT14.

Question 53 : Proposer une requête qui liste les messages et les RFID des Miniporteurs associés aux trajets qui n'ont pas abouti à un déplacement, en utilisant une jointure.

Question 54 : Expliquer les conditions nécessaires pour pouvoir supprimer un enregistrement de la table utilisateur sans provoquer un problème d'intégrité.

Question 55 : Conclure sur l'intérêt de collecter les informations des trajets dans un objectif de maintenance

Partie 5 - Mise à disposition des informations utilisateurs

Objectif : permettre l'accès aux données et à la visualisation des événements d'un site pour un client.

Etude du serveur web

La solution initiale avec des fichiers CSV est exploitée par le fournisseur Hublex au travers d'une plateforme de gestion de site web no code, Google Sites.

Le document DT15 est une capture de l'outil Wireshark, qui est un analyseur de réseau. Il représente une sélection des trames capturées sur l'interface réseau de la machine client, lorsque la société de stockage se connecte sur le site dédié mis à sa disposition par le fournisseur avec l'url : <https://sites.google.com/hublex.fr/client-societe-stockage>

Question 56 : Expliquer ce que représentent les trames 1 et 2 et détailler le fonctionnement de ce protocole.

Les trames 3 à 11 montrent la synchronisation au niveau TCP puis l'initialisation de la session TLS et le protocole « poignée de main » entre le client et le serveur. Le client envoie sa requête pour la page web à la trame 12.

Question 57 : Expliquer pourquoi on ne peut pas lire en clair le mot clé de la requête GET dans cette trame.

Question 58 : Etablir le modèle de réseau en couches OSI et y situer les protocoles mis en œuvre dans le DT 15.

Question 59 : Compléter le diagramme de séquence du DR6 en faisant l'hypothèse que le serveur répond favorablement à la demande de page web du client.

La future évolution pourra s'intégrer dans un serveur sous la maîtrise complète du fournisseur de maintenance du miniporteur.

Les données récoltées sont transférées via le réseau internet sur un serveur qui les stocke dans la base de données. Celle-ci est exploitée au travers d'une architecture logicielle de type LAMP (Linux, Apache, MySQL, PHP) pour permettre l'accès au site web pour le client.

Il peut :

- visualiser les rapports de suivi d'utilisation (durée d'utilisation par véhicule, plages horaires d'utilisation, taux d'alarme batterie faible...)
- soumettre une requête au support ;
- demander un devis.

Pour assurer la confidentialité des données, le client doit s'identifier avant d'accéder à la page des rapports de suivi d'utilisation.

Le code simplifié de la page `index.html` qui contient un formulaire de connexion, est présenté dans le DT16.

Question 60 : Expliquer le rôle des trois fichiers auxquels fait appel cette page et indiquer leur emplacement d'exécution (client ou serveur).

Le DR7 donne le contenu simplifié du fichier script.js qui s'assure de la saisie de tous les champs par l'utilisateur.

Question 61 : Compléter les lignes 4, 6 et 20 du code du DR7 et justifier comment l'un des cas du switch permet la soumission du formulaire.

Le DT17 donne le code partiel du fichier rapports.php. Ce fichier est exécuté lorsque le client s'est correctement identifié. Il utilise la classe PDO pour se connecter à la base de données, en instanciant un objet PDO.

Question 62 : Expliquer le rôle du bloc entre les lignes 10 et 16.

Question 63 : Décrire ce que fait la ligne 21 (echo"<h2> durée des trajets dans la base".DBNAME." </h2>";).

La méthode fetchAll() a été paramétrée pour retourner les résultats de la requête sous la forme d'un tableau de tableau associatif (ou encore dictionnaire).

Question 64 : Compléter la boucle du DR 8 qui affiche le numéro du trajet sur une première ligne, puis la durée (en italique) et la date, sur une seconde ligne.

Question 65 : Conclure sur l'intérêt de pouvoir suivre les informations de type durée d'utilisation pour la société de stockage

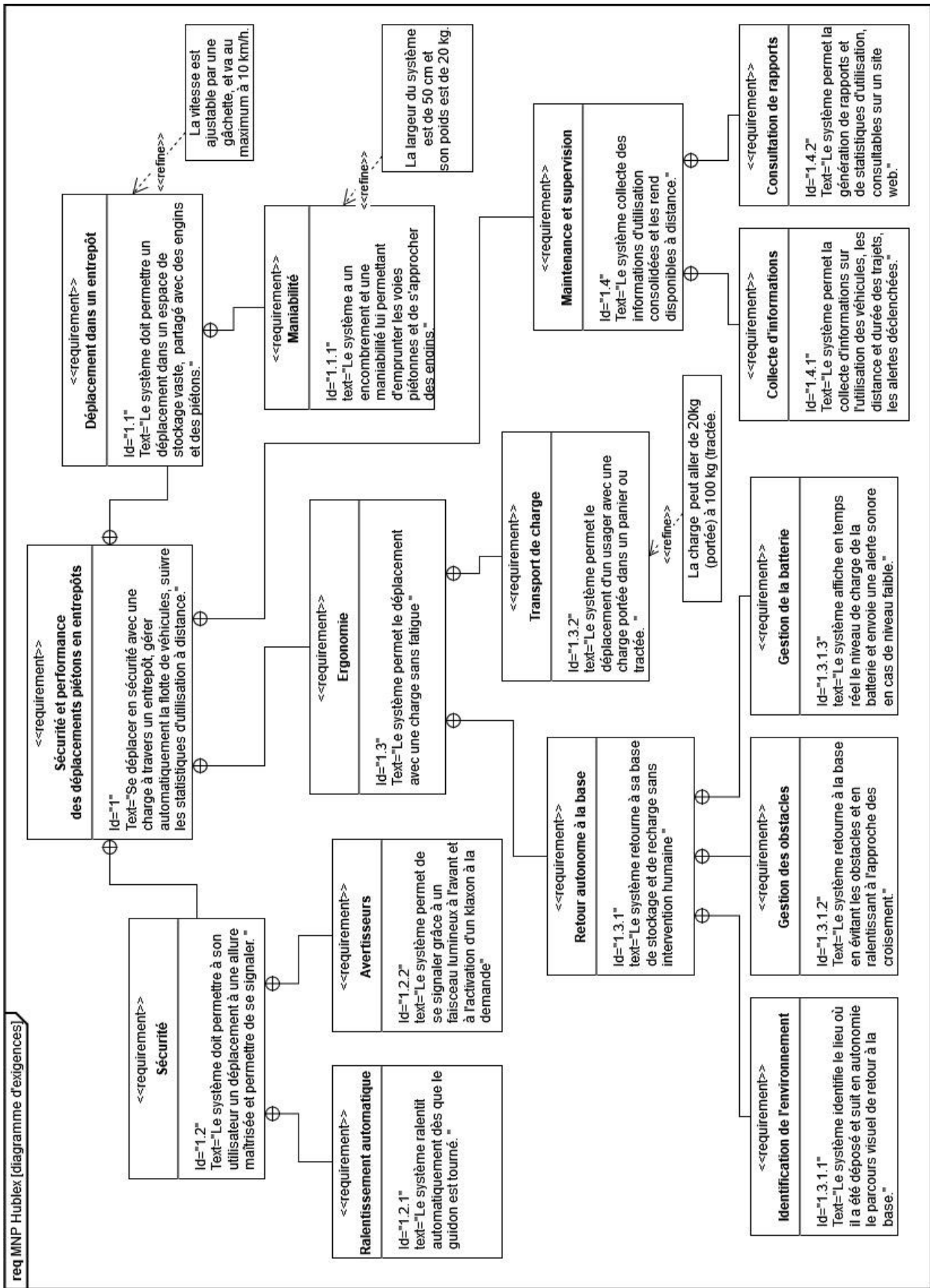
Partie 6 - Conclusion

Question 66 : Conclure sur l'utilité et la gestion d'une flotte de miniporteurs au sein d'un environnement de type entrepôt.

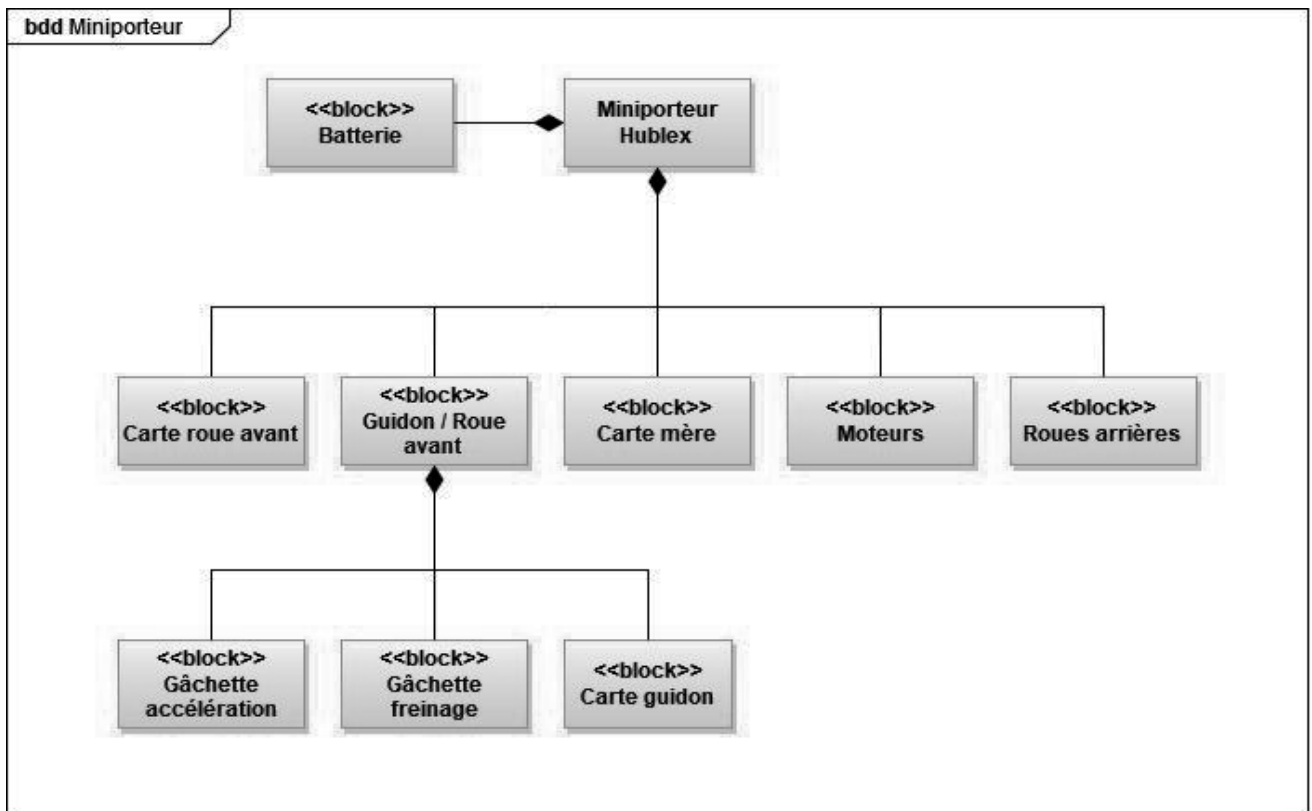
DOSSIER TECHNIQUE

- DT 1 : Diagramme d'exigences partiel
- DT 2 : BDD
- DT 3 : AprilTag
- DT 4 : Tags April reconnus par le miniporteur
- DT 5 : Organisation logicielle partielle simplifiée de l'analyse d'image
- DT 6 : Documentation partielle du capteur VL53L0X
- DT 7 : Longueurs d'ondes et rayonnement Laser
- DT 8 : Shield VL53L0X
- DT 9 : Fichier partiel de la classe VL53L0X
- DT 10 : Extrait de la documentation de la classe PIN
- DT 11 : Format d'une trame CAN
- DT 12 : Documentation CANopen
- DT 13 : Calcul du CRC
- DT 14 : Extraits de tables intermédiaires
- DT 15 : Capture Wireshark
- DT 16 : Fichier simplifié index.html
- DT 17 : Fichier partiel rapports.php

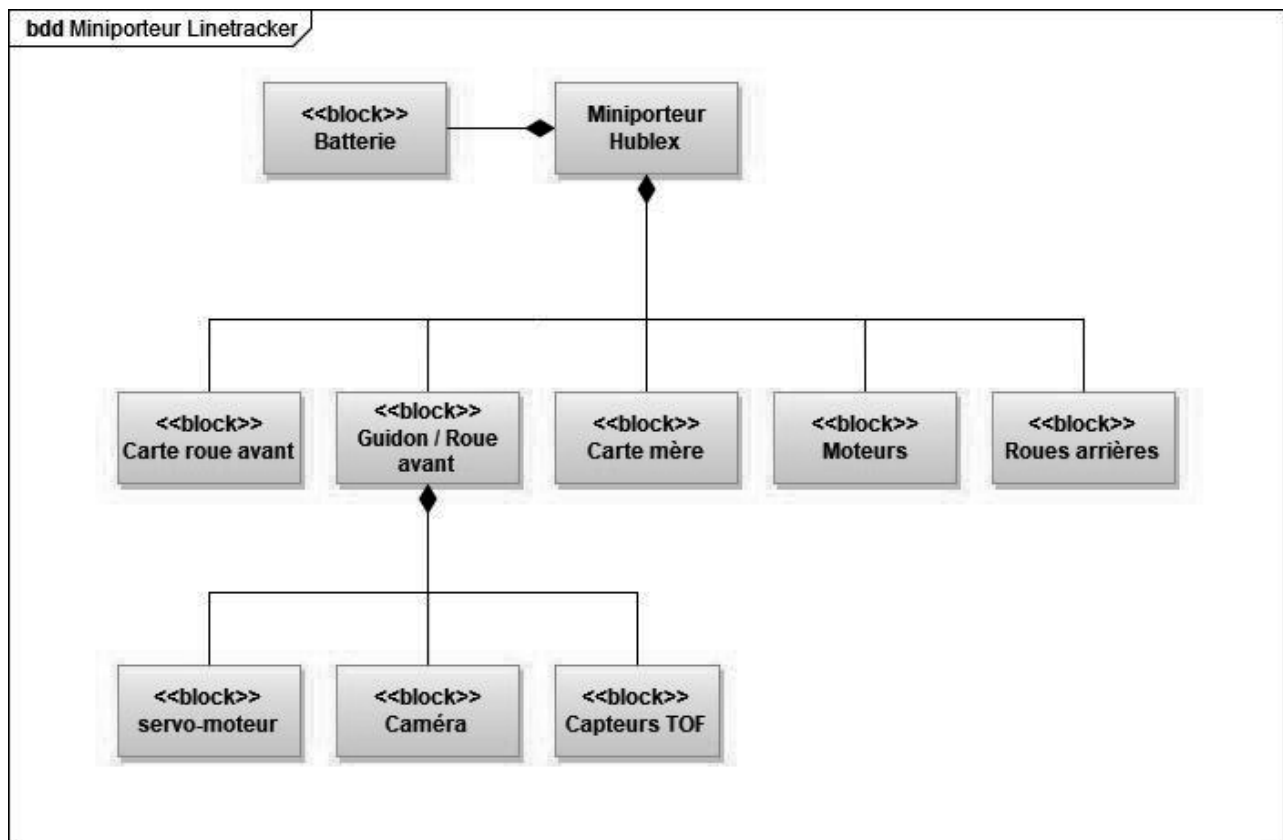
DT 1 : Diagramme d'exigences partiel



DT 2 : BDD



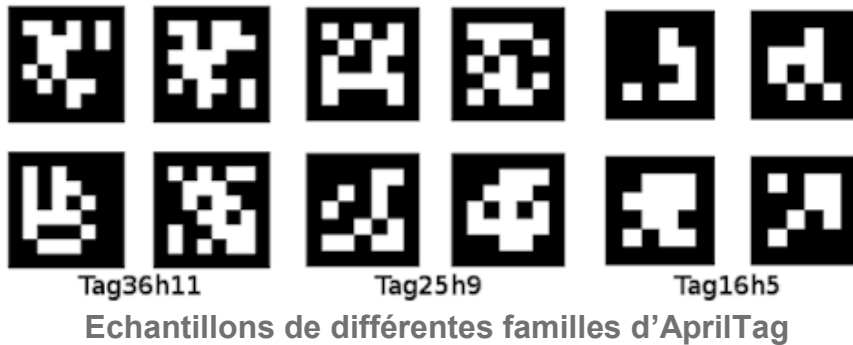
Fonctionnement avec utilisateur



Fonctionnement en autonomie

DT 3 : AprilTag

AprilTag est un système de repères de balises visuels, utile pour une grande variété de tâches, notamment la réalité augmentée, la robotique et l'étalonnage de caméras.



Les cibles peuvent être créées à partir d'une imprimante ordinaire et le logiciel de détection AprilTag calcule la position 3D précise, l'orientation et l'identité des balises par rapport à la caméra. Il est conçu pour être facilement inclus dans d'autres applications, ainsi que pour être portable sur des appareils embarqués. Des performances en temps réel peuvent être obtenues même sur des processeurs de qualité téléphone portable.

La conception des repères et le système de codage sont basés sur un système de codage lexicographique presque optimal, et le logiciel de détection est robuste aux conditions d'éclairage et à l'angle de vue. Les cibles AprilTag sont conceptuellement similaires aux QR codes, en ce sens qu'ils sont un type de code à barres bidimensionnel. Cependant, ils sont conçus pour coder des charges utiles de données beaucoup plus petites (entre 4 et 12 bits), ce qui leur permet d'être détectés de manière plus robuste et à partir de portées plus longues. De plus, ils sont conçus pour une précision de localisation élevée : on peut calculer la position 3D précise de l'AprilTag par rapport à la caméra.

(Licence BSD - extrait de « APRIL Robotics Laboratory » - University of Michigan)

Famille

Une famille d'AprilTag se définit par le nombre de bits utilisés pour l'encodage et la distance de Hamming utilisée. Ainsi, TAG36h11 signifie que 36 bits (6x6) sont utilisés pour coder et que la distance de Hamming vaut 11. La zone codée est entourée d'un carré noir.

Distance de Hamming

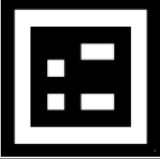
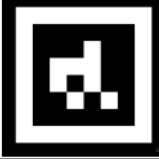


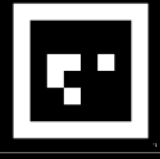
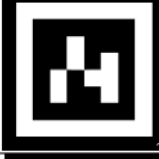



La distance de Hamming permet ici de quantifier la différence entre deux images codées, pour éviter de les confondre en cas d'erreur de codage ou de reconnaissance (une case noire devenant blanche ou vice-versa, en raison de la luminosité, de salissures...). Certaines images parmi toutes les possibilités ne sont ainsi pas disponibles dans une famille donnée.

Tableau des familles avec le nombre d'images réellement utilisables :

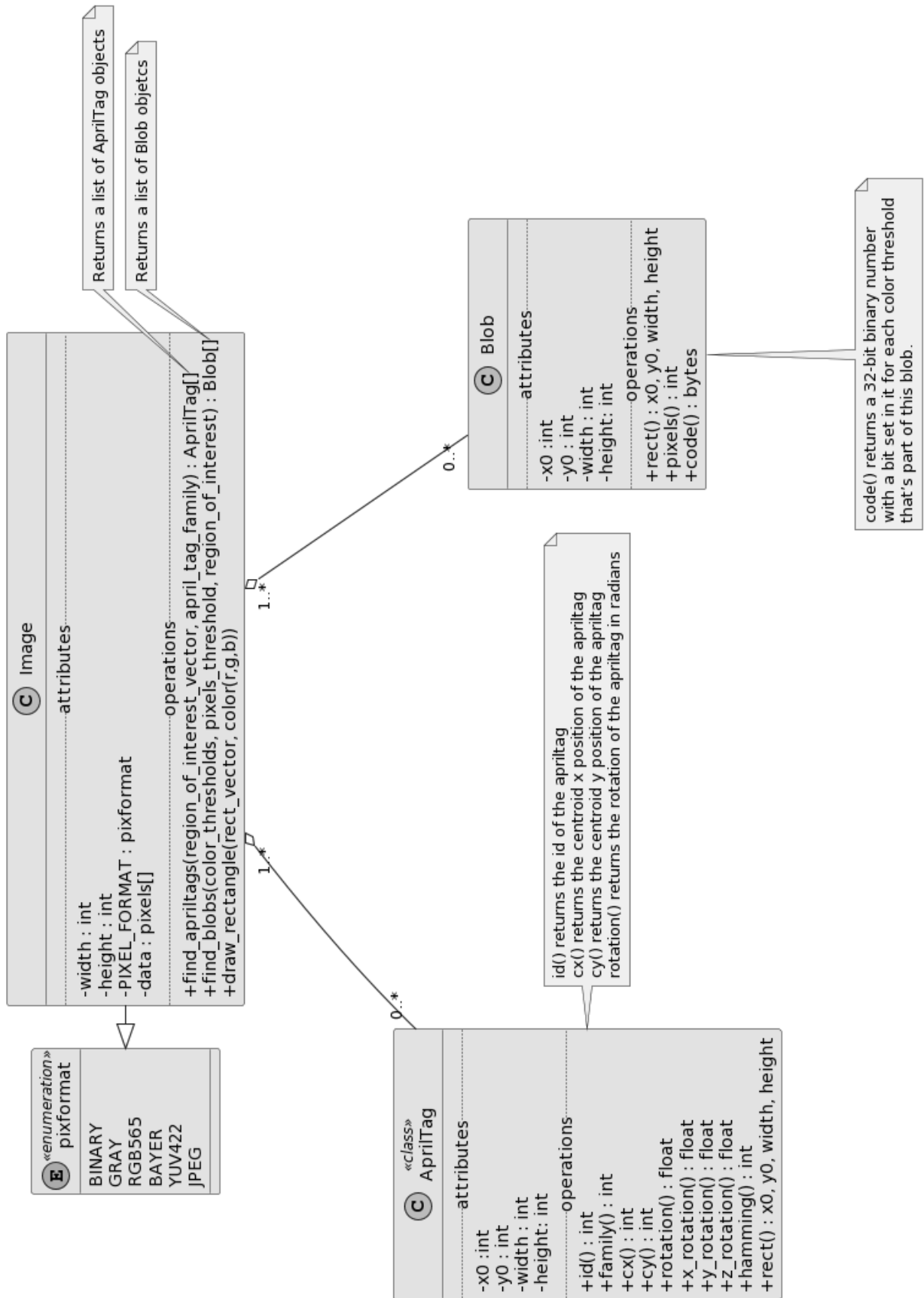
| Famille | Identifiants des cibles disponibles |
|----------|-------------------------------------|
| TAG16H5 | De 0 de à 29 |
| TAG25H7 | De 0 de à 241 |
| TAG25H9 | De 0 à 34 |
| TAG36H10 | De 0 à 2319 |
| TAG36H11 | De 0 à 586 |

DT 4 : Tags April reconnus par le miniporteur

Note : deux cadres, 1 noir et 1 blanc, ont été ajoutés aux cibles de base imprimées pour le miniporteur.

| | | | |
|---|--|--|----------------------------|
|  | 10 Débuter la recherche de ligne et suivre la ligne bicolore trouvée |  | 1 Fixer la vitesse à 2km/h |
|  | 11 Entrée dans la zone de charge |  | 2 Fixer la vitesse à 3km/h |
|  | 12 Fin de zone de charge |  | 3 Fixer la vitesse à 4km/h |
|  | 13 Emplacement de charge |  | 4 Fixer la vitesse à 5km/h |
|  | 0 Fixer la vitesse à 1km/h | | |

DT 5 : Organisation logicielle partielle simplifiée de l'analyse d'image





VL53L0X

World's smallest time-of-flight (ToF) ranging sensor



The VL53L0X is the second-generation laser-ranging sensor based on ST's patented FlightSense™ technology

The VL53L0X is the smallest time-of-flight (ToF) sensor on the market today. It is fully integrated and embeds an infrared, eye-safe laser, advanced filters and an ultra-fast photon detection array. It enhances the ST FlightSense™ product family by enabling measurements of much longer distances with a fast, accurate and robust solution, opening the door to new applications.

KEY BENEFITS

- Absolute distance provided (in mm) up to 2 meters in less than 30 ms
- Fast mode: quick ranging operation at 50 Hz
- High accuracy
- Low power
- 4.4 x 2.4 x 1 mm reflowable package
- Advanced ambient light rejection
- 940 nm invisible light emission
- Works with cover glass

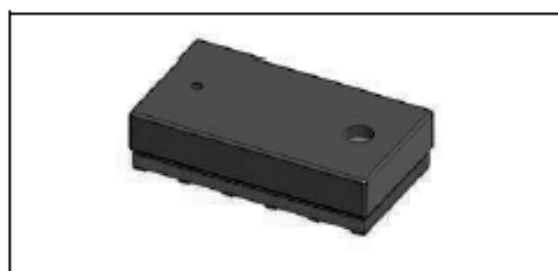
TARGETED APPLICATIONS

- Camera assist (ultra-fast autofocus and depth map)
- User detection for power saving in smartphones or laptops
- Gesture control
- Drones
- Robotic and industrial control
- IoT
- Domestic appliances



World's smallest Time-of-Flight ranging and gesture detection sensor

Datasheet - production data

**Features**

- Fully integrated miniature module
 - 940 nm laser VCSEL
 - VCSEL driver
 - Ranging sensor with advanced embedded micro controller
 - 4.4 x 2.4 x 1.0 mm
- Fast, accurate distance ranging
 - Measures absolute range up to 2 m
 - Reported range is independent of the target reflectance
 - Advanced embedded optical cross-talk compensation to simplify cover glass selection
- Eye safe
 - Class 1 laser device compliant with latest standard IEC 60825-1:2014 - 3rd edition
- Easy integration
 - Single reflowable component
 - No additional optics
 - Single power supply
 - I2C interface for device control and data transfer
 - Xshutdown (reset) and interrupt GPIO
 - Programmable I2C address

Applications

- User detection for personal computers/ laptops/tablets and IoT (energy saving)
- Robotics (obstacle detection)
- White goods (hand detection in automatic faucets, soap dispensers etc.)
- 1D gesture recognition.
- Laser assisted autofocus. Enhances and speeds up camera autofocus system performance, especially in difficult scenes (low light levels, low contrast) or fast moving video mode.

Description

The VL53L0X is a new generation Time-of-Flight (ToF) laser-ranging module housed in the smallest package on the market today, providing accurate distance measurement whatever the target reflectances unlike conventional technologies. It can measure absolute distances up to 2m, setting a new benchmark in ranging performance levels, opening the door to various new applications.

The VL53L0X integrates a leading-edge SPAD array (Single Photon Avalanche Diodes) and embeds ST's second generation FlightSense™ patented technology.

The VL53L0X's 940 nm VCSEL emitter (Vertical Cavity Surface-Emitting Laser), is totally invisible to the human eye, coupled with internal physical infrared filters, it enables longer ranging distances, higher immunity to ambient light, and better robustness to cover glass optical crosstalk.

1 Overview

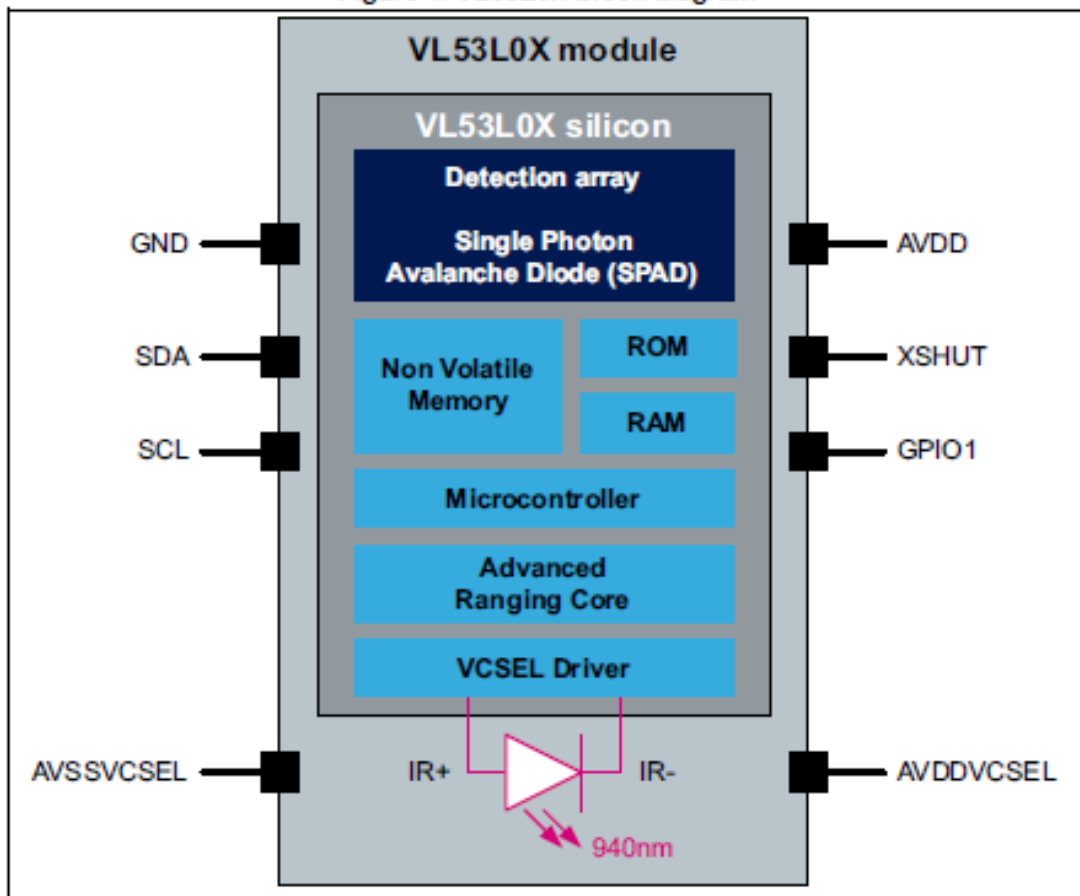
1.1 Technical specification

Table 1. Technical specification

| Feature | Detail |
|------------------------|---|
| Package | Optical LGA12 |
| Size | 4.40 x 2.40 x 1.00 mm |
| Operating voltage | 2.8 to 3.5 V |
| Operating temperature: | -20 to 70°C |
| Infrared emitter | 940 nm |
| I ² C | Up to 400 kHz (FAST mode) serial bus Address: 0x52 |

1.2 System block diagram

Figure 1. VL53L0X block diagram



1.3 Device pinout

Figure 2 shows the pinout of the VL53L0X (see also Figure 22).

Figure 2. VL53L0X pinout (bottom view)

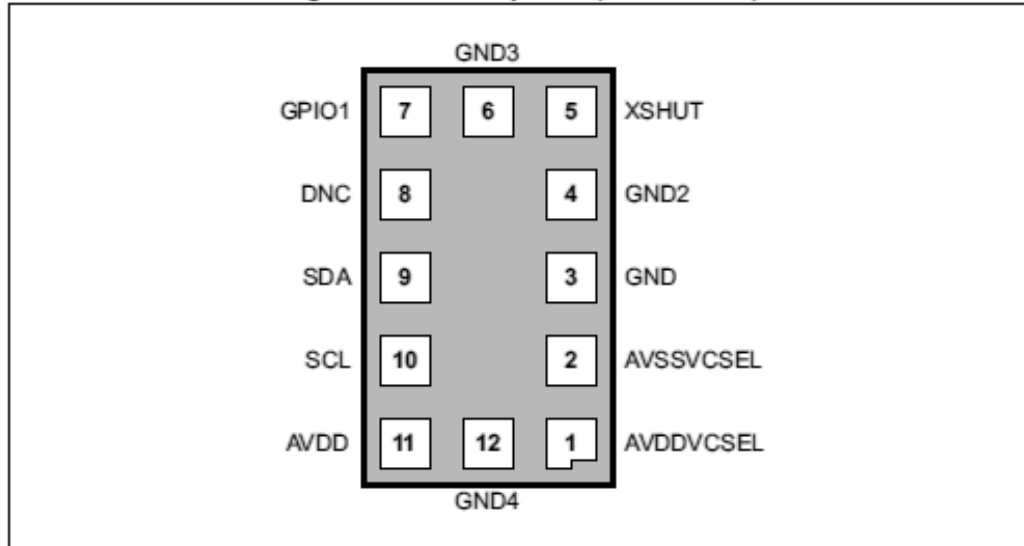


Table 2. VL53L0X pin description

| Pin number | Signal name | Signal type | Signal description |
|------------|-------------|----------------------|--|
| 1 | AVDDVCSEL | Supply | VCSEL Supply, to be connected to main supply |
| 2 | AVSSVCSEL | Ground | VCSEL Ground, to be connected to main ground |
| 3 | GND | Ground | To be connected to main ground |
| 4 | GND2 | Ground | To be connected to main ground |
| 5 | XSHUT | Digital input | Xshutdown pin, Active LOW |
| 6 | GND3 | Ground | To be connected to main ground |
| 7 | GPIO1 | Digital output | Interrupt output. Open drain output. |
| 8 | DNC | Digital input | Do Not Connect, must be left floating. |
| 9 | SDA | Digital input/output | I ² C serial data |
| 10 | SCL | Digital input | I ² C serial clock input |
| 11 | AVDD | Supply | Supply, to be connected to main supply |
| 12 | GND4 | Ground | To be connected to main ground |

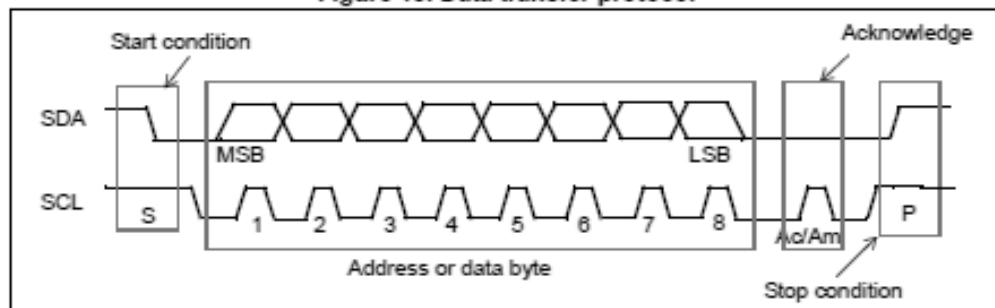
3 Control interface

This section specifies the control interface. The I²C interface uses two signals: serial data line (SDA) and serial clock line (SCL). Each device connected to the bus is using a unique address and a simple master / slave relationships exists.

Both SDA and SCL lines are connected to a positive supply voltage using pull-up resistors located on the host. Lines are only actively driven low. A high condition occurs when lines are floating and the pull-up resistors pull lines up. When no data is transmitted both lines are high.

Clock signal (SCL) generation is performed by the master device. The master device initiates data transfer. The I²C bus on the VL53L0X has a maximum speed of 400 kbits/s and uses a device address of 0x52.

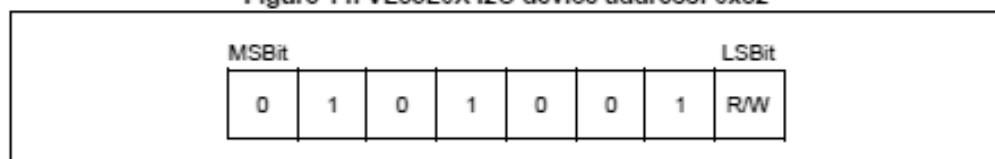
Figure 13. Data transfer protocol



Information is packed in 8-bit packets (bytes) always followed by an acknowledge bit, Ac for VL53L0X acknowledge and Am for master acknowledge (host bus master). The internal data is produced by sampling SDA at a rising edge of SCL. The external data must be stable during the high period of SCL. The exceptions to this are start (S) or stop (P) conditions when SDA falls or rises respectively, while SCL is high.

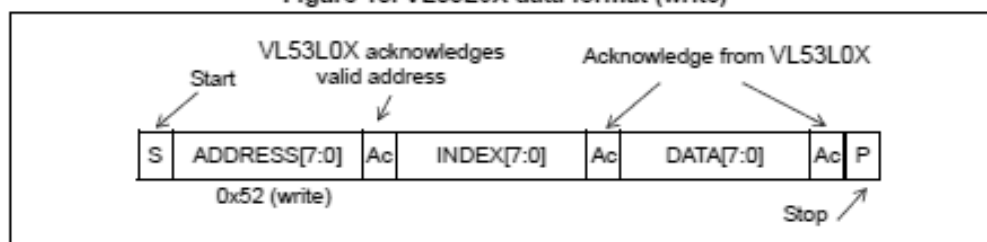
A message contains a series of bytes preceded by a start condition and followed by either a stop or repeated start (another start condition but without a preceding stop condition) followed by another message. The first byte contains the device address (0x52) and also specifies the data direction. If the least significant bit is low (that is, 0x52) the message is a master write to the slave. If the lsb is set (that is, 0x53) then the message is a master read from the slave.

Figure 14. VL53L0X I2C device address: 0x52



All serial interface communications with the camera module must begin with a start condition. The VL53L0X module acknowledges the receipt of a valid address by driving the SDA wire low. The state of the read/write bit (lsb of the address byte) is stored and the next byte of data, sampled from SDA, can be interpreted. During a write sequence the second byte received provide a 8-bit index which points to one of the internal 8-bit registers.

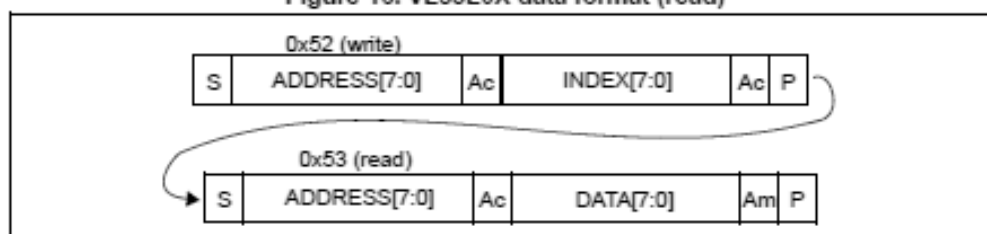
Figure 15. VL53L0X data format (write)



As data is received by the slave it is written bit by bit to a serial/parallel register. After each data byte has been received by the slave, an acknowledge is generated, the data is then stored in the internal register addressed by the current index.

During a read message, the contents of the register addressed by the current index is read out in the byte following the device address byte. The contents of this register are parallel loaded into the serial/parallel register and clocked out of the device by the falling edge of SCL.

Figure 16. VL53L0X data format (read)

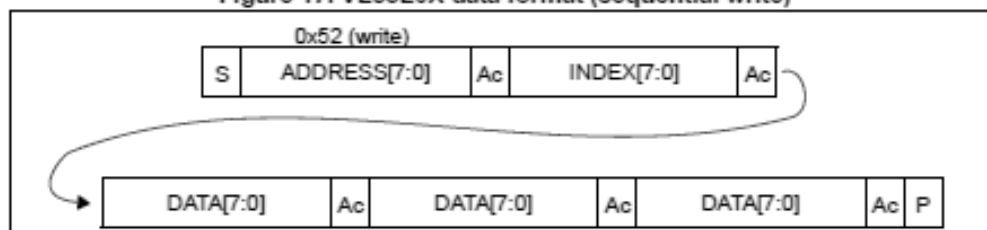


At the end of each byte, in both read and write message sequences, an acknowledge is issued by the receiving device (that is, the VL53L0X for a write and the host for a read).

A message can only be terminated by the bus master, either by issuing a stop condition or by a negative acknowledge (that is, not pulling the SDA line low) after reading a complete byte during a read operation.

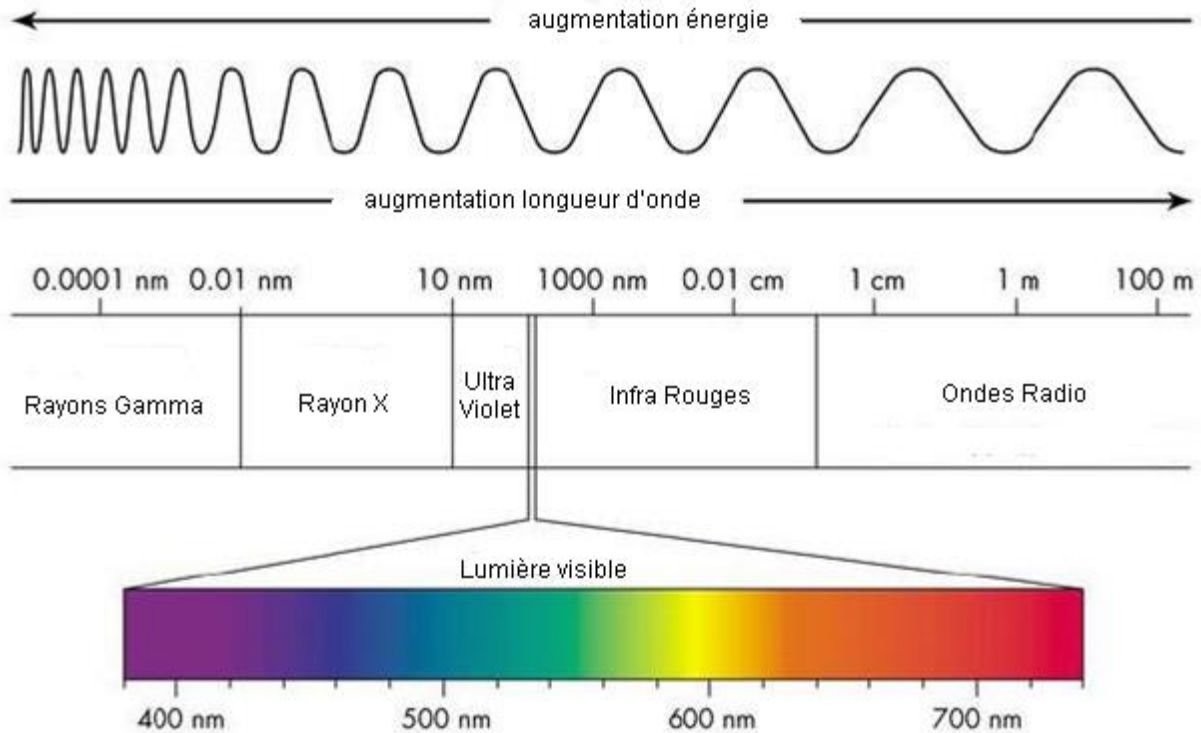
The interface also supports auto-increment indexing. After the first data byte has been transferred, the index is automatically incremented by 1. The master can therefore send data bytes continuously to the slave until the slave fails to provide an acknowledge or the master terminates the write communication with a stop condition. If the auto-increment feature is used the master does not have to send address indexes to accompany the data bytes.

Figure 17. VL53L0X data format (sequential write)



DT 7 : Longueurs d'ondes et rayonnement Laser

Longueurs d'ondes



Rayonnement laser (INRS.fr)

À chaque laser ses dangers

Les lasers sont utilisés dans de nombreux secteurs d'activités. Les yeux sont les organes les plus vulnérables. Les équipements de travail utilisant des lasers sont classés selon leur dangerosité. Les mesures de prévention à respecter, permettant une utilisation en toute sécurité, sont fonction de cette classification. Le port de lunettes de protection et de gants ininflammables est indispensable lors de l'emploi d'un laser de classe élevée.

Le laser (Light Amplification by Stimulated Emission of Radiation) produit et amplifie une **onde lumineuse**. La lumière qu'il produit est **monochromatique**, c'est-à-dire d'une couleur correspondant à une seule longueur d'onde définie, qui peut être dans l'**infrarouge**, le **visible** ou l'**ultraviolet**.

Les lasers sont utilisés dans des secteurs d'activité aussi variés que l'industrie, le BTP, les arts du spectacle, le domaine médical et hospitalier, la recherche, l'enseignement ou la défense nationale.

Classes de dangerosité

Chaque laser a une classe, indiquée par le constructeur, qui donne une idée de sa dangerosité. Une norme précise les mentions à apposer sur les appareils pour chaque classe de laser.

Classes de sécurité laser selon la norme EN 60825-1

Des informations et des conditions complémentaires sont rattachées aux définitions ci-dessous (voir la norme)

- **Classe 1** : Laser sans danger pendant leur utilisation, même en cas de vision directe dans le faisceau sur une longue période, même lorsqu'une exposition se produit lors de l'utilisation de dispositifs télescopiques. La classe 1 comprend également les lasers de forte puissance qui sont totalement enfermés de sorte qu'aucun rayonnement potentiellement dangereux ne soit accessible pendant l'utilisation (appareil avec laser incorporé). La vision dans le faisceau des appareils à laser de classe 1 qui émettent une énergie rayonnante visible peut encore produire des effets visuels d'éblouissement, en particulier à de faibles niveaux de lumière ambiante.

- **Classe 1M** : Appareils à laser émettant dans la gamme 302,5 à 4000 nm, qui sont sans danger, y compris la vision directe dans le faisceau sur une longue période pour l'œil nu. L'EMP¹ peut être dépassée et des lésions oculaires peuvent apparaître après une exposition avec un dispositif optique comme des jumelles pour un faisceau collimaté avec un diamètre tel que spécifié par la norme. La vision dans le faisceau des appareils à laser de classe 1M qui émettent une énergie rayonnante visible peut encore produire des effets visuels d'éblouissement, en particulier à de faibles niveaux de lumière ambiante.

- **Classe 1C** : Appareils à laser destinés à une application directe du rayonnement laser sur la peau ou les tissus corporels internes dans le cadre de procédures médicales, de diagnostic, thérapeutiques ou cosmétiques comme l'épilation, la réduction des rides ou de l'acné. Bien que le rayonnement laser puisse être aux niveaux des classes 3R, 3B ou 4, les expositions oculaires sont empêchées grâce à un ou plusieurs moyens techniques. Le niveau d'exposition de la peau dépend de l'application.

- **Classe 2M** : Appareils à laser qui émettent des faisceaux visibles et qui sont sans danger pour une exposition de courte durée uniquement, à l'œil nu. L'EMP peut être dépassée et des lésions oculaires peuvent apparaître après une exposition avec un dispositif optique.
Les recommandations liées aux conséquences d'éblouissement, d'aveuglement... sont identiques à celles de la classe 2.
De plus, l'étiquetage des appareils de classe 2M met aussi en garde contre une exposition des utilisateurs d'instruments optiques télescopiques.

- **Classe 3R** : Appareils à laser qui émettent des rayonnements pouvant dépasser l'EMP pour une vision directe dans le faisceau, mais le risque de lésion dans la plupart des cas est relativement faible. Le risque de lésion augmente avec la durée d'exposition et l'exposition peut être dangereuse pour une exposition oculaire dans les conditions les plus défavorables ou une vision directe dans le faisceau de manière intentionnelle. Il convient de n'utiliser les lasers de classe 3R que lorsque la vision directe dans le faisceau est peu probable.

- **Classe 3B** : Appareils à laser qui sont normalement dangereux lorsque l'exposition oculaire dans le faisceau se produit (à l'intérieur de la DNDO), y compris une exposition de courte durée accidentelle. La vision de réflexions diffuses est normalement sans danger. Les lasers de classe 3B qui s'approchent de la LEA³ de la classe 3B peuvent produire des lésions mineures de la peau, voire présenter un risque d'inflammation de matériaux inflammables. Cependant cela ne peut se produire que si le faisceau a un petit diamètre ou s'il est focalisé.

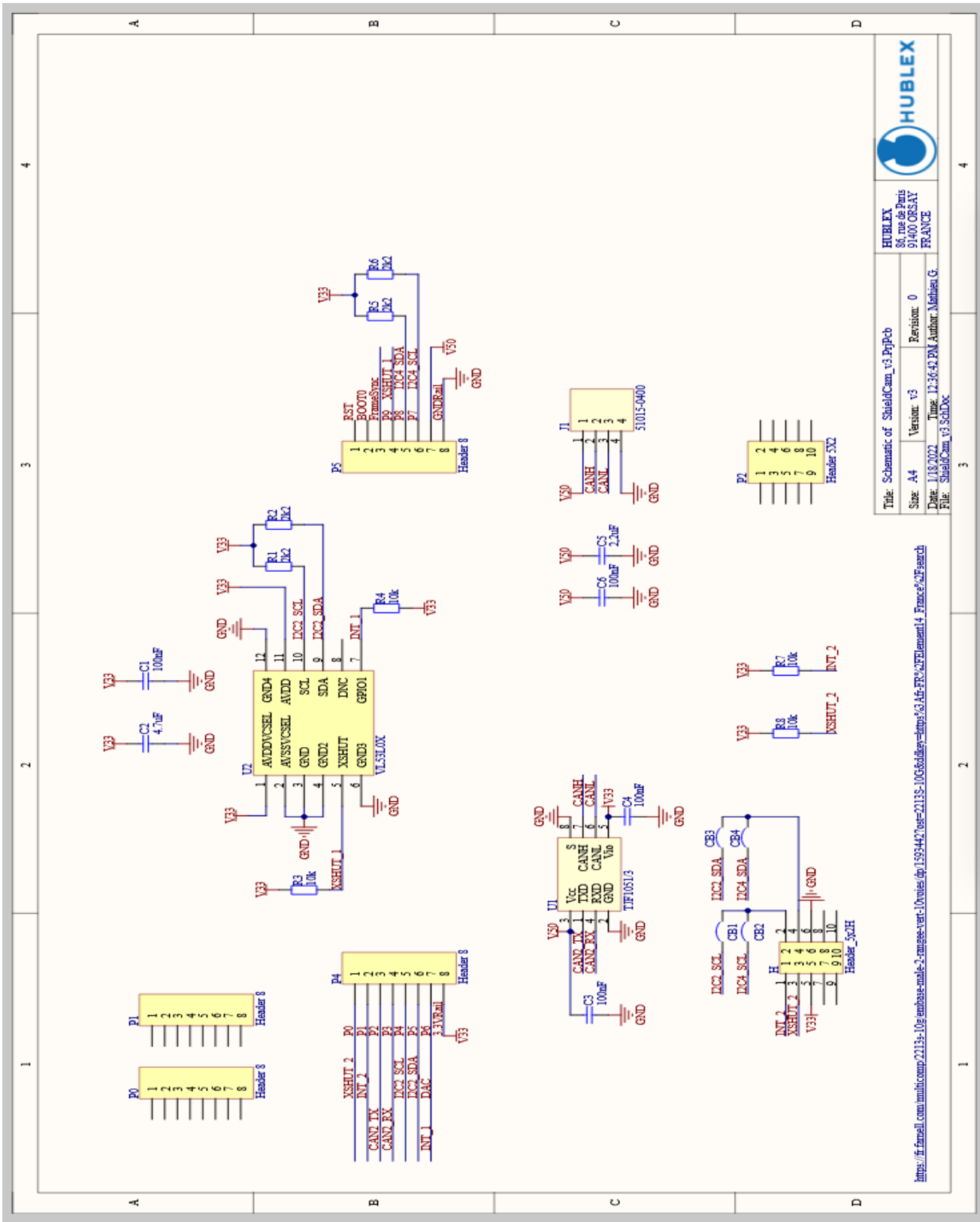
- **Classe 4** : Appareils à laser pour lesquels la vision dans le faisceau et l'exposition de la peau sont dangereuses, et pour lesquels la vision de réflexions diffuses peut être dangereuse.
Ces lasers représentent aussi souvent un danger d'incendie.

Note (1) : EMP = Exposition maximale permise – niveau du rayonnement laser auquel des personnes peuvent être exposées dans les conditions normales sans subir les effets nuisibles.

Note (2) : DNDO = Zone nominale de danger oculaire ou zone à l'intérieure de laquelle l'EMP pour la cornée est dépassée

Note (3) : LEA = Limite d'émission accessible – émission maximale permise dans une classe particulière.

DT 8 : Shield VL53L0X



HUBLEX
 30, rue de Paris
 91400 ORSAY
 FRANCE

Title: Schematic of ShieldCam_v3.Pcb
 Size: A4 Version: V3 Revision: 0
 Date: 1/18/2022 Time: 12:36:42 PM Author: Mathias G.
 File: ShieldCam_v3.SchDoc

https://fr.farnell.com/umhicomp/2113-10g-embase-male-2-embase-vert-10voies/dp/159344?on=21135-10G&oldid=159344&f=FR%2FElement4_France%2Fsearch

DT 9 : Fichier partiel de la classe VL53L0X

```
class VL53L0X:
    def __init__(self, i2c, address=0x29):
        self.i2c = i2c
        self.address = address
        self._started = False

    def _registers(self, register, values=None, struct='B'):
        if values is None:
            size = ustruct.calcsize(struct)
            data = self.i2c.mem_read(size, self.address, register)
            values = ustruct.unpack(struct, data)
            return values
        data = ustruct.pack(struct, *values)
        self.i2c.mem_write(data, self.address, register)

    def _register(self, register, value=None, struct='B'):
        if value is None:
            return self._registers(register, struct=struct)[0]
        self._registers(register, (value,), struct=struct)

    def _flag(self, register=0x00, bit=0, value=None):
        data = self._register(register)
        mask = 1 << bit
        if value is None:
            return bool(data & mask)
        elif value:
            data |= mask
        else:
            data &= ~mask
        self._register(register, data)

    def _config(self, *config):
        for register, value in config:
            self._register(register, value)

    def init(self, power2v8=True):
        self._flag(_EXTSUP_HV, 0, power2v8)

    def set_address(self, new_address):
        self._register(_I2C_SLAVE_DEVICE_ADDRESS, new_address & 0x7f)
        self.address = new_address
```

DT 10 : Extrait de la documentation de la classe PIN

Partie 7 - class Pin – control I/O pins

A pin is the basic object to control I/O pins. It has methods to set the mode of the pin (input, output, etc) and methods to get and set the digital logic level.

`Pin.value([value])`

Get or set the digital logic level of the pin:

- With no argument, return 0 or 1 depending on the logic level of the pin.
- With `value` given, set the logic level of the pin. `value` can be anything that converts to a boolean. If it converts to `True`, the pin is set high, otherwise it is set low.

DT 11 : Format d'une trame CAN

Une trame CAN est composée des champs suivants :

| | | | | | | |
|-------|-----------|----------|---------|-----|-----|-----|
| Start | Arbitrage | Commande | Données | CRC | ACK | EOF |
|-------|-----------|----------|---------|-----|-----|-----|

SOF (Start Of Frame) : Début de trame. Il est réduit à sa plus simple expression : 1 bit dominant qui peut apparaître dès la fin de l'IFS. Il sert de synchronisation pour toutes les stations.

Champ d'arbitrage : Il contient 2 données qui sont l'identificateur (11 bits en mode standard et 29 en mode étendu) et le bit de RTR (Remote Transmission Request) qui est dominant (= '0') pour une trame de données et récessif (= '1') pour une trame de requête.

Champ de commande : Il comporte 6 bits. Les 2 premiers sont toujours dominants en mode standard. Les 4 suivants représentent le DLC (Data Length Code), c'est-à-dire le nombre d'octets présents dans le champ Data.

Champ de données (Data) : Il peut varier de 0 à 8 octets. Dans le cas d'une trame de requête, ce champ est vide.

Champ de CRC (Contrôle de Redondance Cyclique) : Il est composé de 16 bits (15 + 1). La séquence calculée CRC est contenue dans les 15 premiers bits tandis que le dernier bit est un délimiteur de fin de champ de CRC (bit toujours récessif).

Champ d'acknowledge (acquiescement) : Il est composé de 2 bits laissés libres par la station émettrice. Le premier correspond à l'acquiescement par l'ensemble des nœuds ayant reçus le message ('0' signifie acquiescé et donc, pas d'erreur détectée ; '1' signifie non acquiescé, émission d'une trame d'erreur). Le second est toujours récessif et délimite l'« acknowledge ».

EOF (End of Frame) : Ce champ de fin de trame est constitué de 7 bits récessifs.



CANopen

1. Avertissement

Cette partie décrit l'interface de communication avec les contrôleurs HUBLEX basée sur le protocole CANopen.

Des différences entre ce document et le standard CANopen peuvent exister. Ce document fait référence dans le cadre de l'utilisation des contrôleurs moteur HUBLEX.

Pour plus d'informations sur le CANopen : <https://en.wikipedia.org/wiki/CANopen> ou le site du CiA : <https://www.can-cia.org/>

2. Description du protocole

Les contrôleurs moteur Hublex disposent d'une interface CAN implémentant le protocole CANopen.

Le protocole CANopen définit le vocabulaire suivant :

- Un **noeud** : chaque périphérique sous protocole CANopen possède un id unique ou noeud sur le bus CAN.
- Un **dictionnaire d'objets** : il s'agit d'une liste d'objets représentant les données modifiables/accessibles sur le bus CAN. Chaque objet est identifié par un index et d'éventuels sous-index.
- Un **index** sur 16 bits identifie un objet du dictionnaire d'objets.
- Un **sous-index** sur 8 bits identifie un sous-objet d'un objet.

Par ailleurs, le CANopen définit les messages suivants :

- Les **Services Data Object (SDO)**. Ils permettent l'accès aux données du périphérique via un système de requête/réponse. Ils permettent également de modifier des données du périphérique. La donnée concernée par l'écriture/la lecture est identifiée par le couple (index/sous-index).
- Les **Process Data Object (PDO)** sous 2 formes : les **Transmit PDO (TPDO)** et les **Receive PDO (RPDO)**. Les contrôleurs Hublex n'utilisent que les TPDO. Ils permettent l'envoi périodique de données au sein de 4 messages CAN par noeud.
- Le **Heartbeat** : un message d'un octet envoyé périodiquement qui indique l'état du périphérique.

3. Service Data Object

2 types de messages permettent l'échange de données :

- Les requêtes/commandes qui permettent à un "maître" de demander une donnée à un noeud ou d'en modifier une.
- Les réponses qui permettent à un "esclave"/noeud de répondre à la requête.

1. Commande SDO ou requête

Le format du message est le suivant :

| Header | DLC | Payload | | | | | |
|----------|-----|----------|----------|----------|-----------|----------|-----------|
| 0x600+nd | 8 | Byte 0 | | | Bytes 1-2 | Byte 3 | Bytes 4-7 |
| | | bits 4-7 | bits 2-3 | bits 0-1 | index | subindex | data |
| | | css | xx | xx | | | |

Où :

- **nd** est le noeud de destination,
- **css** est le Client Command Specifier (2 si une commande, 4 si une requête),
- **xx** n'est pas utilisé,
- **index** et **subindex** définissent la donnée à accéder,
- **data** sont les données (4 octets maximum).

2. Réponse SDO

Le format du message est le suivant :

| Header | DLC | Payload | | | | | |
|----------|-----|----------|----------|----------|-----------|----------|-----------|
| 0x580+nd | 8 | Byte 0 | | | Bytes 1-2 | Byte 3 | Bytes 4-7 |
| | | bits 4-7 | bits 2-3 | bits 0-1 | index | subindex | data |
| | | css | xx | xx | | | |

Où :

- **nd** est le noeud de destination,
- **css** est le Client Command Specifier (4 si une réponse à une requête, 6 si la commande a fonctionné, 8 si elle a échoué),
- **xx** n'est pas utilisé,
- **index** et **subindex** définissent la donnée à accéder,
- **data** sont les données (4 octets maximum). Applicable seulement en cas de réponse à une requête (**css** = 4).

3. Transmit PDO

Les messages TPDO sont au nombre de 4 par noeud. Seuls certains peuvent être activés selon l'application. La fréquence périodique d'envoi dépend également de l'application.

Les ids des messages TPDO sont les suivants :

- **TPDO1** : id=0x180+noeud,
- **TPDO2** : id=0x280+noeud,
- **TPDO3** : id=0x380+noeud,
- **TPDO4** : id=0x480+noeud.

Les 8 octets de données de la trame CAN sont disponibles pour y insérer des données du périphérique.

4. Heartbeat

Le format est le suivant :

| Header | DLC | Byte 0 |
|----------|-----|--------|
| 0x700+nd | 1 | State |

Où :

- **nd** est le noeud de destination ;
- **State** est une information d'état du périphérique.

Exemple :

4. Calibration moteur

CANOpen id : 0x2020

Description :

Ce message permet d'effectuer la procédure de calibration du moteur-roue branché au contrôleur. L'algorithme effectue plusieurs mises en rotation dans les deux directions afin de déterminer :

1. L'ordre de branchement des phases (0 : standard ; 1 : inversé) ;
2. Le décalage angulaire sur 16 bits signés entre les effets Hall et l'axe de référence interne à l'algorithme.

Le nombre de cycles de rotation n'est pas constant et dépend de chaque moteur. L'algorithme peut ainsi mettre un certain temps avant de trouver la bonne valeur de calibration.

Si le contrôleur n'a jamais été calibré ou suite à une mise à jour de la flash, la procédure de calibration peut être lancée sans condition.

Si le moteur a déjà été calibré, la procédure ne peut être lancée que si aucune commande non nulle n'a été envoyée depuis le dernier boot.

IMPORTANT : Le moteur doit être libre de toute rotation pendant cette phase. La procédure doit être effectuée après chaque changement de moteur roue.

Sous-index : Non applicable

Argument : La clé 0x4B584248 doit être envoyée.

Réponse : Aucune

Exemple : Procédure de calibration du moteur (noeud 1) :

| Header | DLC | Payload | | | | | |
|----------|-----|----------|----------|-----------|---------------|---------------------|---------------------|
| 0x601 | 8 | Byte 0 | | | Bytes 1-2 (1) | Byte 3 | Bytes 4-7 (1) |
| | | bits 4-7 | bits 2-3 | bits 0-1 | 0x2020 | - | 0x4B 0x58 0x42 0x48 |
| | | 2 | 0 | 0 | | | |
| Data CAN | | 0x20 | | 0x20 0x20 | 0x00 | 0x48 0x42 0x58 0x4B | |

(1) : Octet de poids faible en première position

DT 13 : Calcul du CRC

| | | |
|---|--|--|
| BOSCH | <h3 style="margin: 0;">Data Frame</h3> | |
| <p>DATA FRAME: admissible numbers of data bytes: {0,1,.....,7,8}. Other values may not be used.</p> <p>DATA FIELD (Standard Format as well as Extended Format) The DATA FIELD consists of the data to be transferred within a DATA FRAME. It can contain from 0 to 8 bytes, which each contain 8 bits which are transferred MSB first.</p> <p>CRC FIELD (Standard Format as well as Extended Format) contains the CRC SEQUENCE followed by a CRC DELIMITER.</p> <div style="border: 1px solid black; padding: 10px; margin: 10px auto; width: fit-content;"> </div> <p>CRC SEQUENCE (Standard Format as well as Extended Format) The frame check sequence is derived from a cyclic redundancy code best suited for frames with bit counts less than 127 bits (BCH Code). In order to carry out the CRC calculation the polynomial to be divided is defined as the polynomial, the coefficients of which are given by the destuffed bit stream consisting of START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, DATA FIELD (if present) and, for the 15 lowest coefficients, by 0. This polynomial is divided (the coefficients are calculated modulo-2) by the generator-polynomial:</p> $X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1.$ <p>The remainder of this polynomial division is the CRC SEQUENCE transmitted over the bus. In order to implement this function, a 15 bit shift register CRC_RG(14:0) can be used. If NXTBIT denotes the next bit of the bit stream, given by the destuffed bit sequence from START OF FRAME until the end of the DATA FIELD, the CRC SEQUENCE is calculated as follows:</p> <pre style="margin-top: 20px;">CRC_RG = 0; // initialize shift register</pre> | | |
| <small>ROBERT BOSCH GmbH, Postfach 300240, D-7000 Stuttgart 30</small> | | |

DT 14 : Extraits de tables intermédiaires

Résultat de la requête :

SELECT * FROM trajet as T_extrait WHERE ID_Tr<376 AND ID_Tr>354

T_extrait :

| ID_Tr | date | durée (s) | distance (m) | Tag RFID | num. série |
|-------|------------|-----------|--------------|--------------|----------------------------|
| 355 | 19/04/2021 | 218 | 299 | 0x0C00515C65 | 0x104534305234571500430079 |
| 356 | 19/04/2021 | 65 | 44 | 0x0C00515C65 | 0x104534305234571500430079 |
| 357 | 19/04/2021 | 271 | 496 | 0x0C00519C20 | 0x104534305234571500430079 |
| 358 | 19/04/2021 | 207 | 422 | 0x0C00519C20 | 0x104534305234571500430079 |
| 359 | 20/04/2021 | 308 | 623 | 0x0C00519C00 | 0x104534305234571500430079 |
| 360 | 20/04/2021 | 318 | 704 | 0x0C00519C00 | 0x104534305234571500430079 |
| 361 | 20/04/2021 | 116 | 225 | 0x0C00515C65 | 0x104534305234571500430079 |
| 362 | 20/04/2021 | 90 | 134 | 0x0C00515C65 | 0x104534305234571500430079 |
| 363 | 20/04/2021 | 327 | 448 | 0x0C00515C65 | 0x104534305234571500430079 |
| 364 | 20/04/2021 | 0 | 0 | 0xFFFFFFFF | 0x104534305234571500430079 |
| 365 | 20/04/2021 | 198 | 276 | 0x0C00515C65 | 0x104534305234571500430079 |
| 366 | 20/04/2021 | 192 | 252 | 0x0C00515C65 | 0x104534305234571500430079 |
| 367 | 20/04/2021 | 1224 | 126 | 0x0C00515C65 | 0x104534305234571500430079 |
| 368 | 20/04/2021 | 256 | 420 | 0x0C00519C20 | 0x104534305234571500430079 |
| 369 | 20/04/2021 | 69 | 123 | 0x0C00519C20 | 0x104534305234571500430079 |
| 370 | 20/04/2021 | 61 | 117 | 0x0C00519C20 | 0x104534305234571500430079 |
| 371 | 20/04/2021 | 85 | 175 | 0x0C00519D15 | 0x104534305234571500430079 |
| 372 | 20/04/2021 | 0 | 0 | 0xFFFFFFFF | 0x104534305234571500430079 |
| 373 | 20/04/2021 | 0 | 0 | 0xFFFFFFFF | 0x104534305234571500430079 |
| 374 | 20/04/2021 | 22 | 17 | 0x0C00519C20 | 0x104534305234571500430079 |
| 375 | 20/04/2021 | 520 | 881 | 0x0C00519C20 | 0x104534305234571500430079 |

Résultat de la requête :

SELECT * FROM evenement as E_extrait WHERE ID_Ev<21

E_extrait :

| ID_Ev | msg | ID_Tr |
|-------|---------------------------|-------|
| 1 | MOTOR_LEFT_NOT_READY | 9 |
| 2 | MOTOR_LEFT_NOT_READY | 37 |
| 3 | BATTERY_LOW | 41 |
| 4 | MOTOR_LEFT_NOT_READY | 117 |
| 5 | MOTOR_LEFT_NOT_READY | 131 |
| 6 | RECOVERY_WATCHDOG | 292 |
| 7 | MOTOR_LEFT_NOT_READY | 295 |
| 8 | MOTOR_LEFT_NOT_READY | 300 |
| 9 | MOTOR_RIGHT_FAIL_SAFESTOP | 356 |
| 10 | MOTOR_LEFT_FAIL_SAFESTOP | 356 |
| 11 | MOTOR_RIGHT_NOT_READY | 364 |
| 12 | BATTERY_LOW | 372 |
| 13 | BATTERY_LOW | 372 |
| 14 | BATTERY_LOW | 372 |
| 15 | MOTOR_LEFT_NOT_READY | 457 |
| 16 | MOTOR_RIGHT_FAIL_SAFESTOP | 546 |
| 17 | MOTOR_LEFT_NOT_READY | 559 |
| 18 | MOTOR_LEFT_NOT_READY | 559 |
| 19 | MOTOR_RIGHT_NOT_READY | 563 |
| 20 | MOTOR_LEFT_NOT_READY | 580 |

DT 15 : Capture Wireshark

Apply a display filter: ... <Ctrl+>

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------|-----------------|-----------------|----------|--------|--|
| 1 | 0.000000 | 192.168.43.21 | 192.168.43.1 | DNS | 76 | Standard query 0xff4b A sites.google.com |
| 2 | 0.197008 | 192.168.43.1 | 192.168.43.21 | DNS | 92 | Standard query response 0xff4b A sites.google.com A 142.250.178.142 |
| 3 | 0.199223 | 192.168.43.21 | 142.250.178.142 | TCP | 66 | 51556 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 4 | 0.217116 | 142.250.178.142 | 192.168.43.21 | TCP | 66 | 443 → 51556 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1430 SACK_PERM=1 WS=256 |
| 5 | 0.221289 | 192.168.43.21 | 142.250.178.142 | TLSv1.3 | 712 | Client Hello |
| 6 | 0.221534 | 192.168.43.21 | 142.250.178.142 | TLSv1.3 | 60 | Change Cipher Spec |
| 7 | 0.221909 | 192.168.43.21 | 142.250.178.142 | TLSv1.3 | 1657 | Application Data |
| 8 | 0.242205 | 142.250.178.142 | 192.168.43.21 | TLSv1.3 | 816 | Server Hello, Change Cipher Spec, Application Data, Application Data |
| 9 | 0.242205 | 142.250.178.142 | 192.168.43.21 | TLSv1.3 | 116 | Application Data |
| 10 | 0.242205 | 142.250.178.142 | 192.168.43.21 | TLSv1.3 | 85 | Application Data |
| 11 | 0.243178 | 192.168.43.21 | 142.250.178.142 | TLSv1.3 | 138 | Application Data, Application Data |
| 12 | 0.248150 | 192.168.43.21 | 142.250.178.142 | TLSv1.3 | 85 | Application Data |
| 13 | 1.469781 | 142.250.178.142 | 192.168.43.21 | TLSv1.3 | 945 | Application Data |

> Frame 12: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface \Device\NPF_{8695B4CA-8EE9-4742-96FA-4AEDAD05401B}, id 0

- > Ethernet II, Src: IntelCor_ae:3f:1c (40:1c:83:ae:3f:1c), Dst: 8e:f5:a3:4e:4b:b7 (8e:f5:a3:4e:4b:b7)
- > Internet Protocol Version 4, Src: 192.168.43.21, Dst: 142.250.178.142
- > Transmission Control Protocol, Src Port: 51556, Dst Port: 443, Seq: 2352, Ack: 856, Len: 31
- ▼ Transport Layer Security
 - ▼ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
 - Opaque Type: Application Data (23)
 - Version: TLS 1.2 (0x0303)
 - Length: 26
 - Encrypted Application Data: e768c114716b7e3bfd88ce088f2bdf9143784e46bc396188c7ea
 - [Application Data Protocol: http-over-tls]


```

0000 8e f5 a3 4e 4b b7 40 1c 83 ae 3f 1c 08 00 45 00 ...NK.@. ...?....E.
0010 00 47 3c 1f 40 00 80 06 00 00 c0 a8 2b 15 8e fa -G<.@... ..+...
0020 b2 8e c9 64 01 bb 74 28 12 fe 88 d7 38 9b 50 18 ...d.t( ...8.P.
0030 00 fd 2d 80 00 00 17 03 03 00 1a e7 68 c1 14 71 ... ..h..q
0040 6b 7e 3b fd 88 ce 08 8f 2b df 91 43 78 4e 46 bc k~;.....+.CxFN.
0050 39 61 88 c7 ea 9a...
```

DT 16 : Fichier simplifié index.html

```
<!doctype html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title> Site client | Accueil</title>
  <link rel="stylesheet" href="styles/style_client.css">
  <script src="scripts/script.js"></script>
</head>
<body>
  <div class="page">
    <p>Rapports de suivi d'utilisation</p>
    <!-- Formulaire de connexion client-->
    <form action="auth.php" onsubmit="return valider_saisie()" method="POST" name="connexion">
      Identifiant: <input id="user" type="text" name="utilisateur" size="20"> <br>
      Mot de passe: <input id="pwd" type="password" name="mot_de_passe" size="20">
      <input type="submit" name="Envoyer" value="Envoyer">
    </form>
  </div>
</body>
</html>
```

DT 17 : Fichier partiel rapports.php

```
1 <?php
2 /* fichier de configuration de la base de données qui définit des constantes
3 d'environnement : DBNAME (nom), DBHOST(serveur), DBUSER (utilisateur), DBPASS(mot de passe)*/
4 include ('config.php');
5
6 //nom de la base
7 $dbn = "mysql:dbname=".DBNAME."; host=".DBHOST;
8
9 //connexion à la base
10 try{
11   $dbconnect = new PDO($dbn, DBUSER, DBPASS);
12   echo "Connexion établie";
13   $dbconnect->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
14 }catch(PDOException $e){
15   die("Erreur : ".$e->getMessage());
16 }
17 //préparer et exécuter la requête
18 $sql = "SELECT `ID_tr`, `duree`, `date` FROM `trajet`";
19 $req = $dbconnect->query($sql);
20
21 echo"<h2> Durées des trajets enregistrés dans la base".DBNAME." </h2>";
22
23 //Récupérer les données et les afficher
24 $donnees = $req->fetchAll();
25 foreach ($donnees as $resultat) {
26   //afficher les données concernées
27   ...
28 }
29 $req->closeCursor();
30
31 // autres requêtes;
32 // ...
33 ?>
```

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

| | | | | | | | | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|



Prénom(s) :

| | | | | | | | | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

Numéro Inscription :

| | | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

Né(e) le :

| | |
|--|--|
| | |
|--|--|

 /

| | |
|--|--|
| | |
|--|--|

 /

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

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

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

Concours / Examen : **Section/Specialité/Série :**

Epreuve : **Matière :** **Session :**

CONSIGNES

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

DR1 à DR8

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

NE RIEN ECRIRE DANS CE CADRE

DR 1

```
while(len(cam_analysing.tags_on_CAN) > 0) :  
    tag = cam_analysing.tags_on_CAN[0]  
    can.send_tag(tag[0], tag[1], tag[2], tag[3], tag[4])  
    .....
```

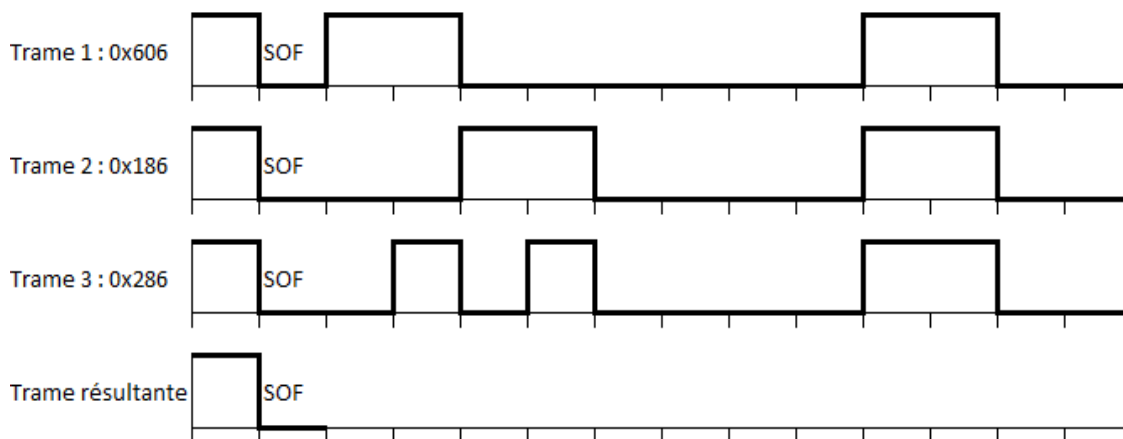
DR 2

```
# obstacle.py  
import gc  
from utime import sleep_ms  
from pyb import I2C, Pin  
gc.collect()  
from vl53l0x import VL53L0X  
from can import offset_distance_front  
  
# Objects (I2C, INT, XSHUT, VL53L0X)  
i2c = I2C(2)  
INT_DOWN = Pin('P6', Pin.IN)  
INT_FRONT = Pin('P1', Pin.IN, pull=Pin.PULL_UP)  
XSHUT_DOWN = Pin(_____, Pin.OUT_OD)  
XSHUT_FRONT = Pin(_____, Pin.OUT_OD)  
laser_down = VL53L0X(i2c, _____)  
laser_front = VL53L0X(i2c, _____)
```

DR 3



DR 4

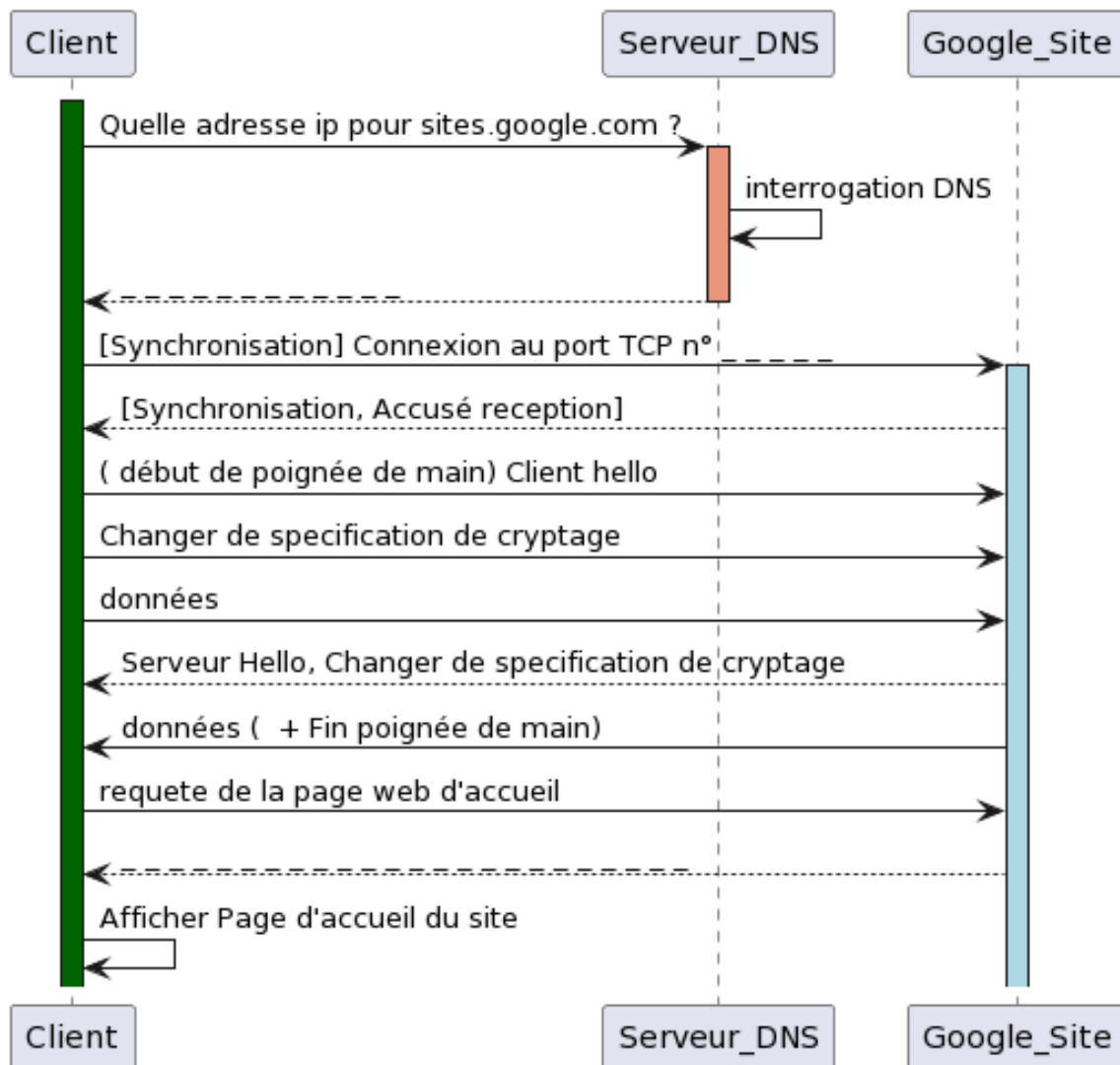


DR 5

| Header | DLC | Payload | | | | | |
|----------|-----|----------|----------|----------|---------------|--------|---------------|
| | 8 | Byte 0 | | | Bytes 1-2 (1) | Byte 3 | Bytes 4-7 (1) |
| | | bits 4-7 | bits 2-3 | bits 0-1 | | | |
| | | | | | | | |
| Data CAN | | | | | | | |

DR 6

Diagramme de séquence lors de la connexion au site de suivi du client.



DR 7

Fichier script.js :

```
1 function valider_saisie() {
2     var entree = 0;
3     var user = document.getElementById("user").value;
4     var mdp = _____ ;
5     if (user == "") { entree += 1 }
6     if (mdp == "") { entree += ____ }
7
8     switch (entree) {
9         case 0:
10            return true;
11            break;
12        case 1:
13            alert("Saisir le nom d'utilisateur ") ;
14            return false;
15            break;
16        case 2:
17            alert("Saisir le mot de passe ");
18            return false;
19            break;
20        case 3:
21            _____ ;
22            return false;
23            break;
24    }
}
```

DR 8

```
17 //préparer et exécuter la requête
18 $sql = "SELECT `ID_tr`, `duree`, `date` FROM `trajet`";
19 $req = $dbconnect->query($sql);
20
21 echo"<h2> Durées des trajets enregistrés dans la base".DBNAME." </h2>";
22
23 //Récupérer les données et les afficher
24 $donnees = $req->fetchAll();
25 foreach ($donnees as $resultat) {
26     echo "Identifiant : <strong>".$resultat['Id_tr']."</strong><br/>";
27     _____ ;
28     _____ ;
29 }
30 $req->closeCursor();
```