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

EDE NUM 1

SESSION 2022

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.

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

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

Tournez la page S.V.P.

COMPOSITION DU SUJET

SUJET :

DOCUMENTS :

Le sujet comporte sept parties. Bien que les parties soient indépendantes, il est tout de même préférable de traiter le sujet en tenant compte de la chronologie proposée.

Toutes les réponses devront être détaillées sur la copie et les résultats encadrés ou soulignés. Préciser les unités des résultats.

Tous les documents réponses, complétés ou non, sont à rendre avec les copies.

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

Projet de surveillance d'un site industriel

AVERTISSEMENT :

Quel que soit le langage utilisé dans le questionnement, la syntaxe retenue pour l'écriture des fonctions (ou des méthodes) et pour la définition des variables (ou des objets) est basée sur la notation *lower Camel Case* (casse du chameau avec le premier mot débutant par une minuscule).

Exemple : Définition d'une fonction (en langage C)

```
void maFonction (int monPremierParametre, float monSecondParametre) {
    …
}
```

Exemple : Définition d'une fonction (en Python)

```
def maFonction (monPremierParametre, monSecondParametre) :
    …
    …
    …
```

Exemple : Définition d'une fonction (en Javascript)

```
function maFonction (monPremierParametre, monSecondParametre) {
    …
    …
    …
}
```

La définition des classes utilise la notation *Upper Camel Case* (casse du chameau avec le premier mot débutant par une majuscule).

Exemple : Définition d'une classe (en langage C++)

```
class MaClasse {
    …
};
```

Partie 1. **Mise en situation et contexte de l'étude**

Objectif : *S'approprier le contexte de l'étude et l'importance de la mise en place d'une surveillance du site de Malakoff.*

1.1 *Réseau de chauffage urbain Centre Loire*

Le réseau de chaleur Centre Loire s'inscrit dans la politique de transition énergétique de Nantes Métropole et son Plan Climat Air Énergie Territorial, avec pour objectif de réduire de 50 % les émissions de CO_2 par habitant d'ici 2030 et d'atteindre 50 % d'énergies renouvelables locales pour les besoins du territoire d'ici 2050.
ERENA, filiale d'Engie Réseaux, Groupe ENGIE, a été choisie par Nantes Métropole pour la Délégation de Service Public du réseau de chaleur « Centre Loire » pour une durée de 20 ans.
Un réseau de chauffage urbain, également appelé réseau de chaleur, est un chauffage central à l'échelle d'une ville. Il permet d'alimenter des bâtiments (privés, publics, industriels) en chauffage, en eau chaude sanitaire ou en process (pour l'industrie : vapeur, eau surchauffée, …).
Le réseau de chauffage urbain est généralement un réseau public, qui transporte la chaleur sous forme d'eau chaude ou de vapeur dans des canalisations enterrées, comme les réseaux électriques et gaziers. Les clients se raccordent à ce réseau pour s'approvisionner en chaleur.
Cette chaleur est produite dans des installations (unités de production de chaleur ou encore chaufferies) de très grande puissance, gérées de façon industrielle avec la meilleure maîtrise possible de la combustion et des rejets dans l'atmosphère.

ERENA a réalisé l'extension du réseau sur près de 68 km et la construction de deux chaufferies bois avec appoint gaz pour disposer ainsi d'un bouquet énergétique constitué à 84 % par des énergies locales et renouvelables (35000 tonnes de CO_2 évitées par an**)**.

Avec 85 km de réseau à terme, Centre Loire deviendra l'un des plus importants réseaux de chaleur de France (figure 1).

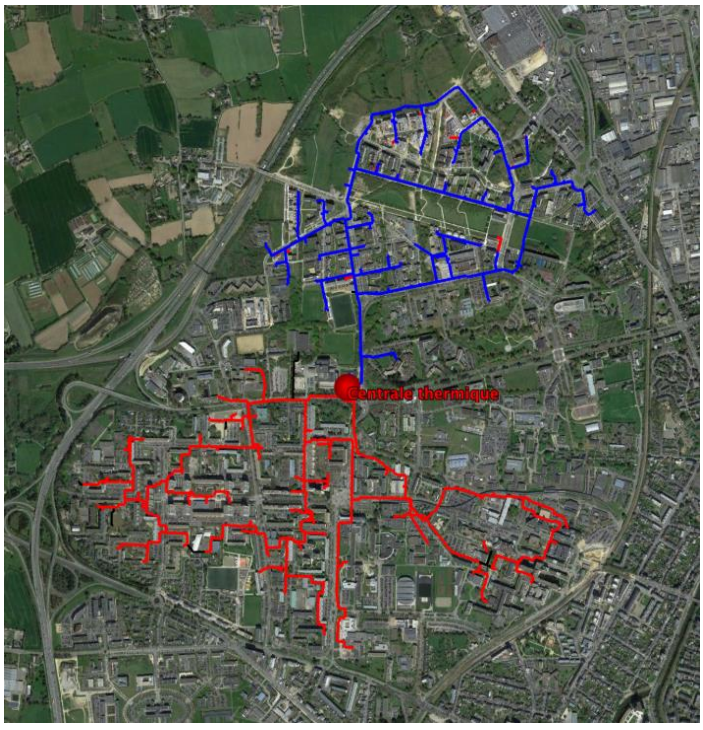


Figure 1 : Plan du réseau de chaleur

1.2 *La chaufferie bois de Malakoff*

La chaufferie (figure 2) a trois sources d'énergie : d'une part, l'usine d'incinération des déchets de la Prairie-de-Mauves, d'une puissance de 30 mégawatts ; d'autre part, deux chaudières à bois, de 15 mégawatts chacune. Enfin, il y a aussi des chaudières d'appoint à gaz, en cas de froid vif, ou en cas de panne.

Pour amener les copeaux de bois à la chaudière, un tapis roulant mû par une vis sans fin d'une longueur de 15 m est utilisé. L'état de cette vis doit être vérifié en période estivale, lorsqu'il n'y a plus aucun copeau de bois à acheminer. Pour ce faire, huit regards ont été prévus mais leur accès reste difficile car l'environnement est particulièrement exigu. La chaufferie prévoit de s'équiper de robots pour obtenir un visuel sur chaque regard.

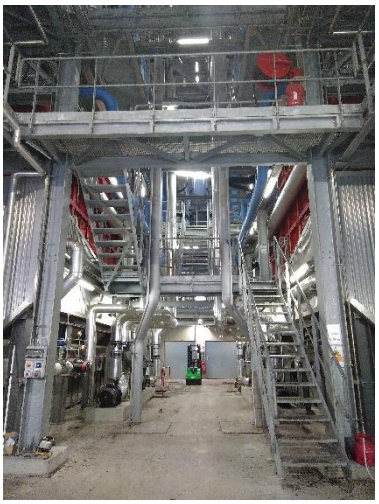


Figure 2 : Chaufferie de Malakoff

1.3 *Robot ArIA*

Pilgrim Technology est un spécialiste et acteur référent de l'inspection industrielle. Il fabrique des drones et des robots afin de répondre aux problématiques spécifiques de ses clients.

ArIA (figure 3) est un robot agile, capable de se déplacer dans les espaces difficiles d'accès en se faufilant par exemple sous les tuyaux ou en enjambant des obstacles.
Il dispose d'une caméra de surveillance et il peut embarquer en option une série de capteurs (distance, gaz, présence, laser) et d'autres types de caméras (thermique, stéréoscopique).

Dans sa version de base, il dispose de deux modes de déplacement. Un mode principal disponible à partir d'une interface Web et un mode secours réalisé à l'aide d'un simple joystick. Dans le mode principal un retour caméra permet à l'opérateur de situer le robot dans son environnement immédiat et permet son pilotage. En cas de défaillance du réseau informatique, l'opérateur peut à tout moment reprendre le contrôle sur les déplacements d'ArIA grâce au mode de secours.

1.4 *Inspection des regards*

A l'intérieur de la chaufferie, ArIA doit se positionner devant chacun des huit regards afin de fournir un visuel sur l'état de la vis d'entrainement. Compte tenu de la situation, les regards sont difficilement identifiables entre eux. Un code-barres de type EAN8 (figure 4), placé à proximité de chaque regard, permet à l'opérateur d'identifier sans ambiguïté chaque lieu grâce à l'interprétation de ce dernier.

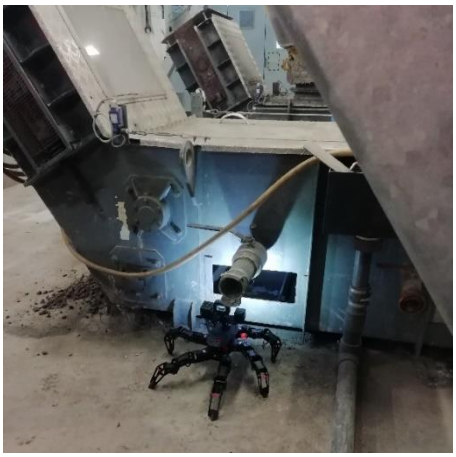


Figure 3 : Robot ArIA devant un regard

1.5 *Création des codes-barres de type EAN8*

Un code-barres de type EAN8 se compose de sept données numériques utiles suivies d'une clé de contrôle sur un seul chiffre. Les sept données utiles du code-barres sont générées de la manière suivante :

- les deux premiers chiffres servent à l'identification du site. Ce nombre est compris entre 0 et 99 inclus ;
- les deux suivants sont réservés au numéro de l'entrepôt. Ce nombre est compris entre 0 et 20 inclus ;les trois d'après permettent l'identification d'un lieu et en particulier d'un regard en commençant à 100. Ce nombre est compris entre 100 et 200 inclus.

Figure 4 : Code-barres EAN8

Tournez la page S.V.P.

Exemple :

- Le site de Malakoff possède le numéro 51.
- La surveillance est réalisée dans le second entrepôt (02).
- Les regards sont identifiés de 100 à 107 inclus.

Ainsi le code-barres EAN 8, de valeur 5102106, associé à sa clé de contrôle de valeur 1, identifie de manière unique le regard 106 du second entrepôt du site de Malakoff.

1.6 *Diagramme de cas d'utilisation et diagramme partiel des exigences*

Le diagramme de cas d'utilisation de la figure 5 reprend les principaux points énoncés.

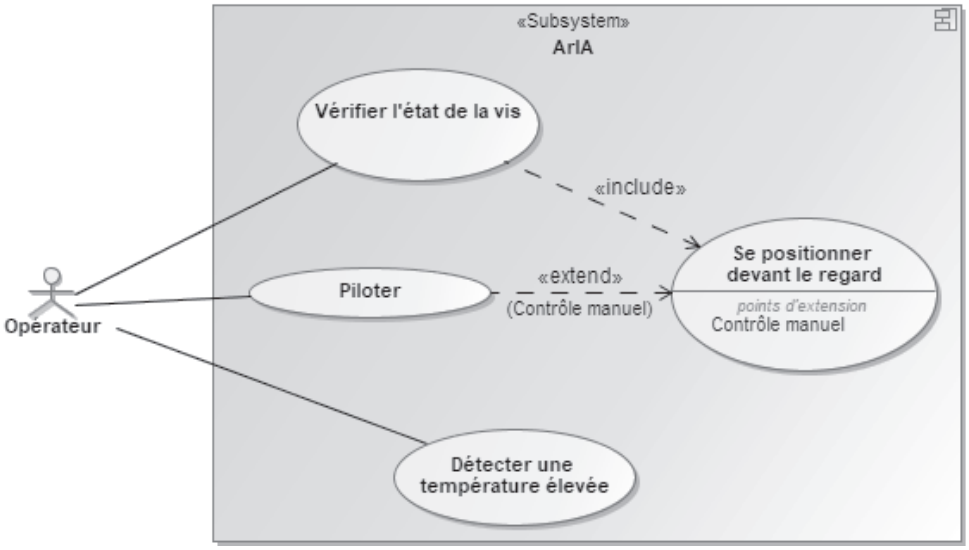


Figure 5 : Diagramme de cas d'utilisation

Le document technique DT1 présente le diagramme partiel des exigences.

1.7 *Contexte de l'étude*

Pour la gestion de sa sécurité, l'usine de Malakoff envisage l'achat d'un à trois robots ArIA. Cinq employés ayant suivi une formation spécifique, nommés **Opérateurs**, seront qualifiés pour procéder aux inspections des différents lieux et en particulier des huit regards identifiés de 100 à 107.

L'opérateur est donc un employé qualifié muni d'un poste de travail composé d'un ordinateur portable ou d'une tablette.

Question 1 : Sur le document réponse DR1, compléter le diagramme de contexte (diagramme de bloc - Block Definition Diagram - de l'outil de modélisation SysML), avec les cardinalités manquantes.

Question 2 : Préciser avec exactitude le rôle des relations entre les blocs suivants : Surveillance de l'usine de Malakoff et Regard, Surveillance de l'usine de Malakoff et ArIA, Employé et Opérateur.

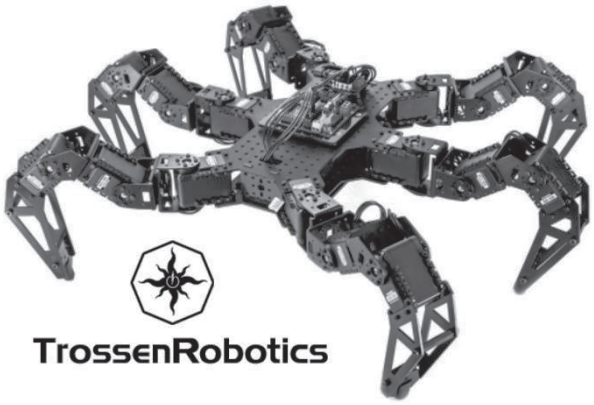
Question 3 : Conclure sur les moyens de surveillance mis en place par l'usine et sur les avantages à utiliser un hexapode par rapport à la mise en place de caméras fixes.

Partie 2. **Architecture matérielle de l'hexapode**

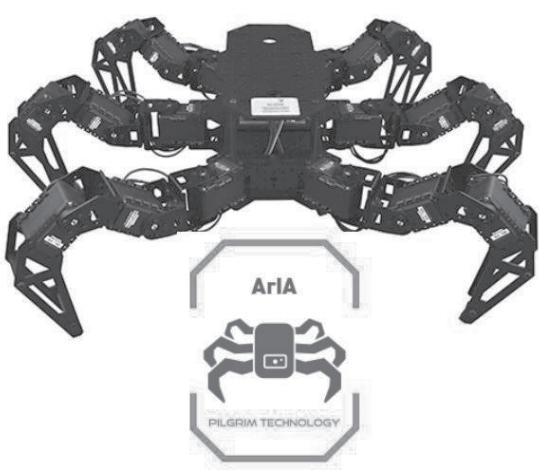
Objectif : *Réalisation de l'exigence 1.1.1 : Appréhender la composition matérielle de l'hexapode d'un point de vue motorisation.*

2.1 *Présentation générale*

Afin de développer le robot **ArIA** le plus rapidement possible et ainsi minimiser le temps de mise sur le marché, la société Pilgrim Technology s'est basée sur une structure de robot Open Source, le **PhantomX Hexapod MK-III Metal** proposée par la société Trossen Robotics. Les deux robots sont présentés sur la figure 6.



PhantomX Hexapod MK-III Metal



ArIA

Figure 6 : Robot PhantomX Hexapod MK-III Metal et Robot ArIA

Question 4 : Préciser ce que signifie le concept d'Open Source dans ce cas de figure.

Bien que le principe de l'Hexapode soit conservé, une partie conséquente de la mise au point de l'ArIA réside dans l'adaptation du robot initial aux besoins spécifiques des missions de surveillance. Pour cela, des éléments supplémentaires ont été intégrés au robot, afin de l'adapter aux missions de surveillance, entrainant de facto une augmentation significative de son encombrement et des besoins de programmation et de pilotage.

Cette augmentation des besoins de programmation et de pilotage a entrainé, entre autres, le remplacement de la carte de pilotage initiale de type **Arduino** (carte **Arbotix**) par une carte permettant de supporter un OS embarqué de type Linux. Le choix définitif s'est porté sur une carte **NVidia Jetson TX1**, cependant une carte de type **Raspberry Pi 3** sous **Raspbian** est suffisante dans la plupart des utilisations et c'est sur cette dernière que la solution proposée est développée.

2.2 *Partie Hardware – Architecture matérielle de l'hexapode*

Chaque patte est constituée de trois servo-moteurs. L'ensemble comporte six pattes.
Les dix-huit servo-moteurs de type **Dynamixel** sont fabriqués par la société **Robotis**.

Les servo-moteurs Dynamixel diffèrent des servo-moteurs plus traditionnels à commande analogique car ils disposent d'une commande totalement numérique, ce qui fait d'eux des smart

Tournez la page S.V.P.

servos. Le principe de base est d'associer à chaque servo-moteur, un identifiant unique (ID) et d'émettre vers ce dernier à l'aide d'une trame qui lui est destinée les ordres à respecter.

Les servo-moteurs utilisés sur le PhantomX sont issus d'une série mono-tour dont le débattement maximal est de 300 degrés. Les ordres sont donnés sur 10 bits.
La position centrale est très simplement repérable par la sérigraphie utilisée sur le servo-moteur et deux traits positionnés en face l'un de l'autre indiquent cette position. Elle correspond à une commande en position de valeur 512. Le servo-moteur est alors en position absolue 150°.

Le document technique DT2 complète la présentation de ces servo-moteurs et présente le protocole de communication utilisé.

Question 5 : Indiquer quels sont les angles absolus en degrés quand la commande émise contient respectivement les valeurs 412 et 612. En déduire le débattement angulaire en degrés entre ces deux valeurs de commande.

2.3 *Organisation générale du robot et spécification des termes utilisés*

Afin de s'affranchir des termes « droite et gauche », parfois ambigus dans ce contexte, il est préférable d'utiliser un vocabulaire issu du champ maritime. Ainsi, dans toute la suite, les termes **tribord** et **bâbord** seront utilisés en lieu et place de droite et gauche et les termes **proue** et **poupe** en lieu et place d'avant et arrière (même si ces derniers sont beaucoup moins ambigus compte tenu du caractère asymétrique du robot sur cet aspect).

Rappel :

- Tribord : Côté droit quand on regarde vers l'avant d'un navire
- Bâbord : Côté gauche quand on regarde vers l'avant d'un navire.
- Proue : Avant d'un navire
- Poupe : Arrière d'un navire

La figure 7, issue de la documentation de Trossen Robotics, reprend ces termes dans le contexte de ce robot et présente les différents numéros d'identifiants utilisés pour les dix-huit servo-moteurs équipant l'hexapode.

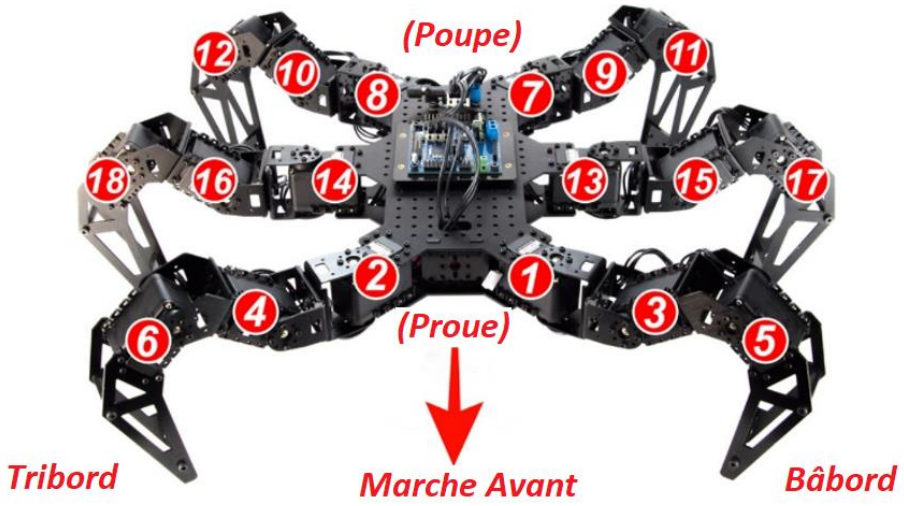


Figure 7 : Terminologie employée et organisation des servo-moteurs

Sur le robot, chacune des six pattes est constituée de trois servo-moteurs, nommés respectivement Coxa (Hanche), Fémur et Tibia comme indiqué sur la figure 8 qui reprend l'organisation retenue sur une patte bâbord.

Pour la suite, la patte, dénommée **patteBabordProue**, constituée des servo-moteurs d'IDentifiants 1, 3 et 5 est prise comme patte de référence.

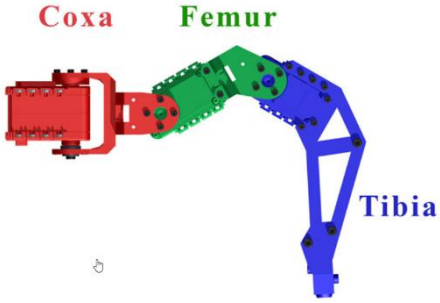


Figure 8 : Présentation des articulations d'une patte

- Servo moteur d'ID 1 : Coxa Bâbord Proue
- Servo moteur d'ID 3 : Fémur Bâbord Proue
- Servo moteur d'ID 5 : Tibia Bâbord Proue

2.4 *Paramétrage des IDentifiants*

Les servo-moteurs, fabriqués en usine sont livrés avec l'ID 1. Il faut ainsi modifier ces derniers pour les identifier définitivement suivant la représentation de la figure 7 avant toute installation sur le robot, car ces opérations s'avèrent nettement plus complexes par la suite du fait de la difficulté d'accéder aux connecteurs.

Il existe actuellement deux protocoles de communication différents, nommés respectivement **Protocole 1** et **Protocole 2**, développés par la société Robotis pour les servo-moteurs de type Dynamixel.

Les servo-moteurs retenus utilisent le **Protocole 1**. Sur ce dernier la vitesse de transmission par défaut est de 1 000 000 bps (bits par seconde) et la communication utilise un câble en nappe asymétrique à 3 fils (figure 9).

Figure 9 : Câble et connecteur

Question 6 : A partir des informations disponibles sur le connecteur et sachant que le processeur central peut aussi bien émettre des ordres vers le servo-moteur que recevoir l'état de ce dernier, indiquer en le justifiant la nature des communications (synchrone, asynchrone, simplex, half duplex ou full duplex) mises en place dans cette communication.

Nous allons nous intéresser à la patte de référence et reprogrammer deux des trois servo-moteurs avec respectivement les IDentifiants 3 et 5.

La société Robotis fournit un logiciel spécialisé dénommé **Dynamixel Wizard 2.0**, disponible sous Windows et sous Linux. Deux copies d'écran sont présentées sur le document technique DT3. Les trames affichées dans la partie basse de chaque copie d'écran sont celles relatives à la zone en surbrillance (respectivement la trame émise à 16:35:59.077 et celle reçue à 16:36:35.285).

Question 7 : A partir du document technique DT3, indiquer quelles sont les actions réalisées par ces deux étapes de paramétrage sachant que le registre d'adresse **3** représente le registre d'**ID**. Quelle est alors la modification obtenue ?

Question 8 : A partir de la documentation disponible sur le document technique DT2, justifier les valeurs 0xEF et 0xF2 présentes dans le champ le plus à droite des deux trames.

Question 9 : Pour le Fémur de la patte de référence, déterminer la trame complète (avec le champ header) nécessaire pour modifier l'ID d'usine. Donner la trame complète

Tournez la page S.V.P.

renvoyée par la commande Read lors de la phase de vérification de cette reprogrammation.

On fait désormais l'hypothèse que les IDentifiants de chacun des dix-huit servo-moteurs constituant les six pattes du robot sont correctement positionnés et correspondent à la description imposée sur la figure 8.

2.5 *Principe du chainage des servo-moteurs*

La mise en place des servo-moteurs repose sur le principe du **Daisy-Chain**, c'est à dire un câblage en chainage. (Un connecteur arrive jusqu'au servo-moteur d'indice i et repart vers un servo-moteur d'indice j).
Le principe est résumé sur le schéma de la figure 10, aisément disponible sur Internet.

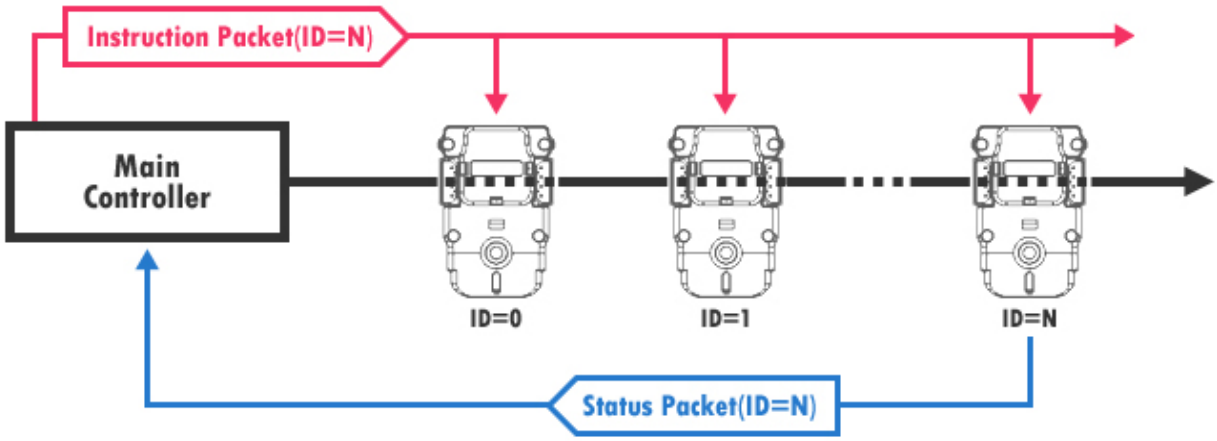


Figure 10 : Chainage des servo-moteurs

Le choix technologique de réalisation de l'hexapode est un chainage par bloc de trois servo-moteurs, directement basé sur la description fonctionnelle des pattes du robot. Ainsi, la patte de référence composée des trois servo-moteurs d'IDentifiants respectifs 1, 3 et 5 est en Daisy Chain.

L'utilisation de deux concentrateurs supplémentaires, associés à un convertisseur **U2D2** (figure 11) permet le chainage vers les six pattes à partir de la sortie USB de la carte **Raspberry Pi 3**.

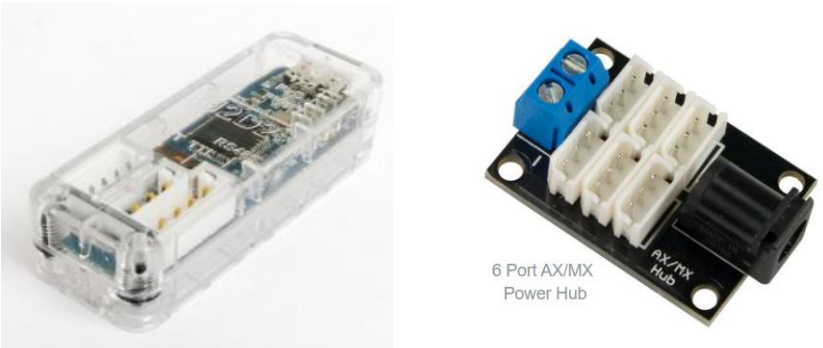


Figure 11 : Convertisseur U2D2 et concentrateur (Hub)

Question 10 : Indiquer quels sont les avantages et les inconvénients de ce type de câblage en réseau.

Question 11 : A partir des informations contenues sur le document technique DT2 et sachant d'une part que le registre d'adresse **30** représente le registre de **Goal Position** (mise en position) et d'autre part que les données liées à la commande souhaitée sont en petit boutisme (little endian), indiquer ce que réalise l'émission sur le bus de la trame représentée sur la figure 12.

Header		ID	Len	I/E	Param			Cksm
FF	FF	FE	05	03	1E	00	02	D9

Figure 12 : Trame sur le bus

Question 12 : Quelle est la durée de la trame émise ? Quel est le pourcentage d'informations utiles par rapport à la trame complète ?

2.6 *Conclusion*

Question 13 : A partir de l'ensemble des informations disponibles, conclure sur la possibilité de programmer le robot et sur la réalisation de l'exigence 1.1.1.

Tournez la page S.V.P.

Partie 3. **Architecture logicielle de l'hexapode**

Objectif : *Réalisation de l'exigence 1.2 : Mettre en place l'architecture logicielle orientée objet permettant la gestion du pilotage de l'hexapode.*

3.1 *Présentation*

Le programme implanté dans la carte **Raspberry Pi 3** est développé en utilisant le langage CPP (C++) et pour faciliter son évolution, suit une démarche Orienté Objet. Après une première réflexion, permettant la mise en place des premières briques logicielles, l'évolution du programme et la mise à jour de sa description UML sont réalisées de manière concourante.

Les fonctionnalités élémentaires à réaliser sont :

- placer l'hexapode à l'équilibre en position de repos ;
- placer l'hexapode à l'équilibre en position de repos surélevée ;
- déplacer l'hexapode vers l'avant d'un pas élémentaire ;
- déplacer l'hexapode vers l'arrière d'un pas élémentaire ;
- déplacer l'hexapode vers tribord d'un pas élémentaire ;
- déplacer l'hexapode vers bâbord d'un pas élémentaire.

Les procédures de vérification du fonctionnement à réaliser sont :

- vérifier la communication avec chaque servo-moteur en faisant clignoter la diode présente à l'arrière de chacun d'entre eux ;
- vérifier la commande de déplacement de chaque servo-moteur en réalisant une séquence de déplacement visible par un opérateur.

Seules les deux premières fonctionnalités élémentaires et les deux procédures de vérification seront abordées dans le cadre de ce sujet.

3.2 *Préliminaire : Gestion du port série – Classe CDcGestionPortSerieStaticOnly*

La carte **Raspberry Pi 3** est directement reliée via un port USB au convertisseur U2D2. Ce dernier est vu comme un port série de nom **/dev/ttyUSB0.** La classe **CDcGestionPortSerieStaticOnly** gère toute la communication série et elle contient trois méthodes statiques différentes. Sa définition est proposée ci-dessous :

```
class CDcGestionPortSerieStaticOnly {
public :
   static int ouverturePortSerie (void);
   static void fermeturePortSerie (int serialPort);
   static void ecriturePortSerie (int serialPort, std::string msg );
};
```

Question 14 : A partir de sa définition, donner la description UML de la classe **CDcGestionPortSerieStaticOnly**.

La méthode **ouverturePortSerie()** ouvre le port **/dev/ttyUSB0** et procède également, si l'ouverture est valide, à tout le paramétrage nécessaire de la communication série, c'est-à-dire à tous les réglages imposés par le protocole de communication bas niveau (8 bits de données, 1 bit de stop, pas de parité et 1 000 000 de bauds). Elle retourne dans ce cas un entier représentatif de ce port. Elle retourne -1 si l'ouverture, et donc in fine, le paramétrage, est impossible.

Question 15 : Ecrire la ligne de code permettant l'ouverture du port série sachant que la valeur retour de la fonction est placée dans la variable **hPortSerie** de type **int**. En déduire la ligne de code nécessaire à la fermeture de ce même port série en fin de programme.

La méthode principale de cette classe est la méthode **ecriturePortSerie()**. Elle attend la variable **hPortSerie** et le message à transmettre. Ce message est contenu dans un objet, instance de la classe **std::string**. Le code de cette méthode est donné ci-dessous.

```
#include <thread>
#include <unistd.h>

void CDcGestionPortSerieStaticOnly::ecriturePortSerie (int serialPort, std::string msg) {
   for (int i=0; i < int(msg.size()); i++) {
      write(serialPort, &(msg[i]), sizeof(msg[i]));
   }
   this_thread::sleep_for(chrono::milliseconds(20));
}
```

Cette portion de code utilise la fonction en langage C, **write()** dont le prototype est reproduit ci-dessous :

ssize_t write(int fd, const void *buf, size_t nbytes);

Question 16 : Indiquer quelle est la nature du second paramètre qui intervient dans l'utilisation de la fonction **write()** au sein de l'écriture de la méthode statique **ecriturePortSerie()**. Combien d'octets sont écrits sur le port USB de la carte **Raspberry Pi 3** à chaque passage dans la boucle for ?

3.3 *Gestion des servo-moteurs – Classe CDJServo*

La classe **CDJServo** permet la gestion de chacun des servo-moteurs de l'hexapode. C'est la dernière classe qui doit rester dépendante du type de servo-moteur utilisé. Il est donc important de conserver en mémoire cette information lors de l'écriture des méthodes de cette classe. Le changement de motorisation pour s'adapter à des nouvelles contraintes, par exemple de couple moteur, exigera ainsi la réécriture de cette classe si le protocole de communication de haut niveau évolue (passage du protocole 1 au protocole 2 de communication) ou si des commandes de pilotage changent d'une série à l'autre.

Cette classe est composée de quatre attributs privés et au minimum de trois méthodes publiques et d'une méthode privée ainsi que d'un constructeur inline qui doit renseigner les quatre attributs privés. Elle n'est pas destinée à devenir une classe mère et ne sera donc pas présente dans un héritage de classes. Sa description UML et son commentaire associé sont donnés sur la figure 13.

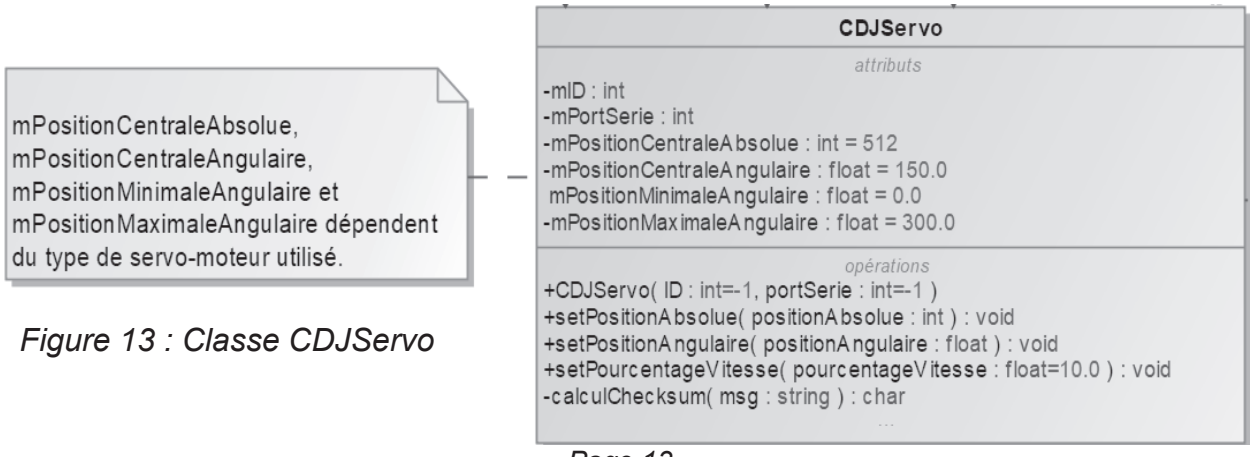


Figure 13 : Classe CDJServo

Tournez la page S.V.P.

Question 17 : A partir de la description UML, donner la définition de la classe **CDJServo** ainsi que le corps du constructeur **inline** sachant que la construction d'une instance d'un servo-moteur impose que la vitesse de ce dernier se situe à 15 % de sa vitesse maximale.

La méthode **calculChecksum()** doit permettre d'obtenir la même somme de contrôle que celle obtenue par le logiciel spécialisé de la société ROBOTIS. Elle requiert pour sa mise au point l'utilisation de la fonction **template** ci-dessous.

```
#include <iostream>
#include <bitset>

template <typename T>
std::string binary_repr (T value) {
   return std::bitset<sizeof(T)*8>(value).to_string();
}
```

Question 18 : Indiquer quel est le principe de fonctionnement d'une fonction **template** et ce que représente le mot clé **typename**. Quel est l'objectif de cette fonction **template** dans le cadre du développement de la fonction **calculChecksum()** ?

Afin de mettre au point la méthode **calculChecksum()** de la classe **CDJServo**, cette dernière est transformée en fonction et elle est testée séparément par un programme minimal qui reprend l'exemple donné sur le document technique DT2.

Le code de la fonction **calculChecksum()** et le code de la fonction principale permettant ce test sont donnés ci-dessous :

```
char calculChecksum (std::string s) {
   unsigned char total = 0;
   cout << "longueur de la chaine : "<<s.size()<< endl;
   for (int i =0; i<int(s.size()); i++) {
      cout << "Caractère courant : "<< binary_repr(char(s[i])) << endl;
      total += char(s[i]);
      cout << "Somme courante : "<< binary_repr(total) << endl;
      cout << "Somme courante (entier) : " << int(total) << endl;
      }
   total=~total;
   cout << "Somme finale complémentée : "<< binary_repr(total) << endl;
   cout << "Somme finale complémentée (entier) : "<< int(total) << endl;
   return total;
}
```

```
int main (void) {
   std::string chaine = {0x01, 0x05, 0x03, 0x0C, 0x64, 0xAA};
   char checksum = calculChecksum(chaine);
   cout << "checksum : "<< binary_repr(checksum) << endl;
   return 0;
}
```

Question 19 : Sachant que le contenu de la variable **total** est initialisé avec la valeur 0, compléter sur le document réponse DR2, la trace de la fonction **calculChecksum()** et vérifier que la fonction fournit bien la valeur attendue.

La méthode **setPositionAbsolue()** de la classe **CDJServo** permet de générer la trame nécessaire à la mise en position du servo-moteur. Elle est écrite en *couche basse*, c'est-à-dire que son écriture requiert la connaissance exacte du protocole de communication et du mot de commande nécessaire à la mise en position du servo-moteur. Elle utilise bien entendu la méthode **calculChecksum()** de cette même classe.

La méthode **setPositionAngulaire()** est écrite en *couche haute* et permet la mise en position du servo-moteur à partir de l'appel de la méthode **setPositionAbsolue()**. Elle doit vérifier que la position angulaire reste dans les limites acceptables. Dans le cas contraire, le servo-moteur est positionné en position centrale. Les ordres sont codés sur 10 bits et le débattement est de 300 degrés. La valeur 512 correspond à la position centrale.

Question 20 : Sur le document réponse DR3, compléter la méthode **setPositionAngulaire()**.

3.4 *Interface IDJVerification pour la classe CDJServo*

Pour permettre une vérification du bon fonctionnement de la motorisation de l'hexapode, la classe **CDJServo** implémente toutes les méthodes de la classe interface **IDJVerification** dont elle dérive.

L'interface **IDJVerification** se compose uniquement des deux méthodes virtuelles pures publiques **verificationIdentification()** et **verificationMouvement()** qui n'attendent aucun argument et ne retournent aucun argument.

Question 21 : Donner la définition de la classe interface **IDJVerification**. Proposer une modification de la représentation UML de la classe **CDJServo** de la figure 13 pour prendre en compte cette dérivation d'interface.

L'implémentation de la méthode **verificationMouvement()** dans la classe **CDJServo** doit permettre la mise en rotation du servo-moteur sur un débattement de +20° et -20° autour de la position centrale. Ces angles sont suffisamment grands pour que le déplacement soit clairement visible par un opérateur et suffisamment petits pour ne pas risquer de détériorer le servo-moteur en le plaçant en butée. Pour que l'opérateur ait le temps de visualiser correctement le fonctionnement du servo-moteur, le cahier des charges impose que la méthode génère deux mouvements complets et une pause de 1 seconde entre chaque déplacement.

Question 22 : Sachant que la pause peut être obtenue à partir de l'utilisation de la fonction **sleep()**, proposer à partir de l'ensemble des méthodes existantes, une implémentation de la méthode **verificationMouvement()** de la classe **CDJServo**.

3.5 *Gestion des pattes – Classe CDJPatte*

Chaque patte est composée de trois servo-moteurs comme représentée sur les figures 7 et 8. La classe **CDJPatte** reprend cette représentation physique et le diagramme UML de ces deux classes est représenté sur la figure 14.

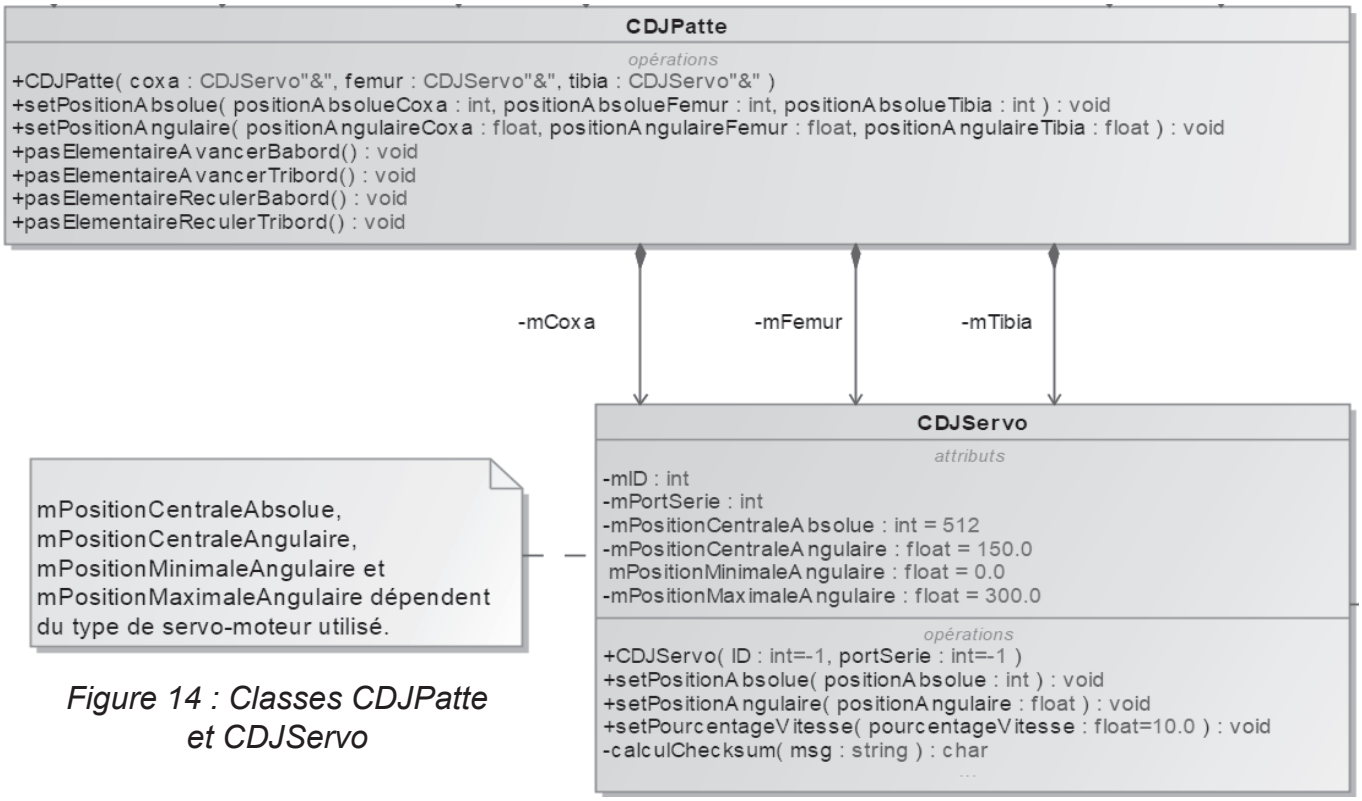


Figure 14 : Classes CDJPatte et CDJServo

Question 23 : Sachant que l'on souhaite piloter indépendamment chaque servo-moteur composant une patte ou piloter directement une patte de l'hexapode, justifier le passage par référence (copie à l'écriture) utilisé dans le constructeur de la classe **CDJPatte**. Quelle autre solution est-il possible d'utiliser ? Ecrire le corps de ce constructeur.

Le diagramme de séquence de la figure 15 présente le fonctionnement de la méthode **setPositionAbsolue()** de la classe **CDJPatte**.

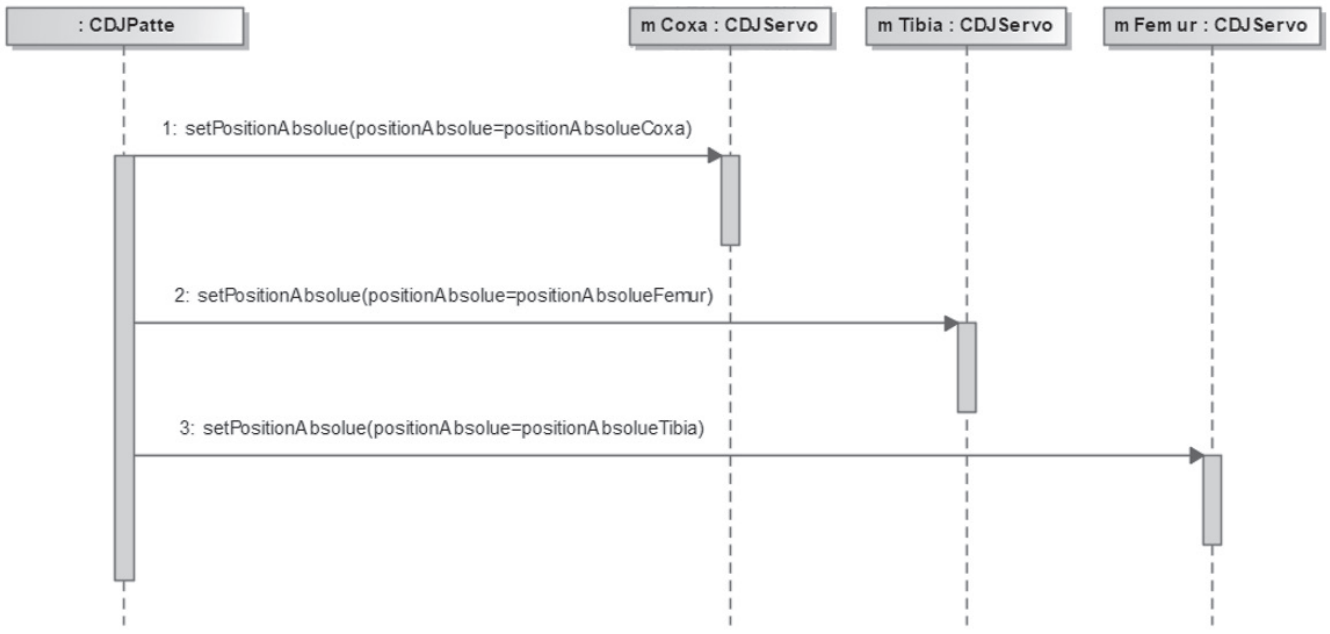


*Figure 15 : Diagramme de séquence de la méthode **setPositionAbsolue()** de la classe CDJPatte*

Question 24 : A partir du diagramme de séquence, établir le code nécessaire au codage de la méthode **setPositionAbsolue()** de la classe **CDJPatte.**

Question 25 : Sans écrire le code de la méthode **setPositionAngulaire()** de la classe **CDJPatte**, préciser quel est l'objectif de cette dernière.

3.6 *Classe CDJPatte – Interface IDJVerification*

La classe **CDJPatte** implémente également la classe interface **IDJVérification** et redéfinit les deux méthodes virtuelles pures déjà présentées.
La vérification du mouvement d'une patte consiste alors à vérifier le mouvement de toutes les articulations qui la composent dans l'ordre chronologique coxa, fémur et enfin tibia.

Question 26 : Proposer une implémentation de la méthode **verificationMouvement()** de la classe **CDJPatte**.

3.7 *Gestion de l'hexapode – Classe CDJHexapode*

De la même manière que la classe **CDJPatte**, la classe **CDJHexapode** est directement inspirée de la représentation physique de l'hexapode. Un objet de la classe **CDJHexapode** sera donc constitué de six pattes et le même choix de passage par référence (copie à l'écriture) lors de la construction de l'objet est mis en place. Le diagramme UML limité à ces deux classes est présenté sur le document technique DT4.

Un schéma cinématique inverse, non étudié, dans le cadre de ce sujet permet d'obtenir l'équilibre haut au repos de l'hexapode. Compte tenu de la disposition physique des servo-moteurs, toutes les valeurs angulaires obtenues sont différentes entre les pattes bâbord et les pattes tribord. On obtient ainsi dix-huit valeurs d'angles différentes pour cette position particulière.

Nous allons nous intéresser uniquement à la mise en position des pattes centrales. Le tableau ci-dessous indique les valeurs des différents angles obtenus par la résolution des équations du schéma cinématique inverse pour une hauteur de 18,5 centimètres en tenant compte de la position centrale des servo-moteurs.

Patte centrale bâbord		Patte centrale tribord	
Coxa	150 degrés	Coxa	150 degrés
Fémur	109 degrés	Fémur	191 degrés
Tibia	227 degrés	Tibia	73 degrés

Question 27 : Sur le document réponse DR4, compléter le corps de la méthode **equilibreHautRepos()** de la classe **CDJHexapode**.

3.8 *Classe CDJHexapode – Interface IDJVerification*

La classe **CDJHexapode** implémente elle aussi la classe interface **IDJVérification** et redéfinit les deux méthodes virtuelles pures déjà présentées.

La procédure de vérification se fait en retournant l'hexapode pour le positionner sur le dos ou en le soulevant du sol. Elle consiste à vérifier le mouvement de toutes les articulations pour toutes les pattes. Elle fait appel à la procédure mise en place pour vérifier le fonctionnement d'une seule patte.

Tournez la page S.V.P.

Question 28 : Ecrire la ligne de code de la méthode **verificationMouvement()** de la classe **CDJHexapode** permettant la vérification du bon fonctionnement de la patte **mPatteBabordProue** (patte de référence) de l'hexapode**.**

Question 29 : En négligeant la durée de tous les appels de fonctions, déterminer la durée effective de la procédure de vérification du fonctionnement des pattes de l'hexapode.

3.9 *Ouverture et conclusion*

On souhaite empêcher la création de plusieurs objets, instances de la classe **CDJHexapode** dans tout le programme.

Question 30 : Quel modèle de développement est-il possible d'utiliser pour le développement de la classe **CDJHexapode** afin d'empêcher la déclaration multiple d'objets, instances de cette classe ?

Question 31 : A partir de l'ensemble des informations disponibles, conclure sur la possibilité de piloter les déplacements du robot et sur la réalisation de l'exigence 1.2.

Partie 4. **Création du serveur Web**

Objectif : *Réalisation de l'exigence 1.4 : Mettre en place un serveur Web permettant la visualisation sur un simple navigateur du flux vidéo de la caméra embarquée sur l'hexapode.*

4.1 *Présentation*

Afin que l'opérateur puisse visualiser l'état de la vis depuis les regards et de manière plus générale l'ensemble des lieux à surveiller lors d'une mission, le robot ArIA dispose d'une caméra embarquée directement montée sur le port spécifique de la carte **Raspberry Pi 3**.

Le flux vidéo ainsi généré est destiné à être visualisé depuis une page Web qui contient également les commandes permettant de :

- démarrer l'affichage du flux vidéo ;
- stopper l'affichage du flux vidéo et le remplacer par une image par défaut ;
- obtenir une ou plusieurs photos et éventuellement les enregistrer sur le poste de travail par un simple clic droit.

Une description sommaire du module est donnée sur le document technique DT5.

4.2 *Mise en œuvre de la caméra et obtention du flux vidéo – Serveur Web Secondaire*

La mise en œuvre et l'arrêt de la caméra ainsi que toute la création et la gestion du flux vidéo est basée sur une solution libre nommée Mjpg Streamer. Cette solution permet de gérer automatiquement la création d'un serveur Web sur un port spécifique autorisant la prise et la diffusion d'images.

L'intégration de la gestion de cette solution dans le robot est réalisée à partir de l'écriture d'une classe **CDcGestionCameraStaticOnly** composée uniquement de deux méthodes statiques. Chaque méthode déclenche un script. Le premier permet de démarrer le flux vidéo en allumant la caméra alors que le second permet d'arrêter ce flux en éteignant cette dernière.

La classe avec ses deux méthodes inline est donnée ci-dessous :

```
class CDcGestionCameraStaticOnly {
public:
   static void cameraStart (void){
      system ("sh cameraStart.sh");
   }
   static void cameraStop (void){
      system ("sh cameraStop.sh");
      }
};
```

La méthode statique **CDcGestionCameraStaticOnly::cameraStart()** est mise en œuvre dès le lancement du programme de gestion de l'hexapode et un affichage des informations est renvoyé sur la console. Cet affichage est reproduit ci-dessous :

```
MJPG Streamer Version.: 2.0
```

Tournez la page S.V.P.

```
        i: fps.....: 5
        i: resolution.....: 640 x 480
        i: camera parameters.....:

Sharpness 0, Contrast 0, Brightness 50
Saturation 0, ISO 0, Video Stabilisation No, Exposure compensation 0
Exposure Mode 'auto', AWB Mode 'auto', Image Effect 'none'
Metering Mode 'average', Colour Effect Enabled No with U = 128, V = 128
Rotation 0, hflip No, vflip No
ROI x 0.000000, y 0.000000, w 1.000000 h 1.000000
        o: www-folder-path.....: ./www
        o: HTTP TCP port.....: 8080
        o: username:password .....:disabled
        o: commands.....: enabed
        i: Starting Camera
Encoder Buffer Size 81920
```

Question 32 : Quel est le port de communication utilisé par le serveur Web de la caméra embarquée, dit serveur Web secondaire, associé au logiciel de gestion Mjpg Streamer ?

4.3 *Serveur Web principal*

La carte **Raspberry Pi 3** intègre également un serveur **Apache**. Ce serveur, différent de celui de la gestion du flux de la caméra, est le serveur principal. Il est destiné à accueillir la page Web principale de gestion de l'hexapode. Il doit également permettre la communication avec le serveur Web secondaire afin de récupérer et afficher le flux vidéo avant d'autoriser la sauvegarde d'images.

4.4 *Page d'accueil*

Le serveur Web principal héberge la page d'accueil dont une représentation est donnée sur la figure 16.

Les fichiers partiels **index.html** et **styleCam.css** sont donnés respectivement sur les documents techniques DT6 et DT7.

Question 33 : Justifiez le choix du nom **index.html** comme fichier de page d'accueil.

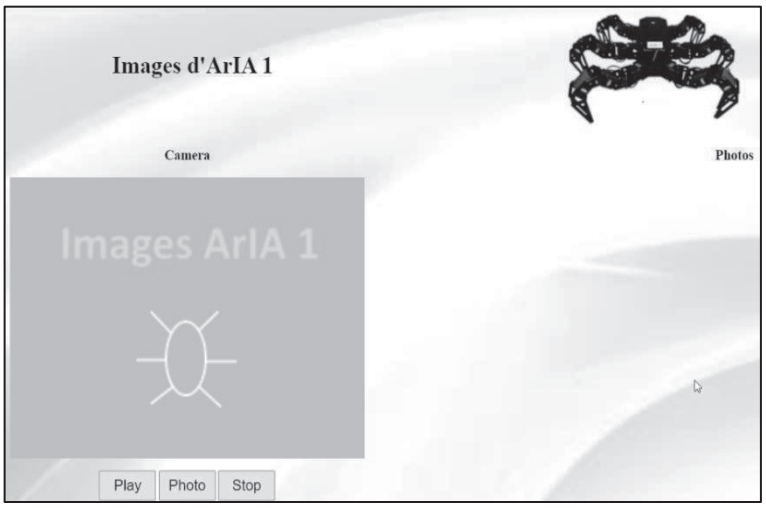


Figure 16 : Page Web d'accueil

Question 34 : Quelle est la différence entre les attributs « class » et « id » dans une balise HTML ? Comment sont-ils repérés dans le document CSS ?

4.5 *Modification du fichier CSS*

On souhaite que tous les éléments « enfants » de la balise « article » soient centrés verticalement dans leurs colonnes.

Question 35 : En vous appuyant sur le document technique DT8, réaliser la mise en forme CSS de la balise article.

On souhaite que l'image « enfant » de la balise « header » ait une largeur de 250 pixels.

Question 36 : Réaliser la mise en forme CSS de la balise « header ».

Afin que l'opérateur puisse visualiser cette page confortablement, aussi bien sur son écran d'ordinateur que sur sa tablette ou éventuellement son smartphone, les deux colonnes de la section doivent pouvoir se présenter l'une au-dessus de l'autre selon le type de média utilisé.

Question 37 : En vous appuyant sur le document technique DT9, réaliser la mise en forme CSS (responsive design) qui permet de faire passer le contenu de la balise « section » de deux colonnes à une seule colonne dès que la taille de l'écran est inférieure à 900 pixels. L'image d'ArIA intégrée à la balise « **header** » devra passer à une largeur de 180 pixels.

4.6 *Acquisition et traitement de l'image vidéo*

Le langage JavaScript est utilisé pour communiquer entre la page Web et le serveur du média caméra. Le script partiel **codeSource.js** est donné sur le document technique DT10.

Question 38 : Indiquer quelle est la fonction appelée lors d'un appui sur les boutons Play, Stop ou Photo de la page **index.html**.

Dans le script, certaines variables sont précédées du mot clé « const » et d'autres du mot clé « let ».

Question 39 : Rappeler quelles sont les différences entre ces deux types de déclaration.

La balise **<img>** repérée par l'id « idvideo » permet l'affichage du flux vidéo.

Question 40 : Ecrire la fonction **imageVideo()** qui remplace le flux vidéo par l'image par défaut de nom « accueil.png » disponible dans le sous-répertoire **images**.

Si le flux vidéo n'est pas affiché à l'écran, l'utilisateur ne doit pas pouvoir prendre de photo. Dans l'hypothèse où il appuierait tout de même sur la commande « Photo », un message situé dans une fenêtre popup doit l'informer que le flux vidéo n'est pas disponible.

Question 41 : Modifier, en conséquence, la fonction **prendrePhoto()**.

4.7 *Fonctionnement global des deux serveurs Web et conclusion*

On suppose désormais que le serveur principal sur la **Raspberry Pi 3** est à l'adresse 192.168.1.27. L'opérateur se connecte via un navigateur à cette adresse. Il commence donc par envoyer une requête http pour ouvrir le site Web. Il appuie sur les différents boutons disponibles pour activer le flux vidéo et sauver les photos qu'il désire conserver. Le serveur vidéo, dit serveur Secondaire se trouve à la même adresse.

Question 42 : Compléter, sur le document réponse DR5, le diagramme de séquence montrant les interactions client-serveurs. Conclure sur la possibilité de prendre des photos à partir de l'interface Web mise en place et in fine sur la réalisation de l'exigence 1.4.

Partie 5. **Pilotage de l'hexapode**

Objectif : Réalisation de l'exigence 1.3 : Mettre en place deux méthodes différentes de commande de l'hexapode afin de garantir une disponibilité maximale du système de pilotage.

5.1 ***Présentation***

Afin de faire fonctionner le robot dans toutes les circonstances, deux solutions de pilotage de l'hexapode co-existent.

Une première réalisée à partir d'une interface Web sur le serveur principal, une seconde basée sur l'utilisation d'un joystick. Chacune d'entre elles a un but différent.

- L'interface Web permet de piloter le robot si l'utilisateur est placé derrière son pupitre de commande. C'est la solution à privilégier.
- Le joystick permet de piloter le robot si l'utilisateur se trouve à proximité immédiate de ce dernier sans avoir nécessairement un retour caméra. Il permet également de reprendre la main sur le pilotage du robot en cas d'indisponibilité du réseau informatique.

5.2 ***Interface de secours – Pilotage par joystick USB sans fil***

Figure 17 : Manette de pilotage

Un joystick USB (figure 17) est directement relié sur un des ports USB disponible de la carte **Raspberry Pi 3** via un dongle de connexion sans fil. Ce joystick est vu par le système d'exploitation comme un fichier de nom **/dev/input/js0.**

La classe **CDcGestionJoystickStaticOnly** gère l'ouverture et la fermeture du joystick. Elle est composée uniquement de deux méthodes statiques différentes. Sa définition est proposée ci-dessous :

```
class CDcGestionJoystickStaticOnly {
public :
   static int ouvertureJoystick (void);
   static void fermetureJoystick (int joystick);
};
```

L'appui sur un bouton (ou le relâchement de ce dernier) ainsi que le déplacement des pads omnidirectionnels génèrent des évènements. Seuls sont pris en compte les événements de type bouton appuyé ou bouton relâché.

Les évènements sont gérés par la classe **CDcJoystickEvenement** composée uniquement des méthodes publiques **lectureEvenement()** et **getEvenement().** Cette classe ne possède qu'un seul attribut privé **mEvenement** de type structuré **js_event**. La totalité de la description de cette classe est donnée ci-dessous.

```
class CDcJoystickEvenement {
private :
   struct js_event mEvenement;

public :
   struct js_event & getEvenement (void);
```

```
    int lectureEvenement(int joystick);
};

struct js_event & CDcJoystickEvenement::getEvenement (void){
        return mEvenement;
}

int CDcJoystickEvenement::lectureEvenement(int joystick) {
    ssize_t bytes;
    bytes = read (joystick, this, sizeof(*this));
    if (bytes == sizeof(*this))
        return 0;
    return -1;          /* Error, could not read full event. */
}
```

La définition de la structure js_event est rappelée ci-dessous :

```
struct js_event {
__u32 time;         /* event timestamp in milliseconds */
__s16 value;        /* value */
__u8 type;          /* event type */
__u8 number;        /* axis/button number */
};
```

La portion de code de la méthode **lectureEvenement()** utilise la fonction en langage C, **read()** dont le prototype est reproduit ci-dessous :

ssize_t read(int fd, void *buf, size_t nbytes);

Question 43 : Indiquer ce que représente l'utilisation des termes **this** et **(*this)** dans l'utilisation de cette fonction au sein de la méthode **lectureEvenement()**. Quelle est la donnée mise à jour lors de l'appel de cette méthode ?

Une boucle **while** est mise en place pour scruter en permanence les événements joystick et agir en conséquence. Cette dernière doit permettre de déclencher les différentes actions à réaliser suivant une structure de sélection de type **switch**.

A chaque bouton est associée une action différente. Nous n'allons nous intéresser qu'aux actions de mise en position liées à l'équilibre de l'hexapode. Ces actions sont déclenchées par deux méthodes de la classe **CDJHexapode**.

Appui sur le bouton 4	equilibreHautMaximal ()
Appui sur le bouton 5	equilibreHautRepos ()

Question 44 : Sachant que l'objet créé se nomme hexapode et à l'aide de la description UML de la classe CDJHexapode disponible sur le document technique DT4, compléter sur le document réponse DR6 les lignes de code manquantes.

5.3 *Interface principale – Pilotage par interface Web de l'hexapode*

Le site sur le serveur Web principal est modifié afin de mettre en place l'interface de pilotage de l'hexapode.

L'opérateur dispose ainsi de cinq boutons sur son interface Web dont la disposition est très voisine de celle du pad du joystick. Cet ensemble est associé à un groupe de trois boutons supplémentaires. Ils permettent de modifier la position d'équilibre du robot. L'interface de pilotage est présentée sur la figure 18.

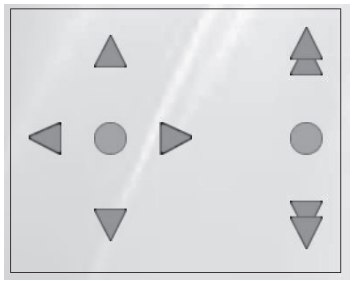


Figure 18 : Pad de pilotage

5.4 *Communication des informations entre l'interface Web et le programme de gestion de l'hexapode*

La méthode de communication retenue entre l'interface Web du serveur principal et le programme de gestion de l'hexapode est la mise en place d'une communication de type Client Serveur par **socket**. L'utilisation de cette solution doit ainsi permettre la communication d'informations unidirectionnelles entre la page Web via l'utilisation des langages **Javascript** et **PHP** et le programme principal de gestion de l'hexapode écrit en langage **CPP**. Les données transmises sont identiques à celles issues des appuis sur les boutons du joystick.

Le principe de fonctionnement peut alors être décrit de la manière suivante :

- Le programme de gestion de l'hexapode écrit en CPP crée un serveur sur le port 50000 nommé **serveurSocket50000**. Il se met alors en attente des ordres émis par un client qui se connecterait à ce serveur.
- L'opérateur manipule les boutons de l'interface Web. Le client PHP se connecte au **serveurSocket50000** pour transmettre l'information.
- Le serveur prend en compte cette information en déclenchant la ou les méthodes adéquate(s) de la classe **CDJHexapode** puis se remet en attente.

La figure 19 reprend le principe de cette communication.

Question 45 : Expliquer succinctement ce que représente un socket.

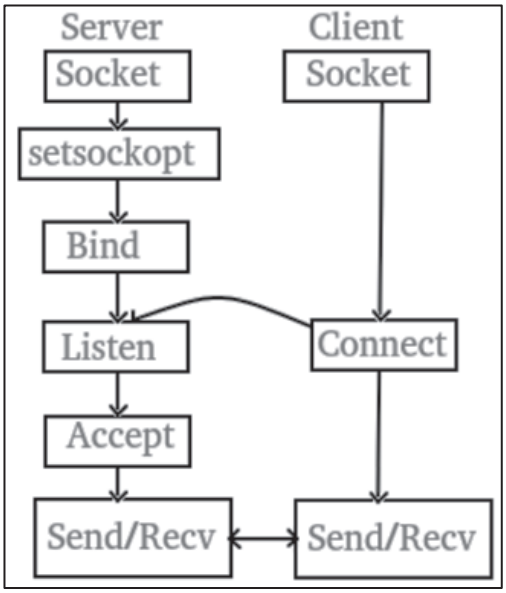


Figure 19 : Modèle Client-Serveur et communication par socket

5.5 *Fonctionnement côté Serveur*

Du côté serveur, la création de cette solution met en place les lignes de code proposées ci-dessous :

```
int serveurSocket50000, nouvelleSocket;
char bufferReception[8192];
struct sockaddr_in address;
int addrlen = sizeof(address);

serveurSocket50000 = socket(AF_INET, SOCK_STREAM, 0);

address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons (PORT);          // Avec PORT défini à 50000

bind(serveurSocket50000, (struct sockaddr *)&address,sizeof(address));
```

```
listen(serveurSocket50000, 5);
```

Question 46 : Quel est le rôle de la fonction **bind()** ?

5.6 *Fonctionnement détaillé côté client*

A chaque bouton disposé sur l'interface Web est associée une action différente. Pour cela, le fichier Javascript **codeSource.js** est modifié avec l'ajout de la prise en compte des appuis sur les différents boutons de l'interface.

Nous n'allons nous intéresser qu'aux actions de mise en position liées à l'équilibre de l'hexapode et donc qu'aux appuis sur les boutons de la partie de droite de l'interface. Sur cette partie, le bouton **Up** est lié à la variable **btnu** et le bouton **Stop** est lié à la variable **btns2**. Cette partie de l'interface est représentée sur la figure 20.

Figure 20 : Boutons Up et Stop

Pour chaque bouton, une fonction d'écoute est mise en place. Lors d'un appui sur un des boutons, la commande associée est créée. Elle consiste à déclencher la fonction **cmd()** avec la chaine de caractères associée à l'action demandée.

```
//Commande up
btnu.addEventListener("click", function() {
        cmd("up");
});

//Commande stopDroite
btns2.addEventListener("click", function() {
        cmd("stop");
});
```

La fonction **cmd()** par l'intermédiaire d'**Ajax** émet alors une requête **POST** en déclenchant le script **cmdSocket.php** sur le serveur principal hébergé par l'hexapode.

```
function cmd(chCmd) {
        const chstring = "cmd="+chCmd;
        $.ajax(
        {
                type : "post",
                url : "cmdSocket.php",
                data : chstring
        });
}
```

Le script **cmdSocket.php** émet enfin un message, via les sockets, à destination du serveur en CPP. Ce dernier, à l'écoute de ce type de message agit alors en conséquence en déclenchant la méthode adéquate de la classe **CDJHexapode**.

Le tableau ci-dessous résume de manière macroscopique le fonctionnement de cette partie de l'interface et de l'effet souhaité sur l'hexapode.

Tournez la page S.V.P.

Appui sur le bouton **Up** (btnu) 	Déclenchement de la méthode **equilibreHautMaximal()**
Appui sur le bouton **Stop** (btns2) 	Déclenchement de la méthode **equilibreHautRepos()**

Question 47 : A partir du script **cmdSocket.php** donné sur le document réponse DR7, indiquer quelle est l'adresse IP utilisée lors de la communication de type Client-Serveur par socket. Que représente-t-elle ?

Question 48 : Compléter sur le document réponse DR7, les lignes de code manquantes correspondant aux commandes à émettre dans le cas de la modification de l'équilibre de l'hexapode.

Question 49 : Compléter, sur le document réponse DR8, le diagramme de séquence montrant les interactions entre l'opérateur, le serveur Web principal, la transmission d'informations par socket et la prise en compte de la modification de l'équilibre de l'hexapode dans le cas de l'appui sur le bouton **Up** de l'interface Web.

5.7 *Conclusion sur les méthodes de pilotage*

Question 50 : Conclure sur les méthodes de pilotage mises en œuvre pour garantir le pilotage de l'hexapode et sur la réalisation des exigences 1.3.1 et 1.3.2. L'exigence 1.3 est-elle satisfaite ?

Partie 6. **Création et exploitation d'une base de données**

Objectif : *Réalisation des exigences 1.5 et 1.5.1 : Développer une base de données pour stocker les différentes informations des missions de surveillance.*

6.1 *Présentation*

Afin d'estimer la plus-value qu'apporterait la conservation d'un historique des différentes missions des robots de surveillance, le site d'exploitation de Malakoff décide de créer et d'exploiter une base de données relative à tous les moyens d'inspection et de surveillance, sous la forme d'un prototype rapidement obtenu. Dans l'hypothèse où cette fonctionnalité s'avérerait réellement pertinente, une solution plus pérenne serait développée dans un second temps.

Nous nous limitons donc ici à la partie relative à l'inspection des regards du site de Malakoff et nous n'utilisons volontairement que des outils de conception rapide.
Le choix se porte sur un outil de conception libre et gratuit disponible sur le Web, **Mocodo** (http://www.mocodo.net/) et sur le gestionnaire de base de données **DB Browser for SQLite**, logiciel libre et Open Source (disponible sur tous les OS courants). L'ensemble sera réalisé sur une machine Linux équipée d'un OS **Ubuntu**.

6.2 *Installation de l'outil DB Browser for SQLite*

L'outil retenu n'est pas présent de manière native sur la machine Linux équipée de l'OS Ubuntu. Une recherche rapide permet cependant de déterminer qu'il est disponible dans un dépôt de fichiers.

Dans un terminal, on tape successivement les trois lignes de commande ci-dessous :

```
sudo add-apt-repository -y ppa:linuxgndu/sqlitebrowser
sudo apt-get update
sudo apt-get install sqlitebrowser -y
```

Question 51 : Indiquer avec précision les actions réalisées par ces trois lignes de commande.

Pour l'utilisation future de cet outil, il faut rendre accessible un complément qui n'est pas installé de manière native. Pour cela, nous avons besoin de télécharger un fichier supplémentaire et de le compiler.
Ce fichier sera téléchargé dans le répertoire **pourSQLite**, sous-répertoire du répertoire courant.

Question 52 : Proposer la ligne de commande qui permet de créer le répertoire **pourSQLite** en donnant toutes les permissions (lecture, écriture, exécution) pour tous les utilisateurs. Indiquer la ligne de commande qui permet de vérifier ces permissions et enfin indiquer la ligne de commande qui permet de se placer dans ce répertoire.

Le téléchargement est réalisé grâce à la commande wget. Le lien de téléchargement est https://www.sqlite.org/src/file/ext/misc/ et le fichier à télécharger se nomme **fileio.c**.

Question 53 : Donner la ligne de commande qui permet de télécharger ce fichier. Quelle autre commande pouvons-nous utiliser ?

Après quelques petites manipulations sur le fichier téléchargé, consistant principalement à régler des problèmes de codage de caractères et de suppression d'entête, ce dernier peut être compilé.

Tournez la page S.V.P.

La compilation requiert l'utilisation de la bibliothèque **libsqlite3-dev** qu'il faut désormais installer (disponible dans le même dépôt que celui qui contient les paquets relatifs à SQLite).

Question 54 : Donner la ligne de commande qui permet de télécharger et d'installer cette bibliothèque.

La compilation se fait grâce à la commande suivante :

```
gcc -g -fPIC -shared fileio.c -o fileio.so
```

Question 55 : Quel est le type de fichier créé par cette étape de compilation ?

6.3 *Conception de la base de données - Schéma Entités - Associations*

Le schéma Entités - Associations retenu se présente sous la forme présentée sur la figure 21.

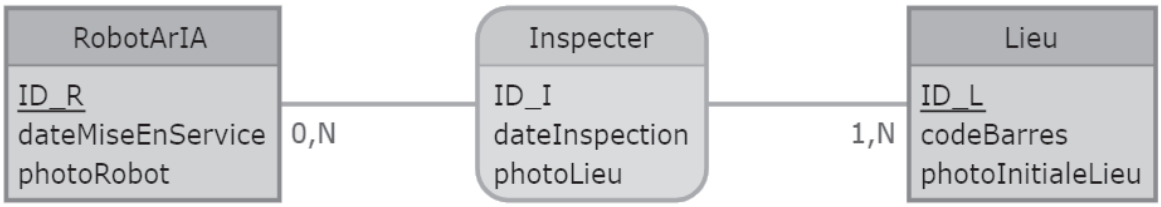


Figure 21 : Schéma Entités - Associations

Question 56 : Indiquer ce que signifie le nombre 0 dans la cardinalité 0, N proche de l'entité RobotArIA.

L'outil Mocodo permet d'obtenir directement les requêtes SQL permettant de générer les tables de la base de données sous plusieurs SGBD ; en particulier SQLite. Suivant le paramétrage, il est parfois nécessaire de modifier ces requêtes afin de s'adapter au contexte souhaité.

Pour la création des tables, les contraintes suivantes sont mises en place :

- les clés primaires et étrangères sont représentées par des nombres entiers (INTEGER) ;
  - la clé primaire de la table INSPECTER est ID_I.
- le code-barres est représenté par un nombre entier (INTEGER) ;
- les photos sont stockées sous forme de fichiers BLOB ;
- les dates sont stockées sous forme de nombre « réel » (REAL) et elles seront gérées grâce au calendrier Julien.

Les requêtes générées automatiquement sont reproduites sur le document réponse DR9.

Question 57 : Indiquer pour quelle raison, le couple (ID_R ; ID_L) ne peut pas rester la clé primaire de la table INSPECTER.

Question 58 : Indiquer ce que signifie le terme BLOB et quels sont les avantages et inconvénients à utiliser ce type de fichier dans une base de données. Quelle autre solution serait-il possible de mettre en place si on ne souhaitait pas utiliser ce type de solution ?

Question 59 : Sur le document réponse DR9, modifier la requête de création de la table INSPECTER pour prendre en compte les contraintes sur la clé primaire de cette table.

La base de données de nom ***malakoff.db*** est créée à partir de l'outil DB Browser for SQLite et on obtient ainsi les trois tables qui la composent. Ces tables seront renseignées au fur et à mesure

des mises en service des nouveaux matériels d'inspection, des lieux à inspecter et des missions réalisées.

6.4 *Exploitation de la table LIEU*

Dans un premier temps, les lieux à surveiller sont limités aux différents regards du site de Malakoff.

Question 60 : Ecrire en langage SQL, la requête donnée en algèbre relationnel : $\pi_{codeBarres}\left(\sigma_{ID\_L=106}(LIEU)\right)$.
Quel est l'objectif de cette requête ? Que retourne-t-elle ?

Question 61 : En utilisant l'alias "Nombre de regards" pour l'affichage du résultat, établir la requête en langage SQL permettant d'obtenir le nombre de regards actuellement sous surveillance sur le site de Malakoff. Quelle valeur doit renvoyer cette requête ?

6.5 *Renseignement de la table ROBOTARIA*

Le 14 Juin 2018, le premier robot ARIA (ID_R = 001) est mis en service sur le site mais aucune photo de lui n'est placée dans son enregistrement.

Question 62 : Sachant que les dates sont gérées grâce au calendrier Julien, dont une présentation succincte est proposée sur le document technique DT11, établir en langage SQL, la requête d'insertion dans la table ROBOTARIA du premier robot ARIA.

Afin d'identifier plus facilement le robot, une photo de lui est prise et il est décidé de mettre à jour l'enregistrement précédent sans modifier la date d'enregistrement. Pour cela, la photo du robot est placée dans le répertoire courant sous le nom **ARiA001.png**.
De plus, pour la manipulation des fichiers de type BLOB, nous ferons l'hypothèse que le module **fileio** est préalablement chargé dans l'outil DB Browser.

Question 63 : A l'aide du document technique DT12, établir la requête SQL permettant la mise à jour de l'enregistrement précédent avec la photo du robot.

6.6 *Exploitation de la table INSPECTER*

L'inspection de chaque regard est enregistrée dans la table INSPECTER. Chaque enregistrement consiste à noter le numéro du regard, l'identification du robot de surveillance et la date de la mission avec l'heure. La photo du regard n'est pas nécessaire à chaque enregistrement.

Question 64 : Etablir la requête SQL permettant d'enregistrer la mission de surveillance du regard 106 par le robot 001, le 18 Juin 2018 à 16 heures et 2 minutes exactement. Cette inspection porte le numéro 63 et aucune photo n'est prise.

Question 65 : Donner la requête permettant d'obtenir les dates et heures des missions de surveillance, les identificateurs des robots et les numéros d'identification des missions pour le regard d'identification 106 classés par ordre décroissant de date.

Question 66 : Conclure sur les choix réalisés lors de la réalisation de la base de données et sur la réalisation de l'exigence 1.5.

Tournez la page S.V.P.

Partie 7. **Interface de gestion de la base de données**

Objectif : Réalisation des exigences 1.5.2 : Développer un premier programme en mode Console pour questionner la base de données.

Une fois les requêtes validées, il apparait à l'usage que la gestion de la base de données doit être menée par un personnel qualifié en SQL, ce qui conduit inévitablement à des erreurs ou des oublis d'enregistrement en l'absence d'un programme simple de gestion de cette dernière.

Un programme est réalisé en langage Python et seules quelques fonctions feront l'objet de cette étude. En particulier l'aspect graphique n'est pas traité. Nous ferons l'hypothèse que toutes les bibliothèques nécessaires sont correctement importées et en particulier que les trois imports ci-dessous sont réalisés.

```
import sqlite3 # Import de la bibliothèque sqlite3
from PIL import Image as im
import os
```

7.1 *Ouverture de la base de données*

La fonction **ouvertureBase(base)** permet de charger la base de données ***malakoff.db*** dont le nom est passé en paramètre. Elle est donnée ci-dessous en langage Python et elle est adaptée aux deux environnements (Windows et Linux).

```
def ouvertureBase (nomBase) :
    connexion = sqlite3.connect(nomBase)
    connexion.enable_load_extension(True)                    # 1

    if os.name=="nt": # Test Windows ou Linux
        #print ("sous Windows")
        connexion.load_extension("fileio.dll")               # 2a
    else :
        #print("sous Linux")
        connexion.load_extension("/home/XX/fileio.so")       # 2b

    connexion.text_factory = str
    curseur = connexion.cursor() # Création du manipulateur sur les tables
    return curseur
```

Question 67 : Indiquer quels sont les rôles respectifs des lignes de code 1 et 2a ou des lignes de code 1 et 2b.

7.2 *Exécution d'une requête sur la base de données et affichage des résultats en mode Console*

Le code en langage Python de la fonction **executionRequeteSQL()** est donnée ci-dessous :

```
def executionRequeteSQL (curseur, requeteSQL) :
# Exécution de la requête
    curseur.execute(requeteSQL)
```

```
    resultatRequete = curseur.fetchall()

    return resultatRequete
```

Question 68 : Sachant que le code partiel de la fonction **affichageModeConsole()** donnée sur le document réponse DR10 affiche les résultats présentés dans le cadre de droite, compléter en Python sur ce document réponse les lignes de code manquantes afin de faire afficher le contenu des informations textuelles issues de la requête. Cette fonction est-elle adaptée à l'affichage des données de type BLOB ?

Dans une première approche de l'exploitation de la base de données, on cherche à obtenir une seule photo à la fois. On peut par exemple souhaiter obtenir la photo d'un robot ou la photo d'un lieu. Les requêtes SQL dans cette partie ne doivent ainsi renvoyer qu'une seule photo qui sera stockée dans une image temporaire nommée ***'imageTEMP.png'***. Nous allons chercher à écrire le code de la fonction **affichageModeConsolePhotoUnique()**.

Le code préparatoire à l'appel de cette fonction dans le cas de l'affichage du Robot Aria d'Identifiant 001 est le suivant :

```
# Suppression du fichier temporaire et appel de la fonction pour afficher le fichier
# imageTEMP.png s'il existe.
if os.path.exists('imageTEMP.png'):
    os.remove('imageTEMP.png')

requeteSQL="SELECT writefile('imageTEMP.png',photoRobot) FROM ROBOTARIA WHERE ID_R = 001;"
resultatRequete = executionRequeteSQL(curseur, requeteSQL)
affichageModeConsolePhotoUnique('imageTEMP.png')
```

Question 69 : Compléter en langage Python, sur le document réponse DR11, le code de la fonction **affichageModeConsolePhotoUnique()**. Cette fonction doit afficher une image d'erreur nommée ***'notFound.png'*** (figure 22), disponible dans le répertoire courant si le fichier ***'imageTEMP.png'*** n'existe pas. Justifier l'utilisation des exceptions dans le cadre de l'écriture de cette fonction.

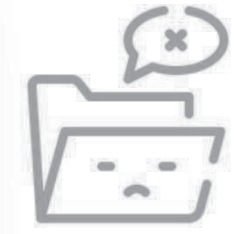


Figure 22 : notFound.png

7.3 *Conclusion*

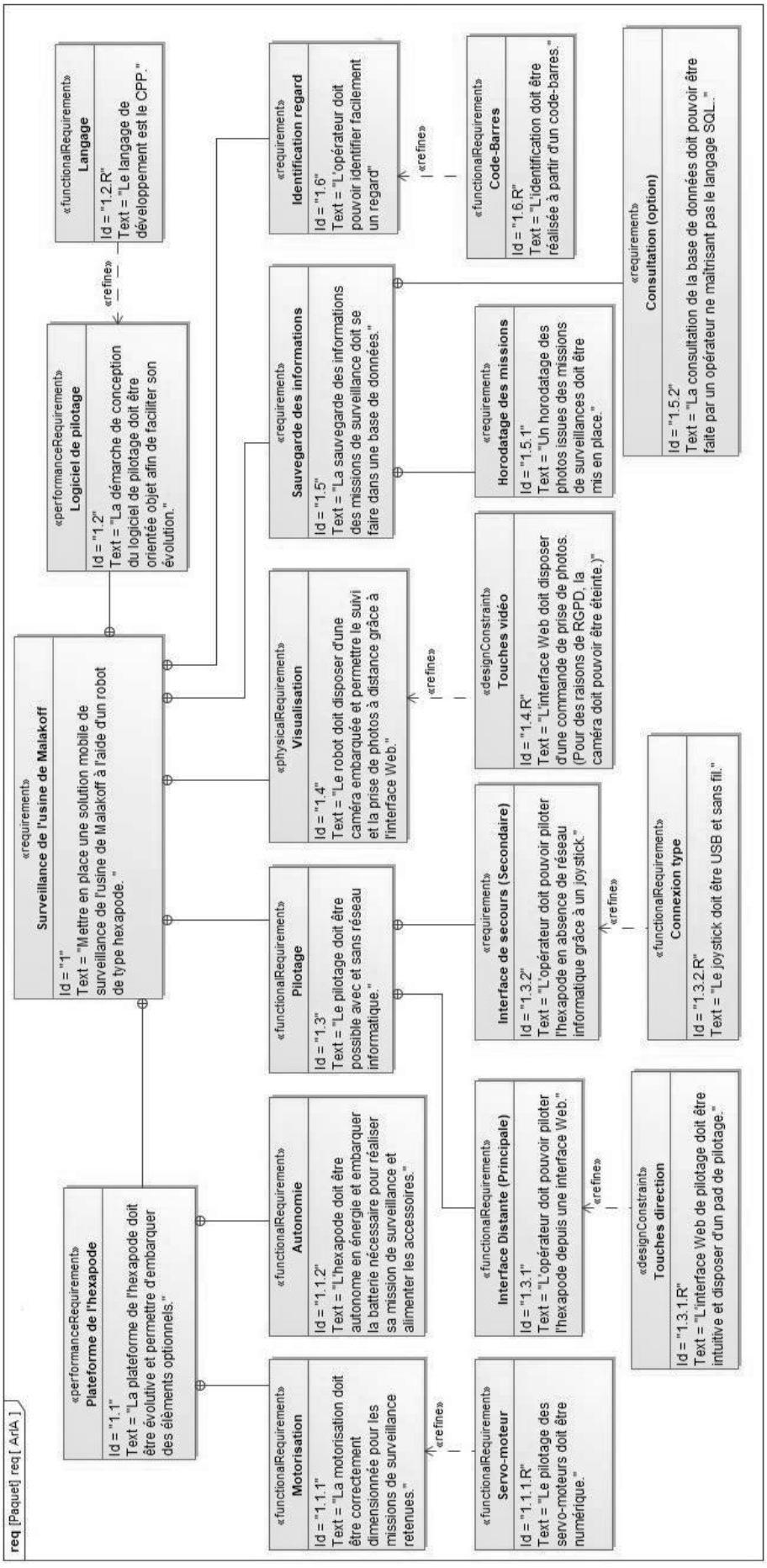
Question 70 : Conclure sur l'apport de la gestion de la base de données et sur la réalisation de l'exigence 1.5.2. Quels sont les développements futurs à mettre en place pour pérenniser cette solution ?

Synthèse sur l'étude – Conclusion générale

Question 71 : En reprenant les différents points abordés, énoncer en quelques lignes, une synthèse des grandes étapes du projet et des limites des solutions proposées. Quels sont les points qui restent à améliorer ?

DOSSIER TECHNIQUE

DT1 – Diagramme partiel des exigences



DT2 – Spécification et protocole utilisés par les servo-moteurs – Extrait

Specification
DYNAMIXEL is a robot exclusive smart actuator with fully integrated DC Motor + Reduction Gearhead + Controller + Driver + Network in one DC servo module.

Input Voltage	Min. [V]	9.0
	Recommended [V]	11.1
	Max. [V]	12.0
Performance Characteristics	Voltage [V]	12.0
	Stall Torque [N·m]	1.50
	Stall Current [A]	1.5
	No Load Speed [rpm]	59.0
	No Load Current [A]	0.14
Resolution	Resolution [deg/pulse]	0.2930
	Step [pulse]	1,024
	Angle [degree]	300
Position Sensor		Potentiometer
Gear Type		Spur
Gear Ratio		254 : 1

Compatible Products
Controller : CM-5, CM-510, CM530, CM-700, OpenCM9.04(+ OpenCM485 Expansion Board), OpenCR
Interface : USB2Dynamixel, U2D2

Factory Default Settings
ID : 1
Baud Rate : 1Mbps
(User can change various settings including ID and baud rate according to environment.)

Tournez la page S.V.P.

Command Signal	Digital Packet
Protocol Type	Half duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)
Link (Physical)	TTL Level Multi Drop Bus
ID	0 ~ 253
Feedback	Position, Temperature, Load, Input Voltage, etc
Protocol version	Protocol 1.0
Operating Mode / Angle	Wheel Mode : Endless turn Joint Mode : 300 [deg]

Packet
Main Controller and DYNAMIXEL communicate each other by sending and receiving data called Packet. Packet has two kinds: Instruction Packet, which Main Controller sends to control DYNAMIXEL, and Status Packet, which DYNAMIXEL responses to Main Controller.

ID
ID is a specific number for distinction of each DYNAMIXEL when several DYNAMIXEL's are linked to one bus. By giving IDs to Instruction and Status Packets, Main Controller can control only DYNAMIXEL that you want to control

Protocol
DYNAMIXEL does the Asynchronous Serial Communication with 8 bit, 1 Stop bit, and None Parity.

If DYNAMIXEL with the same ID is connected, packet will collide and network problem will occur. Thus, set ID as such that there is no DYNAMIXEL with the same ID.
ID of DYNAMIXEL is changeable. For this change, please refer to Changing IDs of DYNAMIXEL.
The factory default setting ID is 1.

Communication Overview
To control DYNAMIXEL, communication should be established according to the protocol of DYNAMIXEL. DYNAMIXEL is driven by receiving binary data.

Instruction Packet
Instruction Packet is the command data sent to the Device.

Header 1	Header 2	Packet ID	Length	Instruction	Param 1	…	Param N	Checksum
0xFF	0xFF	Packet ID	Length	Instruction	Param 1	…	Param N	CHKSUM

Packet ID
The field that indicates the ID of the Device that should receive the Instruction Packet and process it

- Range : 0 ~ 253 (0x00 ~ 0xFD), which is a total of 254 numbers that can be used.
- Broadcast ID : 254 (0xFE), which makes all connected devices execute the Instruction Packet.

Length
The length of the Packet (Instruction, Parameter, Checksum fields). Length = number of Parameters + 2.

Instruction
The field that defines the type of instruction.

Value	Instructions	Description
0x01	Ping	Instruction that checks whether the Packet has arrived to a device with the same ID as Packet ID
0x02	Read	Instruction to read data from the Device
0x03	Write	Instruction to write data on the Device
0x04	Reg Write	Instruction that registers the Instruction Packet to a standby status; Packet is later executed through the Action instruction
0x05	Action	Instruction that executes the Packet that was registered beforehand using Reg Write
0x06	Factory Reset	Instruction that resets the Control Table to its initial factory default settings
0x08	Reboot	Instruction that reboots DYNAMIXEL(See applied products in the description)
0x83	Sync Write	For multiple devices, Instruction to write data on the same Address with the same length at once
0x92	Bulk Read	For multiple devices, Instruction to write data on different Addresses with different lengths at once This command can only be used with MX series.

Parameters
Parameters are used when additional data is required for an instruction.

Instruction Checksum
It is used to check if packet is damaged during communication. Instruction Checksum is calculated according to the following formula.
Instruction Checksum = ~(ID + Length + Instruction + Parameter1 + ... + ParameterN)

Where "**~**" is the Binary Ones Complement operator. When the calculation result of the parenthesis in the above formula is larger than 255 (0xFF), use only lower bytes.

For example, when you want to use below Instruction Packet,

ID=1 [0x01], Length=5 [0x05], Instruction=3 [0x03], Parameter1=12 [0x0C],
Parameter2=100 [0x64], Parameter3=170 [0xAA]

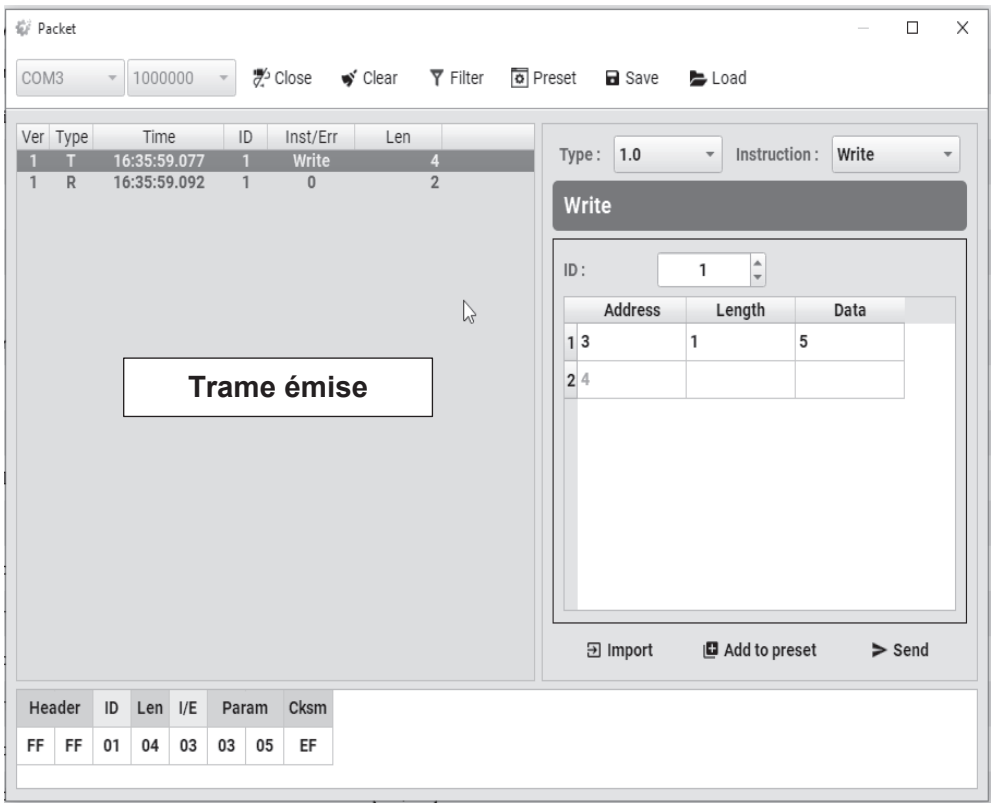
Checksum = **~** (ID + Length + Instruction + Parameter1 + ... + Parameter3) = **~** [0x01 + 0x05 + 0x03 + 0x0C + 0x64 + 0xAA] = **~** [0x123]
// Only the lower byte 0x23 executes the Not operation → 0xDC

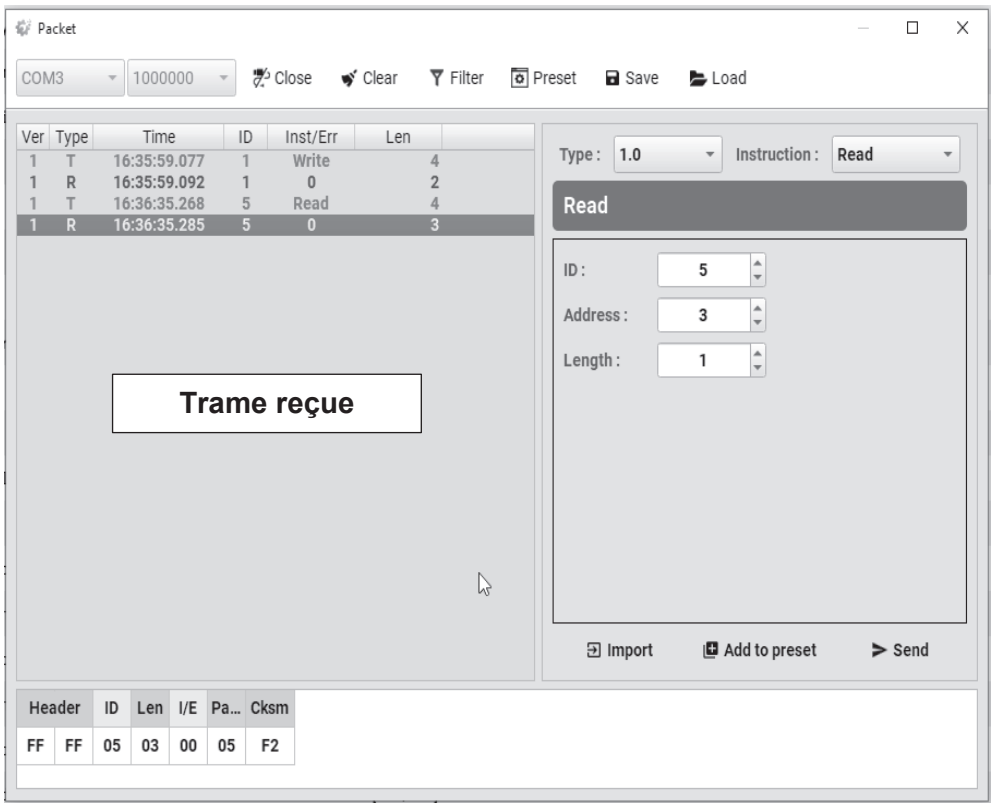
Thus, Instruction Packet should be 0xFF, 0xFF, 0x01, 0x05, 0x03, 0x0C, 0x64, 0xAA, 0xDC.

Tournez la page S.V.P.

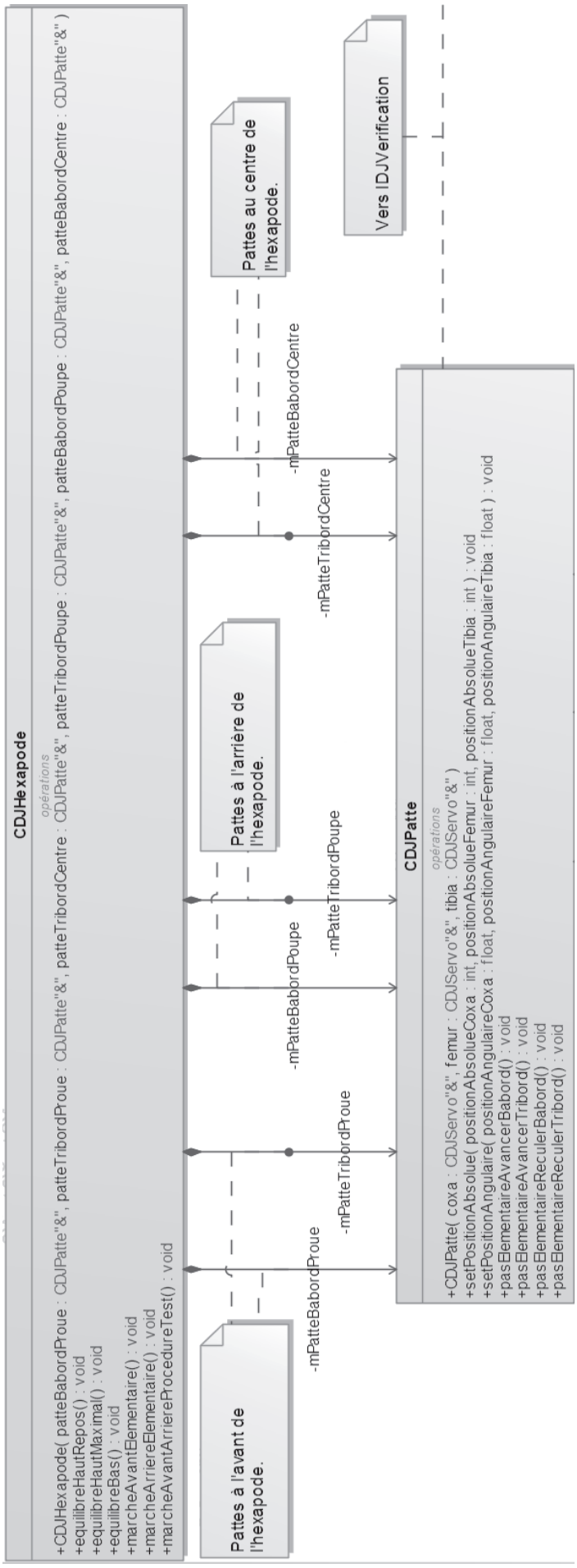
DT3 – Copies d'écran des trames de paramétrage des servo-moteurs

Logiciel Dynamixel Wizard 2.0





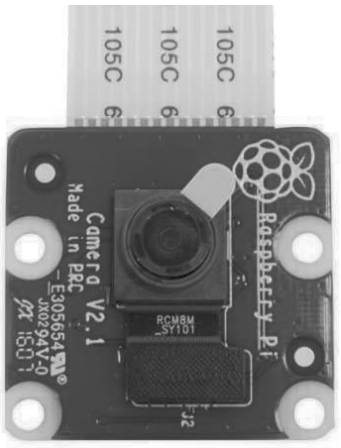
DT4 – Classes CDJHexapode et CDJPatte



Tournez la page S.V.P.

DT5 – Module caméra de la Raspberry Pi 3 – Extrait

Le module caméra haute définition (HD) de carte la Raspberry Pi 3 est à canal unique, 8 mégapixels et prend en charge l'interface de bus CSI-2.



Il est doté d'une capture de fréquence d'image maximale de 30 images par seconde. Il utilise le capteur d'image IMX219PQ de Sony, qui offre une imagerie vidéo haute vitesse et une haute sensibilité. Il offre également une réduction de la contamination de l'image, telle que le bruit de motif fixe et les taches.

Caractéristiques et avantages

- Câble en nappe 15 broches.
- Résolution d'image fixe de 3280 x 2464 pixels.
- Détection automatique de luminance 50 / 60 Hz.
- Etalonnage automatique du niveau noir.
- Commande d'exposition automatique, balance des blancs et filtre de bande.
- Haute capacité de données.
- Images de haute qualité.
- Plage de températures d'utilisation comprise entre -20 et 60 °C.
- Prise en charge des formats 1080p30, 720p60 et VGA90 (640x480p90).

DT6 – Fichier partiel « index.html »

```
<body>
    <header class="titre">
        <h1>Images d'ArIA 1</h1>
        <img src="images/ArIA.png" alt="Robot ArIA">
    </header>
    <section>
        <article>
            <h1>Camera</h1>
            <img id="idvideo" alt="la vidéo" src="images/accueil.png">
            <p>
                <input type="button" id="btnPlay" value="Play">
                <input type="button" id="btnPhoto" value="Photo">
                <input type="button" id="btnStop" value="Stop">
            </p>
            <table id="btn_cmd">
                <tr>
                    <td></td>
                    <td><img src="images/btnf.png" alt="bouton" id="btn"></td>
                    <td></td>
                </tr>
                <tr>
                    <td><img src="images/btnl.png" alt="bouton" id="btnl"></td>
                    <td><img src="images/btns.png" alt="bouton" id="btns"></td>
                    <td><img src="images/btnr.png" alt="bouton" id="btnr"></td>
                </tr>
                <tr>
                    <td></td>
                    <td><img src="images/btnb.png" alt="bouton" id="btnb"></td>
                    <td></td>
                </tr>
            </table>
        </article>
        <article id="idphoto">
            <h1>Photos</h1>
        </article>
    </section>
    …
    …
</body>
```


Tournez la page S.V.P.

DT7 – Fichier partiel « styleCam.css »

```
body {
        background-image : url("images/fond.jpg");
        background-size: cover;
        font-family : 'cantarelloblique';
}

.titre {
        display : flex;
        align-items : center;
        justify-content : space-evenly;
}

section {
        display : grid;
        grid-template-columns : 60% 30%;
        grid-gap : 10px;
        padding : 10px;
}

#idvideo {
        width : 500px;
        height : 375px;
        background-color : #B0B0B0;
}
```

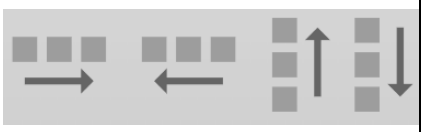
```
table {
        width : 250px;
        height : 200px;
        border : 1px solid black;
        vertical-align : middle;
        text-align : center;
}

td {
        text-align : center;
        width : 20%;
        height : 33.33%;
}

td>img:hover {
        width : 30px;
}

td>img:active {
        transform : translateY(4px);
}

input {
        width : 80px;
        height : 40px;
        font-size : 20px;
        vertical-align : middle
}
```

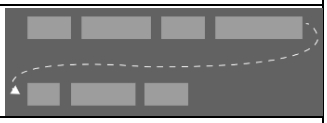
DT8 – Module flexbox – Extrait

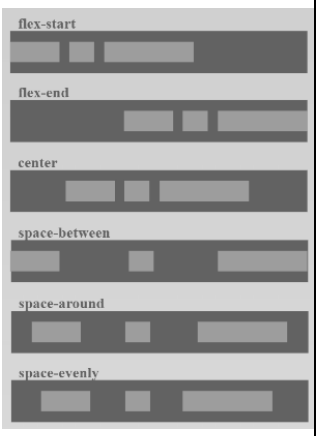
Présentation
« Flexible box » permet d'agencer une page Web. Le principe consiste à disposer des boîtes dans des conteneurs. Le module permet de :

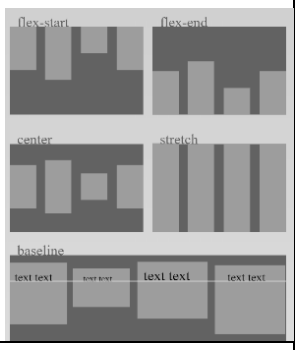
- distribuer horizontalement et verticalement les éléments d'un conteneur, avec passage à la ligne ;
- aligner et justifier les espaces entre les éléments ;
- ordonner les éléments ;
- gérer la taille des espaces et des éléments.

Propriété	Valeurs (exemples)	Description
display	flex	Pour définir le conteneur
flex-direction	row	De gauche à droite
	row-reverse	De droite à gauche
	column	De haut en bas
	column-reverse	De bas en haut
flex-wrap	wrap	Retour automatique à la ligne
justify-content	flex-start	Aligner au début (axe principal)
	flex-end	Aligner à la fin
	center	Aligner au centre
	space-between	Répartir sur l'axe en jouant sur les espaces intermédiaires,
	space-around	en jouant sur tous les espaces (plus grand inter),
	space-evenly	en jouant proportionnellement sur tous les espaces
align-items	flex-start	Aligner au début (axe secondaire)
	flex-end	Aligner à la fin
	center	Aligner au centre
	stretch	Répartir sur l'axe
	baseline	Aligner sur une ligne de base
order	1 ou 2 ou 3…	Ordonne les éléments
flex	1 ou 2 ou 3…	Grossir un élément sur l'espace 1x, 2x, 3x









Tournez la page S.V.P.

DT9 – Responsive design – Extrait

Présentation
Il existe aujourd'hui de nombreux moyens de naviguer sur le Web et on ne peut donc plus envisager de réaliser un site uniquement pour les utilisateurs de PC de bureau.

Il devient ainsi nécessaire, que les pages Web s'adaptent d'elles-mêmes aux différents formats d'affichage.

Le fichier CSS doit intégrer le code qui permet de réduire le format des images, des polices de caractères, d'éliminer des parties d'une page ou encore d'en changer la disposition, lorsque la zone d'affichage devient inférieure à un format donné.

Les media queries sont les propriétés qui permettent l'usage de styles différents.

Exemple d'utilisation :

```
@media screen and (max-width : 640px)
{

}
```

Syntaxes	Commentaires
@media (max-width : 640px)	Largeur de la fenêtre inférieure à 640 px
@media (max-device-width : 640px)	Largeur du périphérique inférieure à 640 px
@media (orientation : landscape)	Périphérique tenu horizontalement

Il est également possible d'associer des conditions avec les mots-clés « and, only, non, or ».

DT10 – Fichier partiel « codeSource.js »

```
function videoCamera() {
    imgVideo.src = lienVideo;
}

// Prise de photo, affichage et enregistrement
function prendrePhoto() {
    const canvasPhoto = document.createElement("canvas");
    const ctx = canvasPhoto.getContext('2d');
    const newText = document.createElement("p");
    newText.innerHTML = "Photo "+cptPhotos;
    photo.appendChild(canvasPhoto);
    ctx.drawImage(imgVideo, 0, 0, 250, 150);
    photo.appendChild(newText);
    cptPhotos +=1;
}

// Eléments présents sur la page
const urlCourante = document.location.href.replace(/\/$/, "");

const imgVideo = document.querySelector("#idvideo");
const btnPlay = document.querySelector("#btnPlay");
const btnPhoto = document.querySelector("#btnPhoto");
const btnStop = document.querySelector("#btnStop");
let cptPhotos = 1; // pour numéroter les photos
const lienVideo = urlCourante+":8080?action=stream";
const lienPhoto = urlCourante+":8080?action=snapshot";

// play
btnPlay.addEventListener("click", function() {
    videoCamera();
});

// photo
btnPhoto.addEventListener("click", function() {
    prendrePhoto();
});

// stop
btnStop.addEventListener("click", function() {
    imageVideo();
});
```


Tournez la page S.V.P.

DT11 – Gestion du calendrier Julien en SQLite – Extrait

Description
The SQLite julianday function takes a date, allows you to apply modifiers and then returns the date as a Julian Day.
A Julian Day is the number of days since Nov 24, 4714 BC 12:00pm Greenwich time in the Gregorian calendar. The julianday function returns the date as a floating-point number.

Syntax
The syntax for the julianday function in SQLite is:
julianday(timestring [, modifier1, modifier2, ... modifier_n])

Parameters or Arguments
timestring : A date value. It can be one of the following :

timestring	Explanation
now	*now* is a literal used to return the current date
YYYY-MM-DD	Date value formatted as 'YYYY-MM-DD'
YYYY-MM-DD HH:MM	Date value formatted as 'YYYY-MM-DD HH:MM'
YYYY-MM-DD HH:MM:SS	Date value formatted as 'YYYY-MM-DD HH:MM:SS'
YYYY-MM-DD HH:MM:SS.SSS	Date value formatted as 'YYYY-MM-DD HH:MM:SS.SSS'
HH:MM	Date value formatted as 'HH:MM'
HH:MM:SS	Date value formatted as 'HH:MM:SS'
HH:MM:SS.SSS	Date value formatted as 'HH:MM:SS.SSS'
YYYY-MM-DD**T**HH:MM	Date value formatted as 'YYYY-MM-DD**T**HH:MM' where **T** is a literal character separating the date and time portions
YYYY-MM-DD**T**HH:MM:SS	Date value formatted as 'YYYY-MM-DD**T**HH:MM:SS' where **T** is a literal character separating the date and time portions
YYYY-MM-DD**T**HH:MM:SS.SSS	Date value formatted as 'YYYY-MM-DD**T**HH:MM:SS.SSS' where **T** is a literal character separating the date and time portions
DDDDDDDDDD	Julian date number

Current Date Example
You could retrieve the current date in SQLite using the "now" *timestring* with the julianday function as follows :
SELECT JULIANDAY ('now') ;
Result: 2459155.88 (Assuming current date is 2020-November-02 around 10 o'clock)

Simple Date Example - 1 -
You could convert a simple date to a Julian Day with the julianday function as follows :
SELECT JULIANDAY ('2002-08-05') ; (05 August 2002)
Result: 2452491.5
SELECT JULIANDAY ('2002-08-05 18:30') ; (05 August 2002 18:30)
Result: 2452492.27083333
SELECT JULIANDAY ('2002-08-05 18:30:20') ; (05 August 2002 18:30:20)
Result: 2452492.27106481

Simple Date Example - 2 -
You could obtain a simple date from a Julian Day with the julianday function as follows :
SELECT JULIANDAY ('2004-05-04 08:18:30') ; (04 May 2004 08:18:30)
Result: 2453129.84618056
SELECT DATE (2453129.84618056) as Birthdate, TIME (2453129.84618056) as Birthtime ;
Result : Birthdate : '2000-05-04, Birthtime : 08:18:30

DT12 – Fonctions readfile() et writefile() en SQLite – Extrait

Description
Le projet SQLite propose l'utilisation des fonctions **readfile()** et **writefile()** pour faciliter la manipulation des fichiers de type BLOB.

Fonction readfile()
La fonction SQL **readfile(X)** lit le contenu entier d'un fichier nommé X et retourne son contenu comme un fichier de type BLOB dans une table. Cette fonctionnalité peut donc être utilisée pour charger un contenu dans une table.

Exemple
```
CREATE TABLE images (name TEXT, type TEXT, img BLOB) ;
INSERT INTO images (name, type, img) VALUES ('icon', 'jpeg', readfile ('icon.jpg')) ;
```

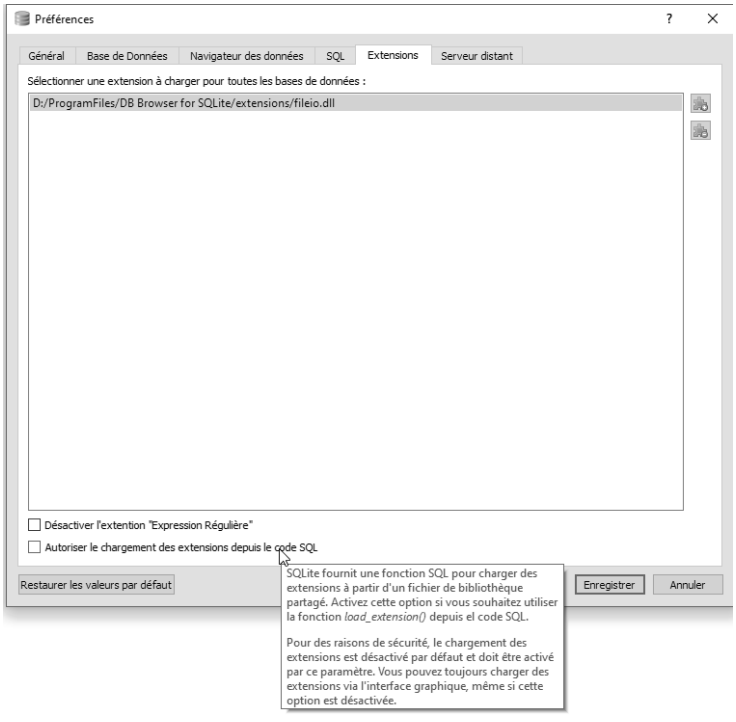
Fonction writefile()
La fonction SQL **writefile(X, Y)** écrit le contenu du BLOB Y dans un fichier nommé X et retourne le nombre d'octets écrits. Elle peut donc être utilisée pour extraire un contenu de type BLOB et le stoker dans un fichier.

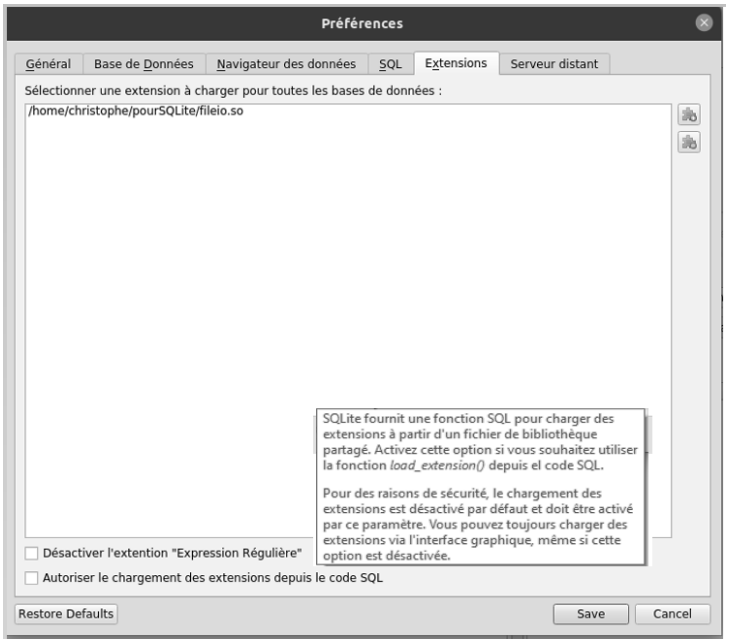
Exemple
```
SELECT writefile ('icon.jpg',img) FROM images WHERE name = 'icon' ;
```

Note importante :

Il faut noter que les fonctions **readfile(X)** et **writefile(X,Y)** ne sont pas des fonctions natives du moteur SQLite et sont des fonctions disponibles à partir du chargement des extensions.
Le fichier source est **fileio.c** ; il est disponible dans les dépôts de SQLite. Il doit être compilé.

L'outil DB-Browser dispose d'une fonctionnalité qui permet de charger directement les extensions disponibles sans passer par des requêtes SQL. Les deux copies d'écran ci-dessous montrent respectivement le chargement sous Windows et sous Linux-Ubuntu.





DOSSIER REPONSES

DR : DOCUMENTS RÉPONSES : documents à compléter et à rendre par le candidat (tous les documents réponses sont à rendre, même non complétés).

Modèle CMEN-DOC v2 ©NEOPTEC

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

Prénom(s) :

Numéro Inscription :

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

Né(e) le : / /

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

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

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

CONSIGNES

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

DR1 à DR5

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

NE RIEN ECRIRE DANS CE CADRE

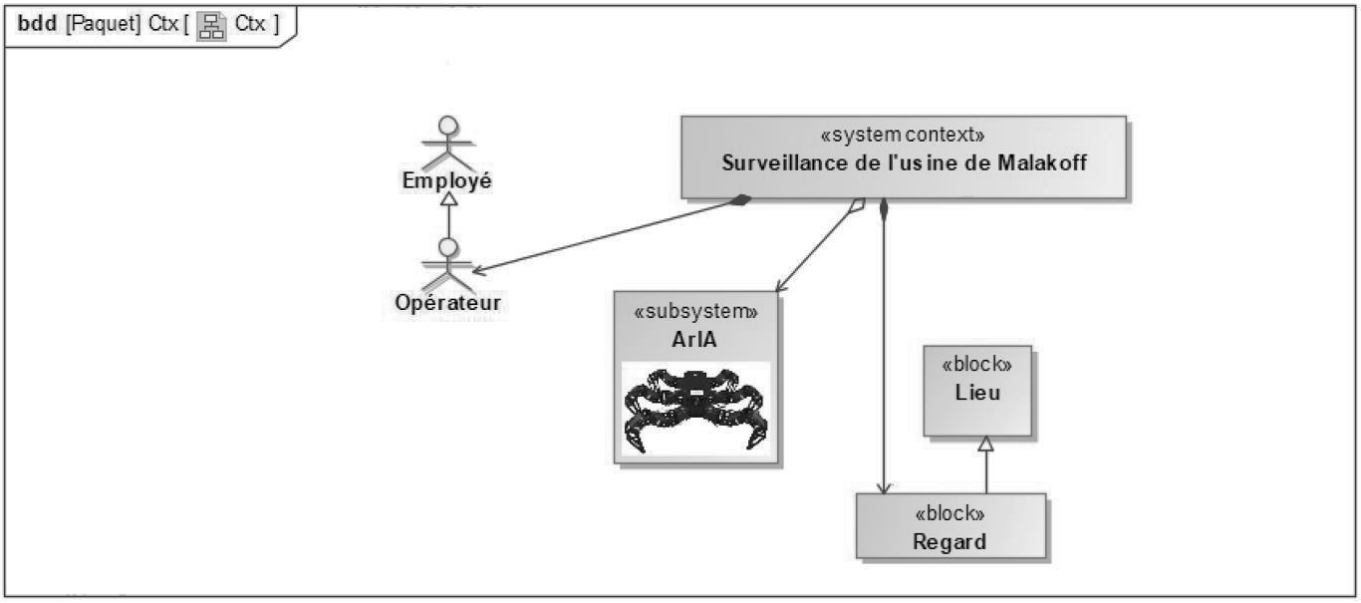
DR3 – Méthode CDJServo::setPositionAbsolue()

```
void CDJServo::setPositionAngulaire (float positionAngulaire) {
   int positionAbsolue;
   if (positionAngulaire > this->mPositionMaximaleAngulaire)
      positionAbsolue = this->mPositionCentraleAbsolue;
   else if (positionAngulaire < this->mPositionMinimaleAngulaire)
      positionAbsolue = this->mPositionCentraleAbsolue;
   else
……
……
}
```

DR4 – Méthode CDJHexapode::equilibreHautRepos()

```
void CDJHexapode::equilibreHautRepos (void) {

   this->mPatteBabordProue.setPositionAngulaire(150+20,150-41,150+77);
   this->mPatteTribordProue.setPositionAngulaire(150-20,150+41,150-77);

   this->mPatteBabordPoupe.setPositionAngulaire(150-20,150-41,150+77);
   this->mPatteTribordPoupe.setPositionAngulaire(150+20,150+41,150-77);

……
……
}
```

DR1 – Diagramme de contexte



DR2 – Trace de la fonction calculChecksum()

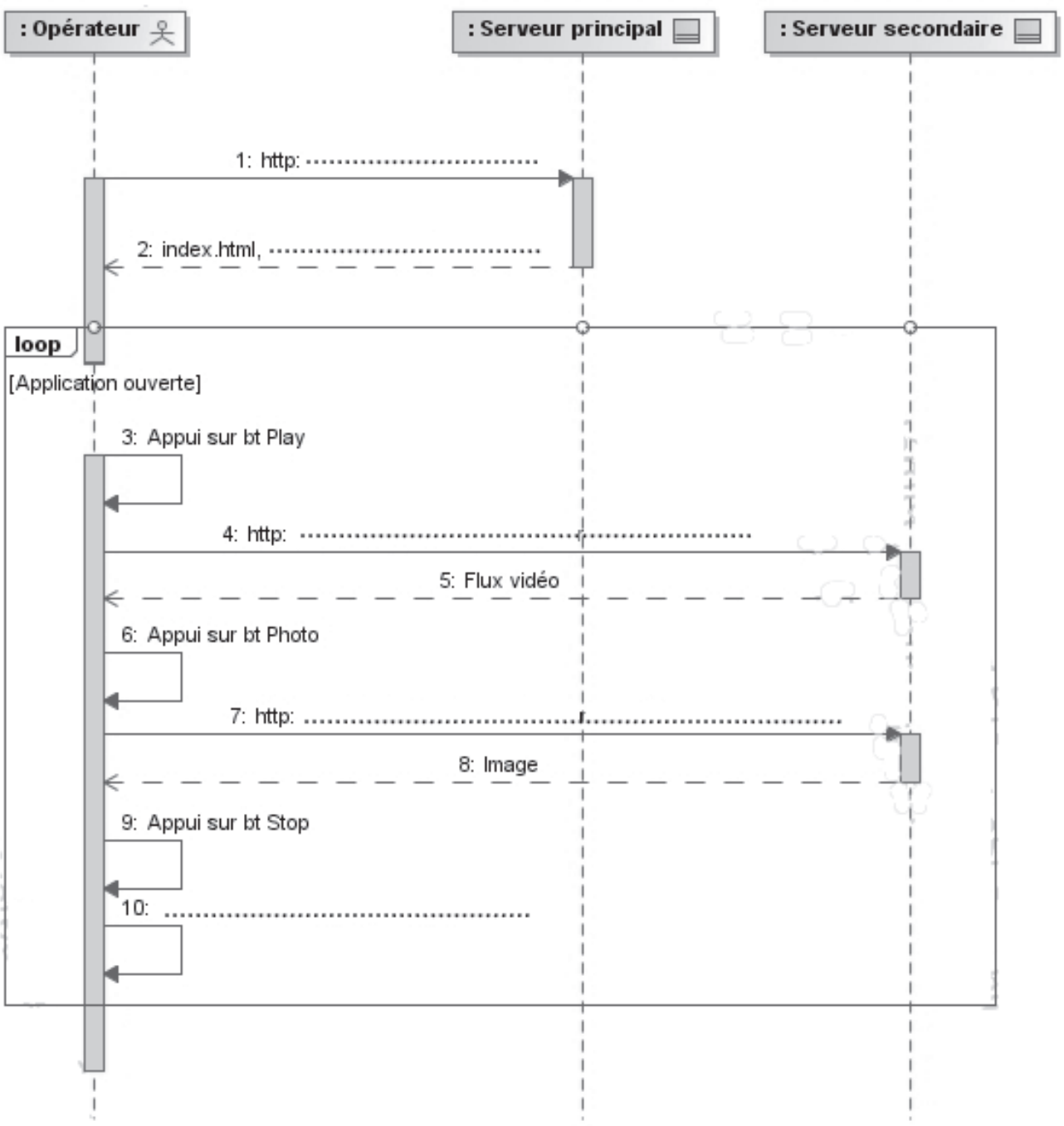
Compléter le tableau ci-dessous avec les différentes valeurs affichées à l'écran.

Valeur de i	Caractère courant	Somme courante	Somme courante (entier)
0	0000 0001	0000 0001	1
1			
2			
3			
4			
5			

Somme finale complémentée	Somme finale complémentée (entier)

Tournez la page S.V.P.

DR5 – Diagramme de séquence : Gestion images



Modèle CMEN-DOC v2 ©NEOPTEC

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

Prénom(s) :

Numéro Inscription :

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

Né(e) le : / /

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

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

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

CONSIGNES

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

DR6 à DR8

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

NE RIEN ECRIRE DANS CE CADRE

DR6 – Commande par joystick

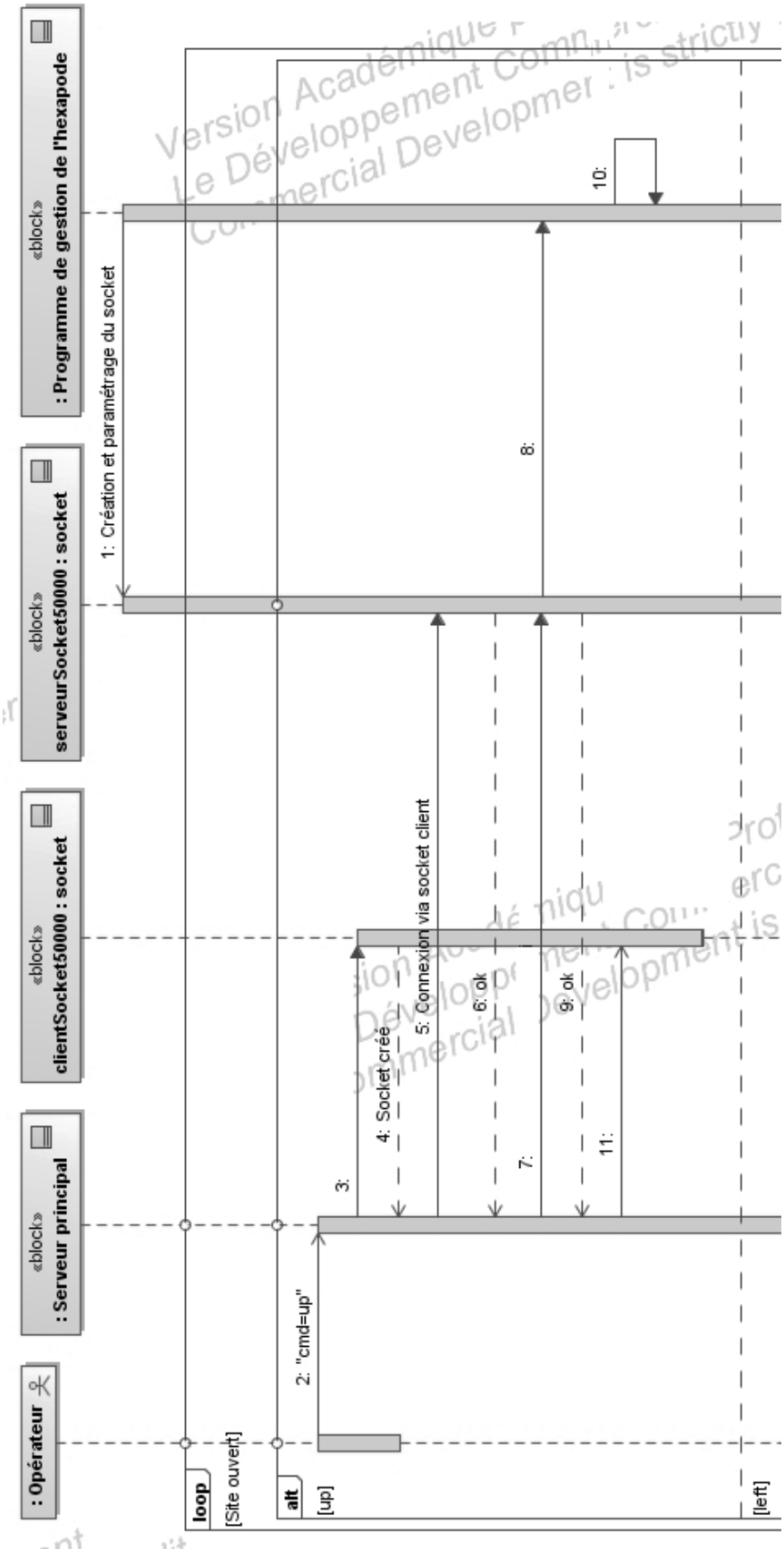
```
while (hEvent.lectureEvenement(hJoystick) == 0) {
   switch (hEvent.getEvenement().type) {

      case JS_EVENT_BUTTON :

//Gestion équilibre Haut Maximal et équilibre Haut Repos
         if ((hEvent.getEvenement().number == 4 ) && (hEvent.getEvenement().value == 1)) {
……
         }
         if ((hEvent.getEvenement().number == 5) && (hEvent.getEvenement().value == 1)) {
……
         }
            break;

      default :
         /* Ignore init events. */
         break;
      }
```

DR7 – Commande par sockets : Fichier « cmdSocket.php »

```
<?php

// Réception de la chaine de caractères
switch ($_POST['cmd']) {

        case "forward":
                $data = 0;
                break;

        case "left":
                $data = 3;
                break;

        case ……:
                ……;
                break;

        case "right":
                $data = 1;
                break;

        case "backward":
                $data = 2;
                break;

        case ……:
                ……;
                break;

        case "down":
                $data = 6;
                break;

        default:
                $data = "A";
                break;
        };

$host = "127.0.0.1";

$port = 50000;

// Création du socket
$clientSocket50000= socket_create(AF_INET, SOCK_STREAM,0) or die("pb pour créer le socket\n");

// Ouverture de la connexion
socket_connect ($clientSocket50000, $host, $port) ;

// Emission de la variable data par le socket au serveur
socket_write($clientSocket50000, $data, strlen ($data)) or die("pb d'écriture\n");

// Fermeture du socket
socket_close($clientSocket50000) ;
?>
```

Tournez la page S.V.P.

DR8 – Diagramme de séquence : Appui sur le bouton Up



Modèle CMEN-DOC v2 ©NEOPTEC

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

Prénom(s) :

Numéro Inscription :

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

Né(e) le : / /

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

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

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

CONSIGNES

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

DR9 à DR11

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

NE RIEN ECRIRE DANS CE CADRE

DR10 – Affichage des informations textuelles en mode console

```
def affichageModeConsole (resultatRequete):
# Information sur le type de données de resultatRequete
    print (type (resultatRequete))
    print (type (resultatRequete[0]))

    for ……
        ……
```

```
<class 'list'>
<class 'tuple'>
```

DR11 – Affichage de la photo en mode console

```
def affichageModeConsolePhotoUnique(imageTEMP):
    try :
        ……
    except :
        ……
    ……
```

DR9 – Modification de la requête SQL de création de la table INSPECTER

Requêtes initiales produites par l'outil MOCODO

```
CREATE TABLE "ROBOTARIA" (
  "ID_R" INTEGER,
  "dateMiseEnService" REAL,
  "photoRobot" BLOB,
  PRIMARY KEY ("ID_R")
);

CREATE TABLE "INSPECTER" (
  "ID_R" INTEGER,
  "ID_L" INTEGER,
  "ID_I" VARCHAR(42),
  "dateInspection" REAL,
  "photoLieu" BLOB,
  PRIMARY KEY ("ID_R", "ID_L"),
  FOREIGN KEY ("ID_R") REFERENCES "ROBOTARIA" ("ID_R"),
  FOREIGN KEY ("ID_L") REFERENCES "LIEU" ("ID_L")
);

CREATE TABLE "LIEU" (
  "ID_L" INTEGER,
  "codeBarres" INTEGER,
  "photoInitialeLieu" BLOB,
  PRIMARY KEY ("ID_L")
);
```

Modification de la requête de création de la table INSPECTER

```
CREATE TABLE "ROBOTARIA" (
[…]
);

CREATE TABLE "INSPECTER" (
  "ID_R" INTEGER,
  "ID_L" INTEGER,
  "ID_I"……,
  "dateInspection" REAL,
  "photoLieu" BLOB,
  PRIMARY KEY ("……"),
  FOREIGN KEY ("ID_R") REFERENCES "ROBOTARIA" ("ID_R"),
  FOREIGN KEY ("ID_L") REFERENCES "LIEU" ("ID_L")
);

CREATE TABLE "LIEU" (
[…]
);
```