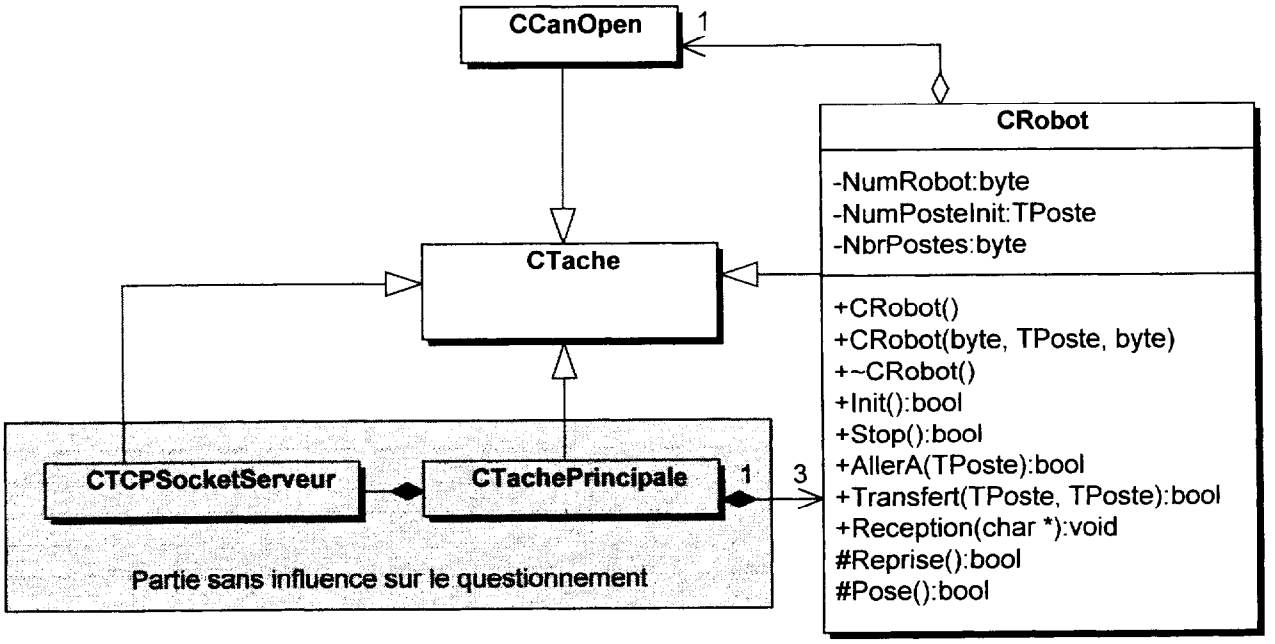
E. Programmation



E.1 Modélisation de la classe CRobot

Question E.1.1
Proposez une déclaration en langage C++ de la classe CRobot conforme au diagramme ci-dessus.

Pour des raisons de maintenance du système (analyse de défaillances, vérification des temps de cycle, ...), il est nécessaire de conserver, au niveau de chaque instance de la classe CRobot, un historique d'activité regroupant les informations datées suivantes : ordres reçus, réponses transmises, capteurs détectés.

Codage d'une information d'ordre :
- un caractère parmi 'I', 'S', 'A', 'P', 'R' pour Init(), Stop(), AllerA(), Pose(), Reprise()
- un numéro de poste (mis par convention à 0 dans le cas des ordres 'I' et 'S')

Codage d'une information capteur (position):
- un identifiant parmi 'Z', 'H', 'B', 'E', 'X', 'G', 'C', 'D'
- un numéro de poste

Codage d'une information de réponse :
- POM ou ACK ou NAK

Question E.1.2
Proposez une structure de données de type TOrdre assurant le codage d'une information d'ordre.

Question E.1.3
De la même manière, donnez une définition du type TPosition assurant le codage d'une information capteur.

Question E.1.4
Les réponses possibles émises par le robot sont implémentées comme suit :
```
enum TReponse { POM = 0xFD, ACK, NAK } ;
```
Expliquez cette ligne de déclaration.

On dispose d'un type structuré de données nommé T_timestamp permettant le codage d'une date et d'une heure avec une précision de la milliseconde.

Question E.1.5
Proposez la définition du type TInfo représentant un élément de l'historique. Cet élément doit regrouper une datation `date` suivie d'une **union** pouvant être une information d'ordre, de capteur ou de réponse.

E.2 Gestion de l'historique

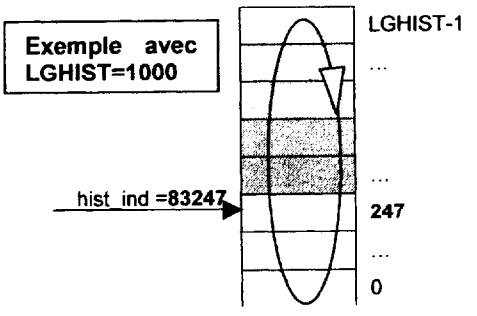
Les membres privés qui suivent sont ajoutés à la classe CRobot :
```
TInfo          hist[LGHIST] ;
unsigned int   hist_ind ;
```

Le tableau **hist** est rempli avec des octets nuls par le constructeur, **hist_ind** est initialisé à 0.

L'historique est en réalité un tampon circulaire assurant le stockage des LGHIST informations les plus récentes.

L'itérateur hist_ind :

- permet le comptage des informations depuis la naissance de l'objet CRobot,
- donne accès en permanence à l'emplacement suivant la dernière information enregistrée.



On dispose de la fonction `T_timestamp& GetTimestamp()` assurant la lecture instantanée des date et heure du système.

Question E.2.1
Les extraits de code suivants proposent des solutions d'enregistrement d'une datation. Pour chaque extrait de code, préciser s'il répond aux besoins, justifiez en cas de réponse négative.

solution 1 :
```
hist[hist_ind++].date = GetTimestamp() ;
```

solution 2 :
```
if ( hist_ind >= LGHIST )     hist_ind = 0 ;
hist[hist_ind++].date = GetTimestamp() ;
```

solution 3 :
```
hist[hist_ind].date = GetTimestamp() ;
if ( hist_ind++ > LGHIST )    hist_ind = 0 ;
```

solution 4 :
```
hist[ hist_ind++ % LGHIST ].date = GetTimestamp() ;
```

Question E.2.2
Proposez le développement d'une nouvelle méthode privée permettant l'inscription dans l'historique d'une information de type ordre. Cette méthode doit répondre au prototype suivant :
`void CRobot::AjoutHisto(TOrdre& ordre) ;`

Question E.2.3
On souhaite, de la même manière, pouvoir inscrire les informations de type position et réponse dans l'historique. Proposez des prototypes de surcharge de la méthode précédente afin de satisfaire à ce besoin.

Les informations maintenues dans l'historique sont susceptibles d'être remontées vers le système de supervision. La classe CRobot dispose en réalité pour cela d'un jeu de méthodes publiques permettant aux objets clients d'interroger son historique.

Question E.2.4
Développez en quelques lignes la méthode `TOrdre CRobot::DernierOrdre()` permettant la récupération de la dernière information de type TOrdre daté dans l'historique (hors de tout mécanisme lié au temps réel).

F. Communication et réseau

F.1 Organisation du réseau

Le réseau mis en œuvre est à base d'une architecture Ethernet 100baseT. La communication entre le système temps réel et le système de supervision utilise un protocole propriétaire.

Question F.1.1
Compléter le tableau en donnant le numéro et le nom de la couche du modèle OSI concernée par les différentes entités ou protocoles présents sur le réseau utilisé.

Entité/protocole	Câble UTP	Routeur ADSL	802.3	Connecteur RJ45	TCP	IP	hub	switch
Couche	1		Compléter le document Réponse					
Nom de la couche	Physique							

Le réseau possède une adresse IP de classe C : 192.168.17.0

Question F.1.2
Proposer un plan d'adressage possible pour les différentes machines connectées.

	Routeur ADSL	PC de gestion	PC de commande	Système temps réel
Adresse IP		Compléter le document Réponse		
Masque				

F.2 Maintenance de réseau

Pour des questions de maintenance, le technicien en charge du réseau souhaite effectuer une capture des trames émises par le système temps réel. Pour ce faire il dispose d'un ordinateur portable équipé d'un port 100BaseT / RJ45, et d'un logiciel de capture et d'analyse de trames.

On connecte cet ordinateur sur le commutateur (switch - voir diagramme de déploiement et extrait de documentation en Annexe 8)

Le système temps réel est connecté sur le port 4 et le superviseur sur le port 5.

Question F.2.1
Sur quel port faut-il connecter l'ordinateur portable ?

Question F.2.2
La question F.2.1 a-t-elle une raison d'être si l'organe de liaison est un concentrateur et non un commutateur ? Justifier la réponse.

Question F.2.3
Le câble UTP/RJ45 à utiliser est-il un câble croisé ou un câble droit ?

F.3 Client-Serveur

La communication des informations capteur entre le système temps réel et le système de supervision utilise le réseau Ethernet 100baseT et le protocole IP .

On décide d'établir une communication en mode **connecté**. Le système temps réel joue le rôle de serveur ; le PC de supervision est un client de ce dernier. Le port d'écoute fixé par le service est le port **2467**.

Question F.3.1
Les lignes suivantes proposent des solutions en langage C pour l'ouverture de la socket de communication sur le PC de supervision et sur le système temps réel. Pour chaque ligne, préciser, sans justifier, si elle répond aux besoins de l'application.

```
Solution 1 : int Sock = socket(AF_INET, SOCK_STREAM, 0) ;
Solution 2 : int Sock = socket(AF_INET, SOCK_DGRAM, 0) ;
Solution 3 : int Sock = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP) ;
Solution 4 : int Sock = socket(AF_UNIX, SOCK_STREAM, 0) ;
Solution 5 : int Sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) ;
Solution 6 : int Sock = socket(AF_IPX, SOCK_STREAM, 0) ;
Solution 7 : int Sock = socket(AF_INET, SOCK_STREAM, IPPROTO_UDP) ;
```

La socket de communication étant ouverte avec succès tant au niveau du PC de supervision que du système temps réel, l'application serveur attache maintenant cette socket à un point de communication. Cet attachement (binding) nécessite la fourniture d'une structure de type **sockaddr_in**.

```
struct sockaddr_in
{
    sa_family_t         sin_family; /* address family: AF_INET */
    u_int16_t           sin_port;   /* port in network byte order */
    struct in_addr      sin_addr;   /* internet address */
};

/* Internet address. */
struct in_addr
{
    u_int32_t     s_addr;     /* IPv4 address in network byte order */
};
```

Les questions qui suivent permettent de définir les valeurs des différents membres de la structure de type **sockaddr_in** à fournir à l'appel bind() lors de l'initialisation de la socket serveur.
(prototype : int bind(int Socket, struct sock addr *Name, int NameLen);)

Question F.3.2
Le type **u_int16_t** correspond à un entier non signé sur 16 bits.
Quelle valeur doit être fournie pour le champ **sin_port** de la structure sur le système temps réel?

Question F.3.3
A quoi correspond le champ **in_addr** de cette structure dans ce cas ? Ce champ est souvent initialisé avec la valeur symbolique **INADDR_ANY**. Que signifie cette valeur ?

Question F.3.4
L'appel à **bind()** sera suivi d'un appel à la primitive **listen()** puis à la primitive **accept()**. Quels rôles jouent chacun de ces appels systèmes dans une communication en mode connecté ?

F.4 Dialogue Système Temps Réel / Supervision

Les deux trames reproduites ci-dessous sont extraites d'un relevé réalisé par l'analyseur de protocole lors de l'envoi de la séquence informant la supervision que le robot R1 est passé au-dessus de l'un des marqueurs d'un bain.

L'analyse présente, pour les trames 13 et 14, la description des différents protocoles utilisés : Ethernet, IP, TCP.
Pour chaque protocole, la première ligne donne les caractéristiques principales, les lignes suivantes les valeurs et la signification des différents champs.
La dernière partie donne pour chaque trame le contenu brut en hexadécimal suivi d'une représentation ASCII.

```
Trame   Heure       Adr MAC src     Adr MAC dst   Protocole   Description

13      5.643554    000BDB14E06B    LOCAL         TCP         .AP..., len:    3,

  Frame: Base frame properties
      Frame: Total frame length: 60 bytes
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
    + ETHERNET: Destination address : 00E018B96B0B
    + ETHERNET: Source address : 000BDB14E06B
      ETHERNET: Frame Length : 60 (0x003C)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 46 (0x002E)
  IP: ID = 0x6C12; Proto = TCP; Len: 43
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service

      IP: Total Length = 43 (0x2B)
      IP: Identification = 27666 (0x6C12)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xEB49
      IP: Source Address = 192.168.17.22
      IP: Destination Address = 192.168.17.10
      IP: Data: Number of data bytes remaining = 23 (0x0017)
      IP: Padding: Number of data bytes remaining = 3 (0x0003)
  TCP: .AP..., len:    3, seq:2700201124-2700201127, ack:2053738035, src: 2467  dst: 4425
      TCP: Source Port = 0x09A3
      TCP: Destination Port = 0x1149
      TCP: Sequence Number = 2700201124 (0xA0F1CCA4)
      TCP: Acknowledgement Number = 2053738035 (0x7A698E33)
      TCP: Data Offset = 20 (0x14)
      TCP: Flags = 0x18 : .AP...
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....1... = Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Checksum = 0x3D42
      TCP: Data: Number of data bytes remaining = 3 (0x0003)

00000:  00 E0 18 B9 6B 0B 00 0B DB 14 E0 6B 08 00 45 00   .à.¹k...Û.àk..E.
00010:  00 2B 6C 12 40 00 80 06 EB 49 C0 A8 11 16 C0 A8   .+l.@.€.ëIÀ¨..À¨
00020:  11 0A 09 A3 11 49 A0 F1 CC A4 7A 69 8E 33 50 18   ...£.I ñÌ¤ziŽ3P.
00030:  FA F0 3D 42 00 00 43 06 00 00 00 00               úð=B..C.....

********************************************************************************


Trame   Heure       Adr MAC src     Adr MAC dst   Protocole   Description

14      5.846679    LOCAL           000BDB14E06B  TCP         .A...., len:    0

  Frame: Base frame properties
      Frame: Total frame length: 54 bytes
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
    + ETHERNET: Destination address : 000BDB14E06B
    + ETHERNET: Source address : 00E018B96B0B
      ETHERNET: Frame Length : 54 (0x0036)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 40 (0x0028)
  IP: ID = 0xD73F; Proto = TCP; Len: 40
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
```

```
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 40 (0x28)
      IP: Identification = 55103 (0xD73F)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xC01F
      IP: Source Address = 192.168.17.10
      IP: Destination Address = 192.168.17.22
      IP: Data: Number of data bytes remaining = 20 (0x0014)
TCP: .A...., len:    0, seq:2053738035-2053738035, ack:2700201127, src: 4425  dst: 2467
      TCP: Source Port = 0x1149
      TCP: Destination Port = 0x09A3
      TCP: Sequence Number = 2053738035 (0x7A698E33)
      TCP: Acknowledgement Number = 2700201127 (0xA0F1CCA7)
      TCP: Data Offset = 20 (0x14)
      TCP: Flags = 0x10 : .A....
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Checksum = 0x8053

00000:  00 0B DB 14 E0 6B 00 E0 18 B9 6B 0B 08 00 45 00   ..Û.àk.à.¹k...E.
00010:  00 28 D7 3F 40 00 40 06 C0 1F C0 A8 11 0A C0 A8   .(×?@.@.À.À¨..À¨
00020:  11 16 11 49 09 A3 7A 69 8E 33 A0 F1 CC A7 50 10   ...I.£ziŽ3 ñÌ§P.
00030:  FA ED 80 53 00 00                                 úí€S..
```

Question F.4.1
Indiquez les adresses Ethernet et les adresses IP du système de supervision et du système temps réel.

	PC de supervision	Système temps réel
Adresse Ethernet		
Adresse IP		

~~Compléter le document Réponse~~

Question F.4.2
En identifiant dans la trame 13 le numéro de port source, préciser si cette trame a été émise par le serveur ou par le client.

Question F.4.3
Les paquets IP sont-ils fragmentés ? Justifier la réponse.

Question F.4.4
Les données spécifiques au service de l'application commence à l'octet 36 hexadécimal de la trame 13. Quel est l'identifiant du capteur concerné par cette capture ?

Question F.4.5
Quel est le rôle de la trame 14 ?

Fin du questionnement