



BTS SNIR

Lycée Louis Armand
94 - Nogent sur Marne



Modules Systèmes C / C++

MS1 La calculatrice sur journal lumineux

MS2 Commande d'un système d'éclairage de scène

MS3 Caméras IP de surveillance

MS4 Supervision d'un véhicule sur bus CAN

MS5 Surveillance météorologique

MS6 Partie d'échecs en réseau retransmise sur Internet et sur écran géant à LED

Nom :

Groupe :



Module Système 1

La calculatrice sur journal lumineux



Table des matières

Objectifs, matériels et ressources.....	2
TD - Cahier des charges et analyse UML.....	3
TD - Comprendre la structure d'un code en C + +	4
TP - Défi 1 – Les opérations +, -, *, / de la calculatrice.....	5
TP - Défi 2 – Le menu de la calculatrice.....	6
TP - Défi 3 – La trigonométrie.....	7
TP - Défi 4 – Les conversions : décimal, hexadécimal et binaire.....	8
TP - Défi 5 – Fichier log.....	10
TD – Le protocole du journal lumineux.....	11
TP – Défi 6 – Envoyer un message au journal lumineux avec la Raspberry PI.....	13
TD - La classe SNAfficheur.....	14
TP - Défi 7 - Premier programme pour le journal lumineux – la classe SNAfficheur.....	15
TD – La classe SNLigne.....	16
TP – Défi 8 – Second programme pour le journal lumineux – la classe SNLigne.....	17
TP – Défi 9 – Les résultats de la calculatrice sur le journal lumineux.....	18
Annexe 1 – Diagramme de classe complet des classes Afficheur et Ligne.....	19
Annexe 2 – UML - Guide de mise en place du diagramme de cas d'utilisation.....	20
Annexe 3 – UML - Guide de mise en place du diagramme de déploiement.....	21
Annexe 4 – UML - Guide de mise en place du diagramme de séquence "système" :.....	22
Glossaire.....	24

Objectifs, matériels et ressources

Objectifs

- Découvrir la structure d'un programme en C++.
- Apprendre à écrire des instructions simples en C++ permettant de
 - créer des variables
 - saisir des valeurs
 - traiter des données
 - afficher un résultat.
- Créer des objets en C++ à partir de classes existantes
- Appeler les méthodes d'objets créés.

Matériels

- Un journal lumineux
- Une Raspberry PI



Ressources

- La documentation du journal lumineux ([590996-da-01-en-Communication_protocol_LED_Displ_Board.pdf](#)).
- Les sites de référence des librairies standards du C/C++ : [www.cplusplus.com](#) ou [cppreference.com](#).
- Le site [www.raspberrypi.org](#) pour toute information sur la Raspberry

TD - Cahier des charges et analyse UML

Le cahier des charges

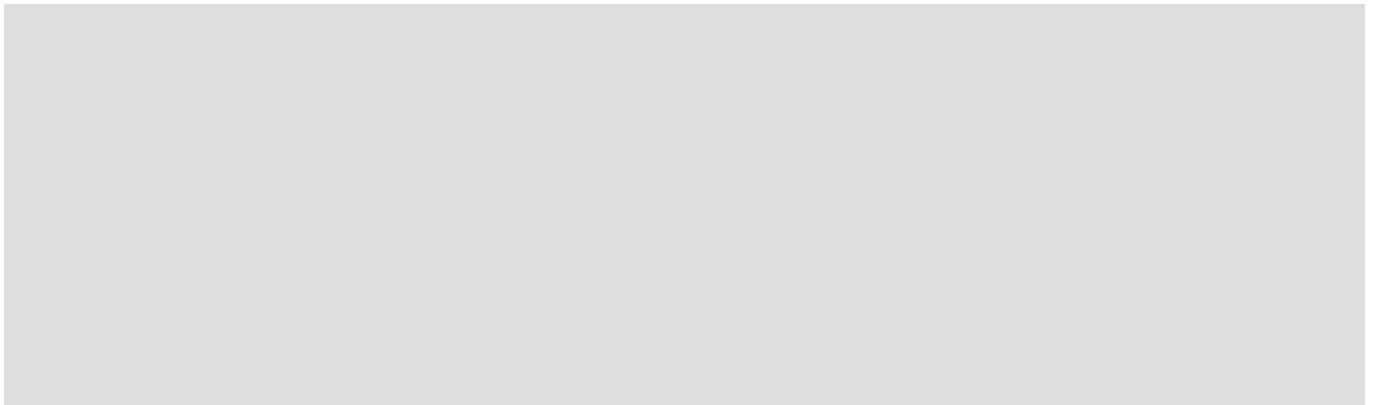
Un professeur de mathématiques souhaite que ces élèves puissent lire sur un journal lumineux les opérations qu'il saisit sur une calculatrice.

Votre mission consiste à développer pour ce prof de maths une application servant de calculatrice. Elle devra afficher tous les calculs sur un journal lumineux. Ce journal lumineux doit être accessible depuis n'importe quelle machine ; on le muni donc d'une Raspberry (un mini PC sous linux) possédant une interface Wifi et/ou Ethernet.

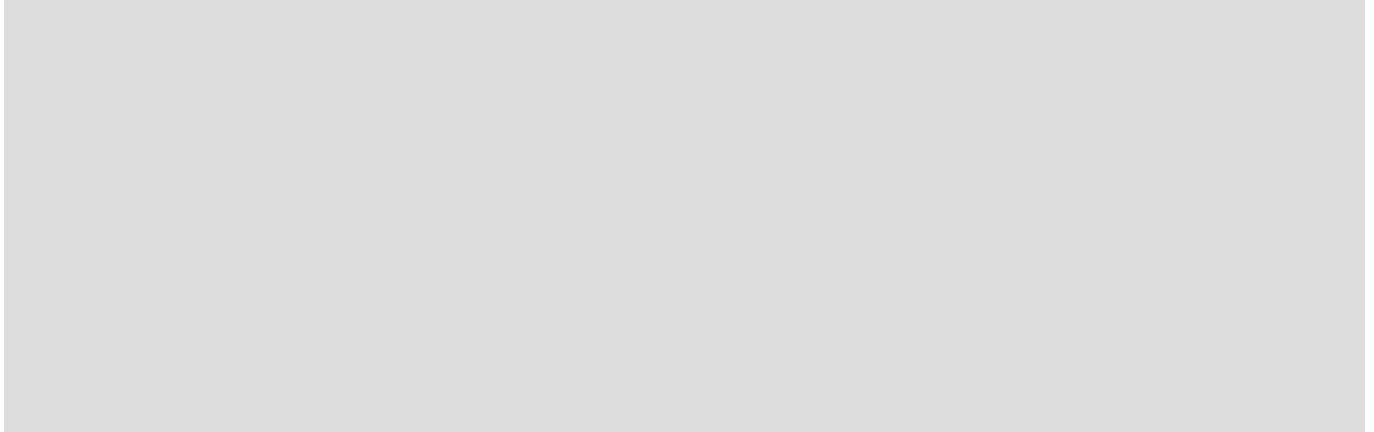
Les diagrammes de cas d'utilisation, de déploiement et de séquence



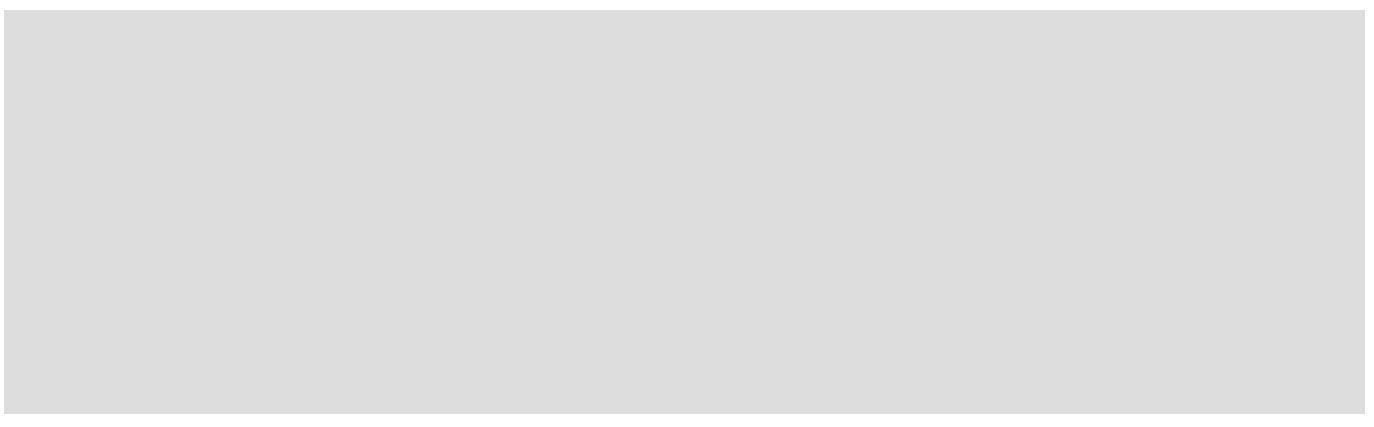
Dessiner le diagramme de cas d'utilisation :



Dessiner le diagramme de déploiement :



Dessiner le diagramme de séquence d'un point de vue système :



TD - Comprendre la structure d'un code en C++

Comprendre la structure d'un code en C++

Soit le code suivant écrit en C++ servant de point de départ à ce module :

```
#include <iostream>
using namespace std;

int main()
{
    float valeur;

    cout << "Saisir une valeur : ";
    cin >> valeur;

    cout << "La valeur que vous avez saisie est : " << valeur << endl;
    return 0;
}
```



Repérer dans ce code :

- la zone des inclusions :
- la définition de la fonction principale et son étendue :
- le nombre d'instructions dans la fonction principale :
- la zone de création des variables :

Le code précédent peut être représenté par le diagramme d'activité ci-dessous :



Associer les activités de ce diagramme avec les instructions du code.



Donner le numéro de ligne des instructions créant des variables.



Donner le nom de la variable créée.



Donner le type de la variable créée.



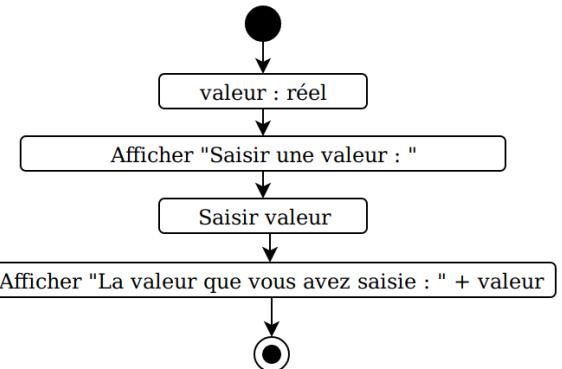
Lister les autres types de variable possibles en C++.



Expliquer l'utilité de l'objet « cout ».



Expliquer l'utilité de l'objet « cin ».

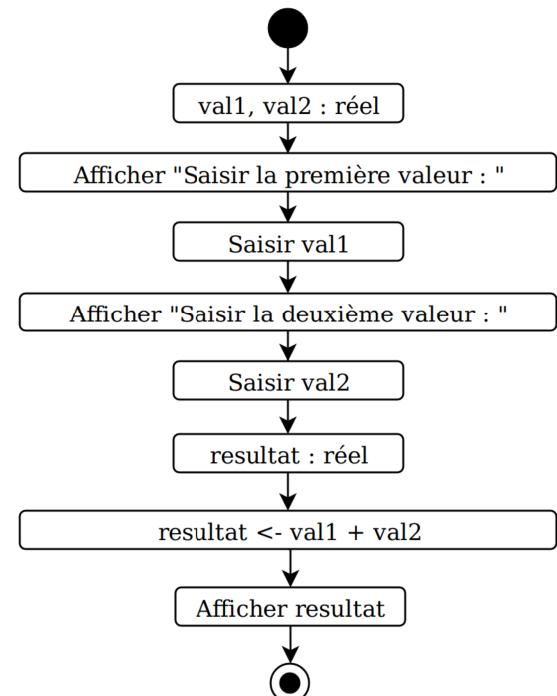
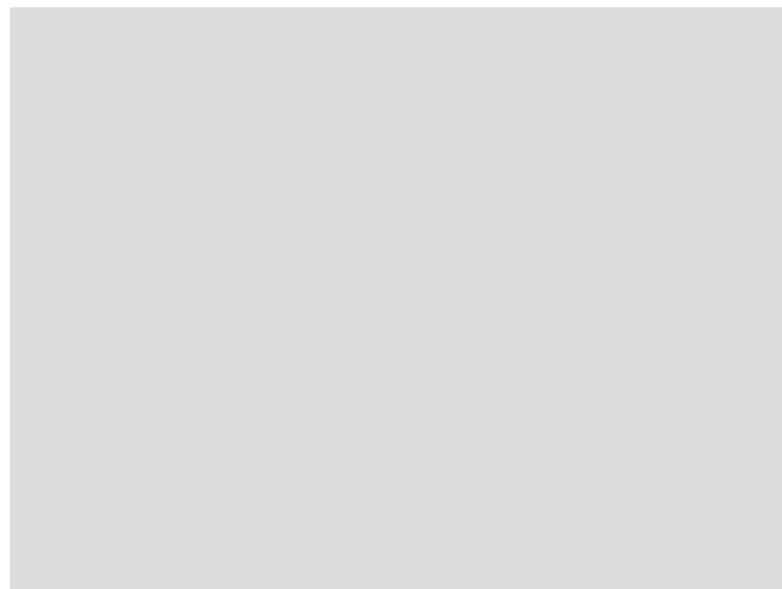


TP – Défi 1 – Les opérations +, -, *, / de la calculatrice

L'addition de 2 nombres

Le diagramme d'activité ci-contre représente la suite des instructions de la fonction principale (*main*) pour effectuer l'addition de 2 nombres.

Interpréter ce diagramme d'activité en C++ :



Créer un projet en mode console. Saisir le code dans la fonction *main* et le tester.

Saisie de l'addition sur une ligne

Avec l'objet « *cin* », il est possible de décomposer ce que saisi l'utilisateur avec son clavier. Nous souhaitons que le prof de maths puisse saisir son calcul en une seule saisie. Par exemple, il pourrait taper

$$15.6 + 7.3$$

sur une seule ligne et quand il frappe la touche entrée, le résultat apparaît.

Les 3 informations saisies seront stockées dans « *val1* » (la première valeur, ici 15,6), dans « *opérateur* » (pour l'opérateur, ici +) et dans « *val2* » (la deuxième valeur, ici 7,3).

Créer une nouvelle variable nommée « *opérateur* » de type *char*.

Effectuer la saisie de *val1*, *opérateur* et *val2* en une seule fois

Gestion des opérateurs +, - * et /.

L'objectif de cette partie est de permettre au prof de maths de faire des additions, des soustractions, des multiplications et des divisions.

En utilisant la structure *if ... else*, gérer les valeurs possibles de la valeur de « *opérateur* » et effectuer le calcul correspondant.

BONUS : utiliser la structure alternative *switch ... case* pour gérer les différentes valeurs possibles de « *opérateur* ».

TP – Défi 2 – Le menu de la calculatrice

Le menu

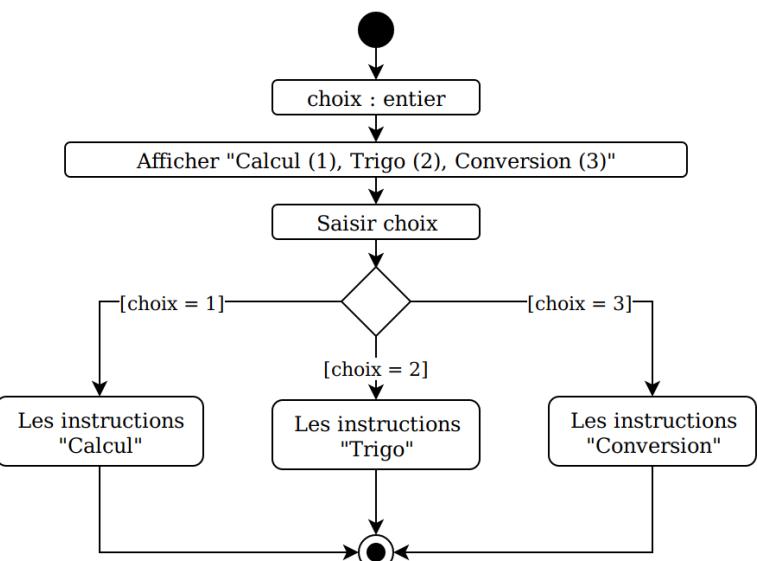
En plus des calculs précédents, notre prof de maths souhaite également faire des calculs trigonométriques (cosinus, sinus et tangente) et des conversions de base (décimal, hexadécimal et binaire).

Cependant, le format des opérations trigonométriques étant différent des autres calculs, le prof de maths pourra choisir au début du programme s'il souhaite faire un calcul (1), une opération trigonométrique (2) ou une conversion (3)



En suivant le diagramme d'activité ci-contre, écrire le code en C++ permettant de proposer un choix entre 'Calcul', 'Trigo' et 'Conversion'.

Remarque : Toutes les instructions (sauf les déclarations) de votre code font parties de l'activité : Les instructions « Calcul ».



TP – Défi 3 – La trigonométrie

Les instructions « Trigo »

Une instruction trigonométrique est composée de deux éléments : la fonction et la valeur. Par exemple, dans

« cos 3.14 »

cos est la fonction et 3.14 est la valeur exprimée en radian.

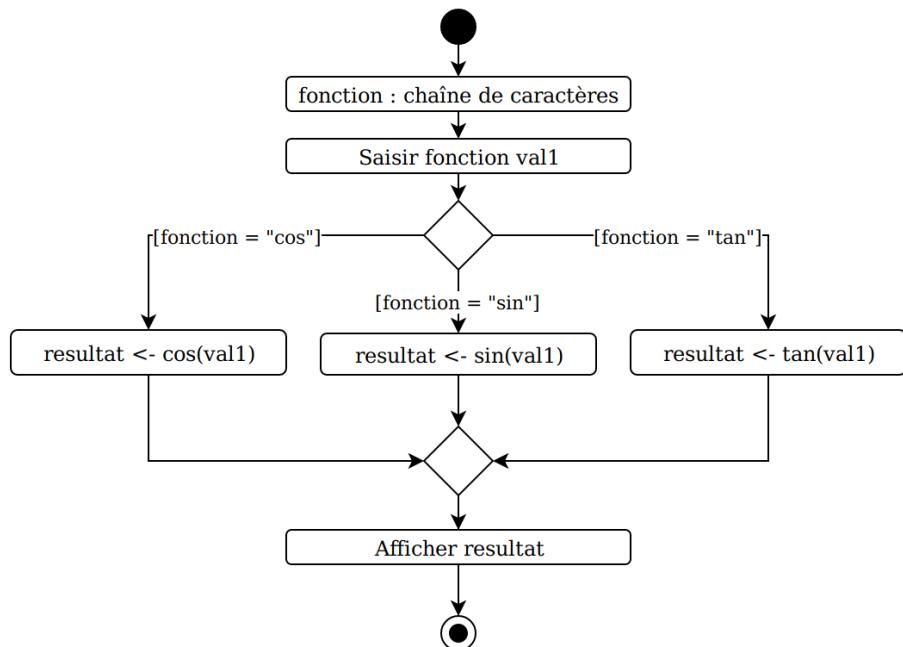
Lorsque le prof de maths saisira un calcul trigonométrique, il faudra donc sauvegarder 1) la fonction et 2) la valeur. La fonction est une suite de caractères, elle s'appellera « fonction » dans le programme et sera un tableau de caractères. La valeur est un réel qui sera enregistré dans « val1 ».

Remarque 1 : La déclaration d'une chaîne de 10 caractères nommée « fonction » en C++ s'écrit :

```
char fonction[10] ;
```

Remarque 2 : la fonction `strcmp()` permet la comparaison de 2 chaînes de caractères. Consulter la documentation de cette fonction sur les sites de référence du C++.

Remarque 3 : la bibliothèque `math.h` contient de nombreuses fonctions mathématiques et trigonométriques. Consulter les sites de référence du C++ sur l'utilisation de cette bibliothèque.



Interpréter en C++ le diagramme d'activité ci-contre en utilisant les remarques ci-dessus :

Le choix de l'unité : radian ou degré

Proposer un menu permettant à l'utilisateur de choisir l'unité qu'il souhaite (degré ou radian) avant de saisir son calcul. S'il choisit degré, convertir la valeur saisie en radian.

TP – Défi 4 – Les conversions : décimal, hexadécimal et binaire

L'objectif de cette partie est de permettre au prof de maths de saisir un nombre dans n'importe quelle base (décimale, hexadécimale ou binaire) et de l'afficher ensuite dans une autre base.

Les conversions en décimal et en hexadécimal

Les conversions en décimal et en hexadécimal fonctionnent sur le même principe. Le cas du binaire étant différent, nous l'examinerons ensuite.

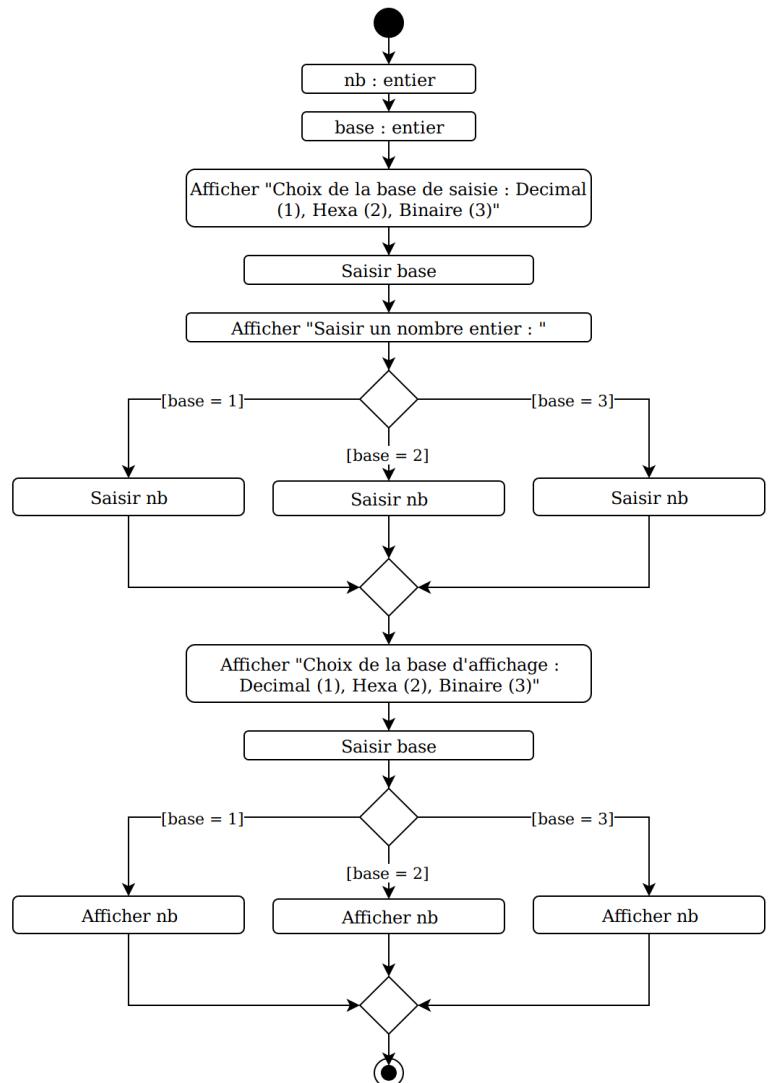
Les objets cin et cout savent saisir et/ou afficher des valeurs en décimal et en hexadécimal. Par défaut, c'est le décimal. Pour changer de base, il faut écrire :

```
cout << hex << nb;
```

La ligne précédente affiche 'nb' en hexadécimal... mais elle affichera toutes les prochaines valeurs en hexadécimal. Pour revenir en décimal, il faut écrire :

```
cout << dec << nb;
```

En suivant le diagramme d'activité et les remarques ci-dessous, écrire le code en C++ de la partie 'Conversion' pour le décimal et l'hexadécimal.



Saisir votre code et le tester

Les conversions en binaire

Par contre, cin et cout ne gérant pas la saisie et l'affichage en binaire, vous utiliserez les fonctions `int SaisirBinaire()` et `string EntierVersChaineEnBinaire(int nombre)`.

Pour saisir une valeur binaire, écrire :

```
nb = SaisirBinaire() ;
```

Pour l'affichage d'une valeur en binaire, écrire :

```
cout << EntierVersChaineEnBinaire(nb);
```

Ajouter la définition des 2 fonctions `SaisirBinaire()` et `EntierVersChaineEnBinaire()` au-dessus de la fonction `main()`.

```
int SaisirBinaire()
{
    char bin[100];
    int nombre = 0;
    cin >> bin;
    for(int i=0;i<strlen(bin);i++)
    {
        nombre = nombre*2+(bin[i]-'0');
    }
    return nombre;
}
string EntierVersChaineEnBinaire(int nombre)
{
    ostringstream oss;
    bool copie = false;
    for(int i=31;i>=0;i--)
    {
        if(((nombre>>i)&1) == 1)
        {
            copie = true;
        }
        if(copie == true)
        {
            oss<<((nombre>>i)&1);
        }
    }
    return oss.str();
}
```

Ces 2 fonctions nécessitent les inclusions suivantes. Ajouter les inclusions manquantes dans la zone des inclusions.

```
#include <sstream>           // pour la classe ostringstream
```

En utilisant le diagramme d'activité précédent, ajouter la conversion en binaire. Tester votre code.

TP – Défi 5 – Fichier log

La chaîne de caractères « phrase »

Au lieu de faire un affichage avec un cout à la fin de chaque calcul, nous allons sauvegarder le calcul complet dans une chaîne de caractères. Par exemple, si le prof de maths a saisi $5 + 3$, nous allons stocker « $5 + 3 = 8$ » dans une chaîne de caractères. Cette chaîne sera ensuite affichée grâce à un cout, enregistrée dans un fichier log et plus tard envoyée au journal lumineux. Cette chaîne de caractères sera appelée « phrase ».



Écrire en C++ la déclaration une chaîne de 200 caractères nommée « phrase » :

Ajouter cette déclaration au début de la fonction *main()*.

Pour écrire la phrase « $5 + 3 = 8$ » dans la chaîne de caractères « phrase », utiliser la fonction *sprintf()* comme suit :

```
 sprintf(phrase,"%3f %c %.3f = %.3f", val1, operateur, val2, resultat);
```

Vous trouverez de nombreuses explications sur les sites de références (<http://www.cplusplus.com/reference/cstdio/sprintf/>).

Remplacer tous les affichages de résultats de calcul par *sprintf()*.

Affichage de « phrase » dans le terminal



Écrire en C++ l'affichage dans un terminal de la chaîne de caractères nommée « phrase ».

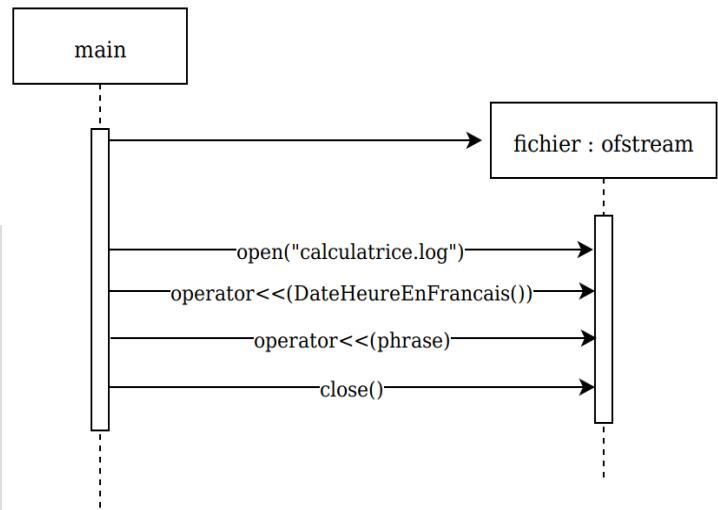
Ajouter ce code à la fin de la fonction *main()*.

Enregistrement de « phrase » dans un fichier

Pour écrire dans un fichier, la bibliothèque standard du C++ fournit la classe *ofstream*. Le diagramme de séquence suivant illustre l'utilisation de cette classe dans notre programme.



Interpréter en C++ le diagramme de séquence ci-contre.



Ajouter ce code après l'affichage dans le terminal de « phrase » ainsi que l'inclusion « *fstream* » : c'est la bibliothèque de la classe *ofstream* !

TD – Le protocole du journal lumineux

La communication avec le journal lumineux doit se faire à 9600 baud, 8 bits de données, pas de parité et 1 bit stop.

Le document constructeur du journal lumineux explique le format de la trame (ou du message) qu'il attend. Voici un exemple de trame :

```
<ID00><L1><PA><FE><MA><WC><FE>BTS SN0C<E>
```

Ce message respecte le format prédéfini par le constructeur du journal lumineux. Dans le tableau suivant, vous trouverez une description du message :

<ID00>	<L1> <PA> <FE> <MA> <WC> <FE> BTS SN	0C	<E>
Identifiant de l'afficheur	Bloc de données : Les effets d'affichage puis le texte à afficher	Checksum	Fin

En utilisant la documentation du constructeur de l'afficheur (pages 5 à 8), expliquer la signification de <L1>, <PA>, <FE>, <MA>, <WC> et <FE>.

Vérifier que le checksum est correct en effectuant un OU EXCLUSIF entre tous les codes ASCII des caractères du bloc de données.



Sans modifier les effets d'affichage, écrire la trame à envoyer pour afficher votre nom. Détailler le calcul du checksum.

TP – Défi 6 – Envoyer un message au journal lumineux avec la Raspberry PI

Les objectifs

Les objectifs de cette partie sont :

- Ouvrir un terminal distant de la Raspberry PI et exécuter quelques commandes utiles
- Envoyer une trame au journal lumineux et vérifier sa validité

Ouvrir d'un terminal distant

Vous allez tous travailler sur le même journal lumineux. Il est connecté en USB sur une Raspberry PI disposant d'un système d'exploitation Raspbian (une distribution du l'OS Linux). Il est possible de travailler à distance sur les OS Linux grâce au protocole SSH.

Donner la signification du sigle SSH et expliquer brièvement l'intérêt de ce protocole.

Pour travailler à distance sur la Raspberry PI, vous disposerez d'un terminal distant sur votre machine.

Ouvrir un terminal distant sur la Raspberry en utilisant un des logiciels suivants : SmarTTY, Putty ou TeraTerm.

Trouver les paramètres du journal lumineux

Le journal lumineux (appelé CP210X) est connecté sur un port USB de la Raspberry PI. La commande suivante permet de connaître la liste des matériels USB connectés à une machine :

```
pi@raspberry:~$ lsusb
```

Taper cette commande pour vérifier que le journal lumineux est bien connecté en USB.

Notre afficheur lumineux est automatiquement vu par le système d'exploitation Linux comme possédant un port série virtuel. La commande suivante permet de connaître le nom de son port série virtuel :

```
pi@raspberry:~$ dmesg | grep usb
```

Saisir la commande précédente dans le terminal et indiquer le nom du port série virtuel du journal lumineux.

Afficher un message sur le journal lumineux avec un programme déjà existant

Une fois le terminal du poste de l'afficheur ouvert, vous allez utiliser la commande « send2LedDisplay » pour envoyer un message à l'afficheur (« send2LedDisplay » est une application fabriquée par vos enseignants).

```
pi@raspberry:~$ send2LedDisplay /dev/ttyUSB0 '<ID00><L1><PA><FE><MA><WC><FE>BTS SN0C<E>'
```

Se reporter au TD précédent (« Le protocole du journal lumineux ») pour comprendre cette trame.

Envoyer un message personnalisé au journal lumineux en ajoutant 2 fois la même lettre dans le texte à afficher. Par exemple, si on ajoute deux 'G', on obtient le message : « <ID00><L1><PA><FE><MA><WC><FE>BTS SNGG0C<E> ».

Envoyer la trame contenant votre nom et votre prénom (voir le TD précédent « Le protocole du journal lumineux »). Vérifier que votre message s'affiche correctement.

TD - La classe SNAfficheur

La classe SNAfficheur a été créée par vos professeurs afin de faciliter la programmation du journal lumineux. Vous trouverez ci-contre son diagramme de classe.

?

Compter le nombre d'attributs de la classe SNAfficheur:

?

Compter le nombre de méthodes de la classe SNAfficheur:

?

Donner les noms d'usage des 2 premières méthodes de cette classe.

SNAfficheur

- portSerie : PortSerieAfficheur
- adresseIPServerUDP : string
- portServerUDP : unsigned short

- + SNAfficheur()
- + ~SNAfficheur()
- + OuvrirPortSerie(nomPortSerie : string) : bool
- + FermerPortSerie() : bool
- + EnvoyerTrame(trame : char *, longueurTrame : int) : bool
- + EnvoyerTrame(trame : string) : bool
- + EnvoyerLigne(ligne : SNLigne) : bool
- + EnvoyerTrameEnUDP(trame : char *, longueurTrame : int) : bool
- + EnvoyerTrameEnUDP(trame : string) : bool
- + EnvoyerLigneEnUDP(ligne : SNLigne) : bool
- + ModifierAdresseIPDuServeurUDP(adresseIPServerUDP : string) : void
- + ModifierPortDuServeurUDP(port : unsigned short) : void

?

La classe SNAfficheur est déclarée dans le fichier SNAfficheur.h. Écrire en C++ l'instruction d'inclusion à ajouter dans le fichier main.cpp pour utiliser la classe SNAfficheur.

?

Écrire en C++ la création d'un objet nommé 'aff' de la classe SNAfficheur.

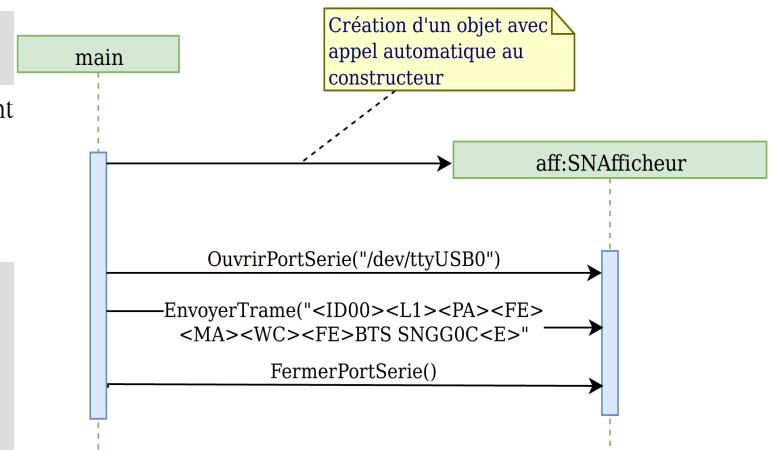
?

Donner le nom de la méthode qui est automatiquement appelée lors de la création d'un objet.

Votre première application utilisera cette classe en suivant le diagramme de séquence ci-contre.

?

Lister les 4 méthodes de la classe SNAfficheur utilisées dans ce diagramme de séquence.



?

Interpréter en C++ le main() de ce diagramme de séquence.

TP - Défi 7 - Premier programme pour le journal lumineux – la classe SNAfficheur

Les objectifs

L'objectif final de cette partie est de développer une application en mode console qui affichera votre nom sur le journal lumineux.

Premier code : envoyer un message prédéfini au journal lumineux

Mise en place du projet C++

Récupérer l'archive « MS1_CodeEtudiant » sur le répertoire de partage. Dézipper-la dans votre répertoire de travail.

Avec WinSCP, créer un dossier sur la Raspberry à votre nom puis déplacer tous les fichiers de l'archive « MS1_CodeEtudiant » dans ce dossier.

Écriture de la fonction principale main

Avec WinSCP, éditer le fichier `main.cpp` et saisir le code du diagramme de séquence du TD précédent (« La classe Afficheur »). Enregistrer.

Compilation et exécution du programme

Dans le terminal, déplacer vous dans votre dossier en tapant :

```
cd MonReperatoire
```

Compiler votre programme en tapant :

```
make
```

Si des erreurs apparaissent, vous devez les corriger, enregistrer vos modifications et recompiler. Si vous n'avez plus d'erreur, lancer votre programme en tapant :

```
./afficheur
```

Vérifier que votre programme fonctionne correctement.

Deuxième code : afficher votre nom et votre prénom sur le journal lumineux

Écrire la trame à envoyer avec les mêmes effets d'affichage, contenant votre nom et possédant le bon checksum (voir le TD « Le protocole du journal lumineux »)

Modifier maintenant le code source avec cette nouvelle trame. Compiler et exécuter votre application.

On désire maintenant dans cette partie modifier quelques effets d'affichage :

- l'apparition du texte se fera avec un effet neige ;
- le temps d'affichage sera de 20 secondes
- le texte sera clignotant;
- le texte disparaîtra sans effet.

Écrire la trame à envoyer pour respecter ces effets. Vous utiliserez la documentation du journal lumineux pour trouver les valeurs.

Modifier votre code source avec votre nouvelle trame. Puis compiler et exécuter votre application.

TD – La classe SNLigne

Dans la suite de cette activité, nous allons utiliser la classe « SNLigne » qui va nous faciliter la mise en place d'une trame à envoyer à l'afficheur. Cette classe SNLigne permet de gérer plus facilement le message à afficher et les effets à appliquer. Son diagramme de classe se trouve ci-contre.

Donner le nombre d'attributs de la classe SNLigne :

Donner le nombre de méthodes de la classe SNLigne :

Donner le nom d'usage de la première méthode de la classe SNLigne et son but.

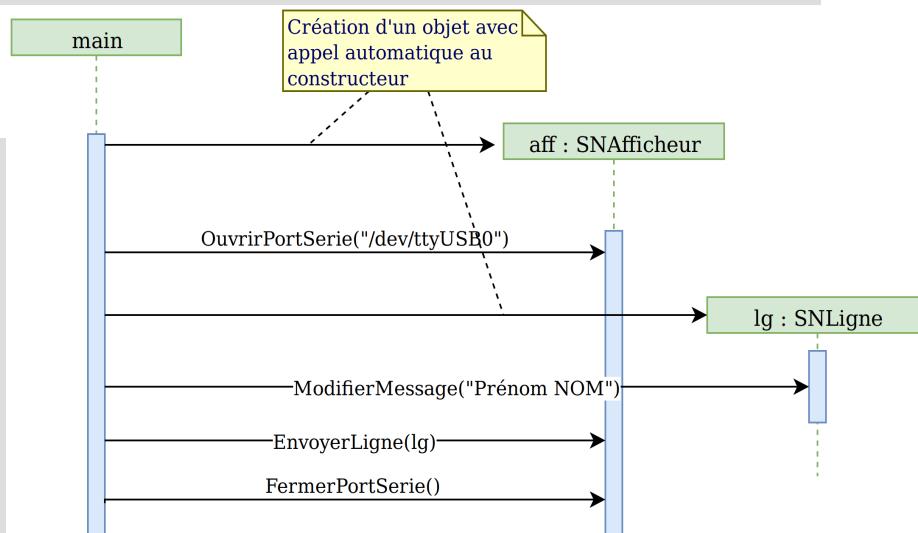
SNLigne
- message : string
- effetChargement : SNEffetChargement
- effetAffichage : SNEffetAffichage
- temporisationAffichage : SNTemporisationAffichage
- effetSortie : SNEffetSortie
+ SNLigne()
+ ~SNLigne()
+ ModifierNumeroLigne(numeroLigne : int) : void
+ NumeroLigne() : int
+ ModifierMessage(message : string) : void
+ Message() : string
+ ModifierEffetChargement(effetChargement : SNEffetChargement) : void
+ EffetChargement() : SNEffetChargement
+ ModifierEffetAffichage(effetAffichage : SNEffetAffichage) : void
+ EffetAffichage() : SNEffetAffichage
+ ModifierTemporisationAffichage(temporisationAffichage : SNTemporisationAffichage) : void
+ TemporisationAffichage() : SNTemporisationAffichage
+ ModifierEffetSortie(effetSortie : EffetSortie) : void
+ EffetSortie() : EffetSortie

Donner le nom d'usage de la seconde méthode de cette classe. Expliquer comment on la reconnaît.

La classe SNLigne est déclarée dans le fichier SNLigne.h. Écrire en C++ l'instruction d'inclusion à ajouter dans le fichier main.cpp pour utiliser la classe Ligne.

Écrire en C++ la création d'un objet nommé 'lg' de la classe SNLigne.

A partir du diagramme de séquence ci-contre, donner le code en C++ de la fonction principale main.



TP – Défi 8 – Second programme pour le journal lumineux – la classe SNLigne

Les objectifs

L'objectif de cette partie est d'utiliser la classe SNLigne pour créer facilement de nouveau message. Le programme final pourra également modifier les effets d'affichage des messages sur le journal lumineux.

Envoyer un message prédéfini au journal lumineux

Écrire le code final du TD précédent (« La classe SNLigne ») dans la fonction principale. Ne pas oublier les inclusions.
Compiler et exécuter votre programme. Corriger les éventuelles erreurs.

Saisir le message à afficher par l'utilisateur

L'utilisateur de votre logiciel doit maintenant pouvoir saisir son propre message.

1. Dans le main, ajouter la déclaration d'une chaîne de caractères nommée « monTexte ».
2. Ajouter l'instruction demandant à l'utilisateur de saisir un texte. Utiliser l'objet « cout ».
3. Ajouter l'instruction permettant de saisir les caractères saisis au clavier dans la chaîne de caractères « monTexte ». Utiliser l'objet « cin ».
4. Modifier l'instruction qui appelle la méthode ModifierMessage() de l'objet « lg ». Passez-lui la chaîne de caractères « monTexte » en paramètre.

Modifier le code. Compiler, exécuter et faire valider l'application par le professeur.

Modifier les effets d'affichages du texte

Dans cette partie, nous allons utiliser quelques méthodes de la classe **Ligne** qui permettent de modifier les effets de chargement, d'affichage de temporisation et de sortie du texte.

Les effets de chargement du texte

Les effets de chargement du texte peuvent être modifiés par la méthode *ModifierEffetChangement()* de la classe SNLigne. Cette méthode attend une valeur parmi celles fournies dans l'énumération nommée SNEffetChangement dont voici le diagramme de classe ci-contre :

En utilisant l'objet Ig de la classe Ligne, appeler la méthode *ModifierEffetChangement()* et passer en argument la valeur permettant un chargement du texte par le haut.

Appeler cette méthode dans votre code pour changer l'effet de chargement du texte. Compiler et exécuter votre code. Vérifier ensuite que le chargement est correct.

<<enum>>	SNEffetChangement
+ LEADING_IMMEDIATE = 0	
+ LEADING_XOPEN	
+ LEADING_CURTAIN_UP	
+ LEADING_CURTAIN_DOWN	
+ LEADING_SCROLL_LEFT	
+ LEADING_SCROLL_RIGHT	
+ LEADING_VOPEN	
+ LEADING_VCLOSE	
+ LEADING_SCROLL_UP	
+ LEADING_SCROLL_DOWN	
+ LEADING_HOLD	
+ LEADING_SNOW	
+ LEADING_TWINKLE	
+ LEADING_BLOCK_MOVE	
+ LEADING_RANDOM	
+ LEADING_HELLO_WORLD	
+ LEADING_WELCOME	
+ LEADING_AMPLUS	

Les autres effets sur le texte

Il existe d'autres effets possibles sur le texte : l'effet d'affichage, la modification du temps d'affichage et l'effet de sortie. Pour chacun de ces effets, une énumération a été mise au point. Vous trouverez l'ensemble de ces énumérations autour de la classe SNLigne dans le diagramme de classe se trouvant en annexe.

Modifier l'effet d'affichage pour que le texte soit clignotant. Compiler, exécuter et tester votre code.

Modifier la temporisation d'affichage pour que le texte reste affiché 10s. Compiler, exécuter et tester votre code.

Modifier l'effet de sortie pour que le texte disparaisse par le bas. Compiler, exécuter et tester votre code.

TP – Défi 9 – Les résultats de la calculatrice sur le journal lumineux

Les objectifs

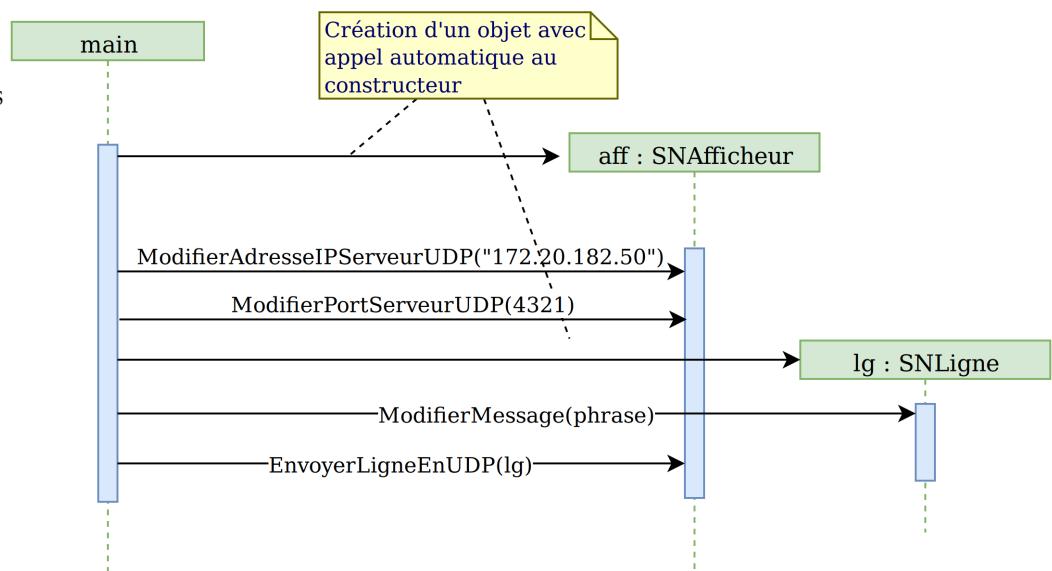
L'objectif de cette partie est d'afficher les calculs saisis par le professeur de mathématiques sur le journal lumineux. Le message à afficher sera transmis en UDP au serveur tournant sur la Raspberry PI. Ce serveur enverra le message reçu au journal lumineux.

La calculatrice et le journal lumineux

Vous allez repartir maintenant dans le code de la calculatrice tel que vous l'avez laissé à la fin du TP – Défi 1.

L'objectif sera d'envoyer en UDP la chaîne de caractères « phrase » à la Raspberry. Le diagramme de séquence suivant explique comment procéder :

Traduire en C++ le diagramme de séquence ci-contre.



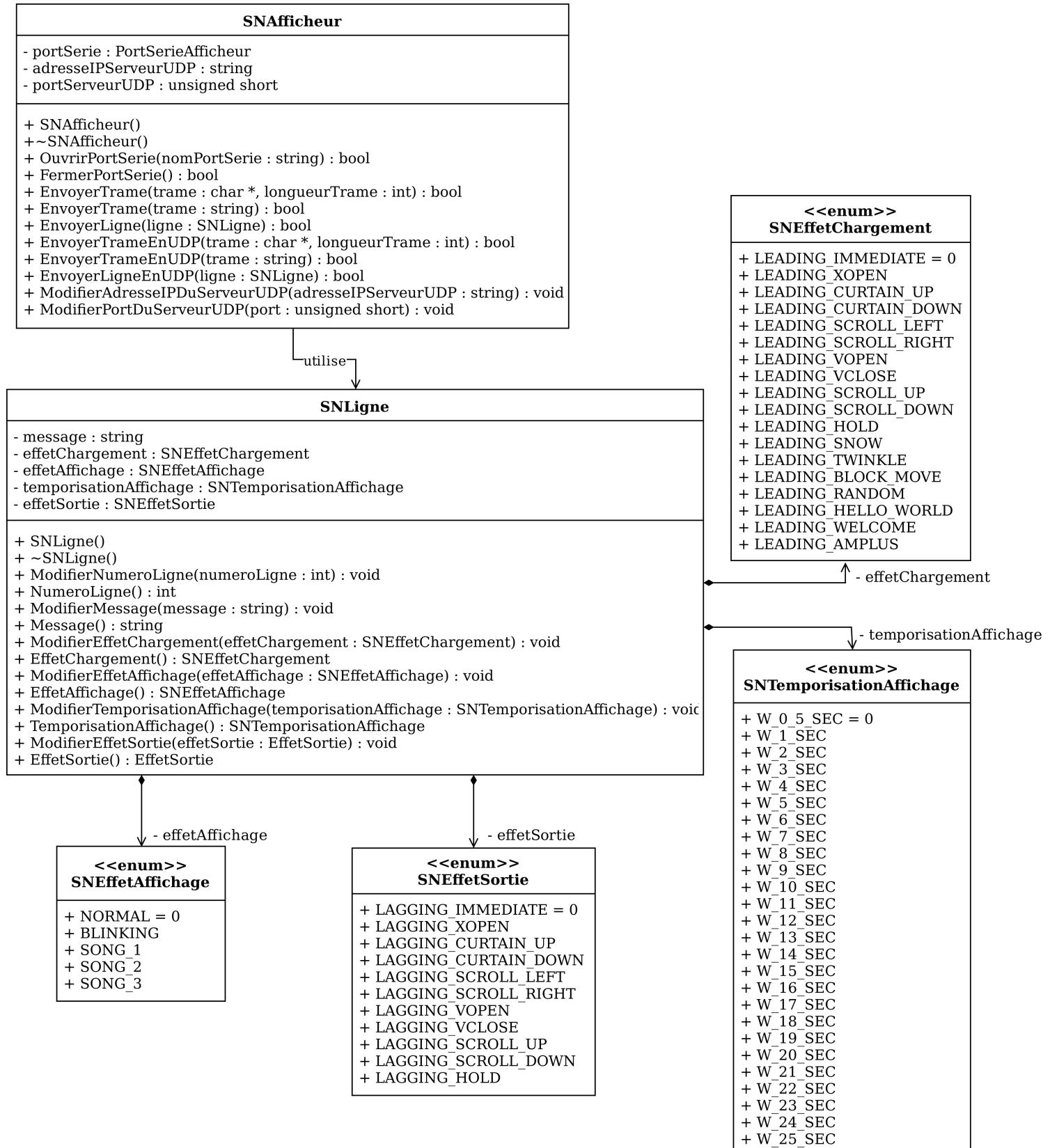
Ajouter ce code à la fin de la fonction principale, juste après l'enregistrement de « phrase » dans le fichier log.

Ajouter dans votre projet C++ les fichiers sources `SNAfficheur.h`, `SNAfficheur.cpp`, `SNLigne.h`, `SNLigne.cpp`, `PortSerieAfficheur.h` et `PortSerieAfficheur.cpp`. Ne pas oublier pas les inclusions.

Compiler, exécuter et tester votre code. Utiliser Wireshark pour vérifier que le message transite correctement sur le réseau.

Bonus : Proposer un menu permettant au professeur de mathématiques de changer les effets du texte avant l'envoi vers l'afficheur.

Annexe 1 – Diagramme de classe complet des classes Afficheur et Ligne



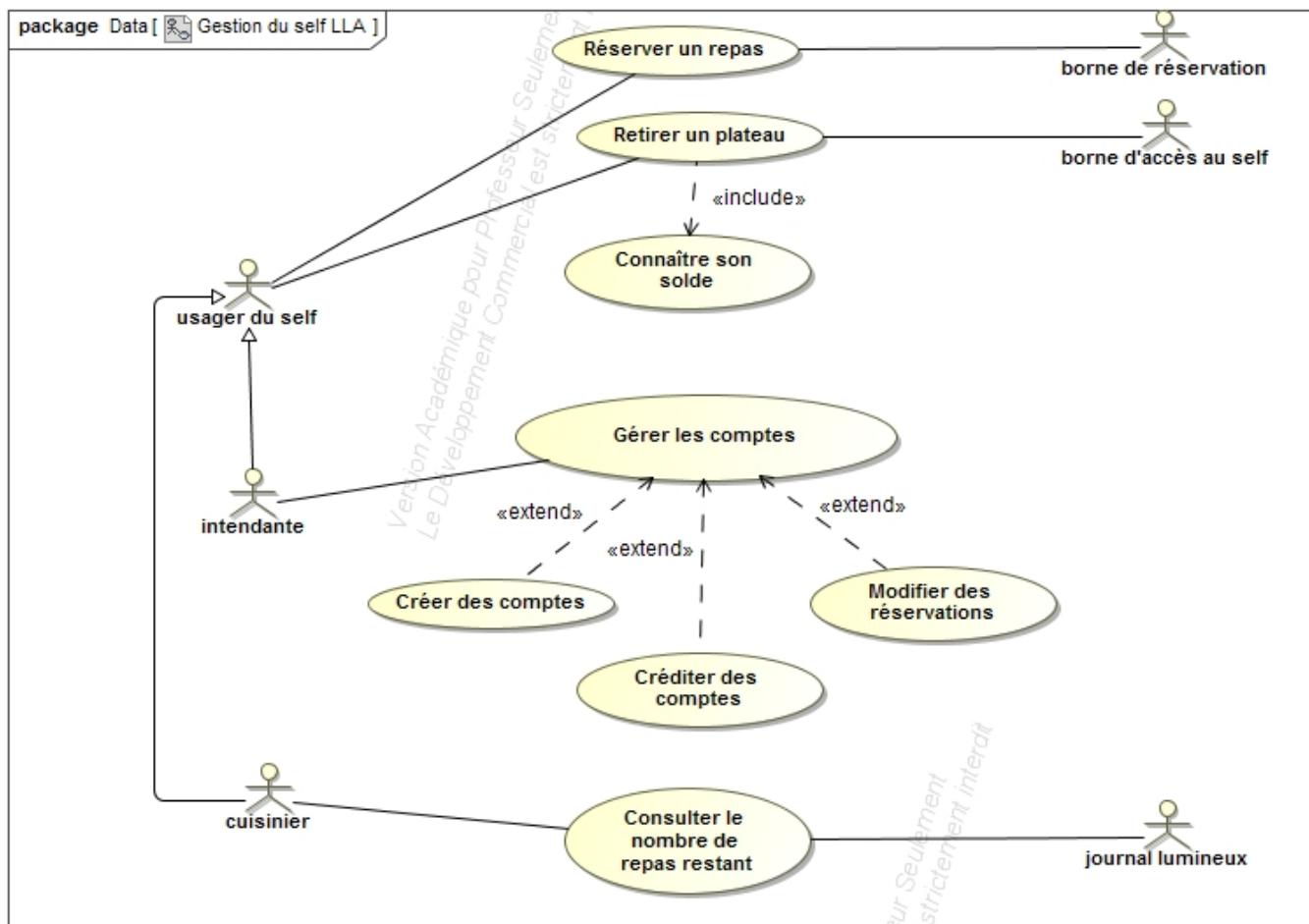
Annexe 2 – UML - Guide de mise en place du diagramme de cas d'utilisation

La procédure de mise en place

Voici la liste des étapes à suivre pour créer un diagramme de cas d'utilisation :

- Faire la liste des acteurs (humains et machines). Les PC sont sous-entendus !
- Faire la liste des services rendus à l'utilisateur du système informatique complet.
- Ajouter les associations (traits continus sans flèche) entre acteurs et services.
- Ajouter les associations (traits pointillés avec une flèche) entre les services :
 - « include » pour un service incluant un autre service,
 - « extend » pour un service ajoutant une extension (option) à un autre service.

Exemple de diagramme de cas d'utilisation



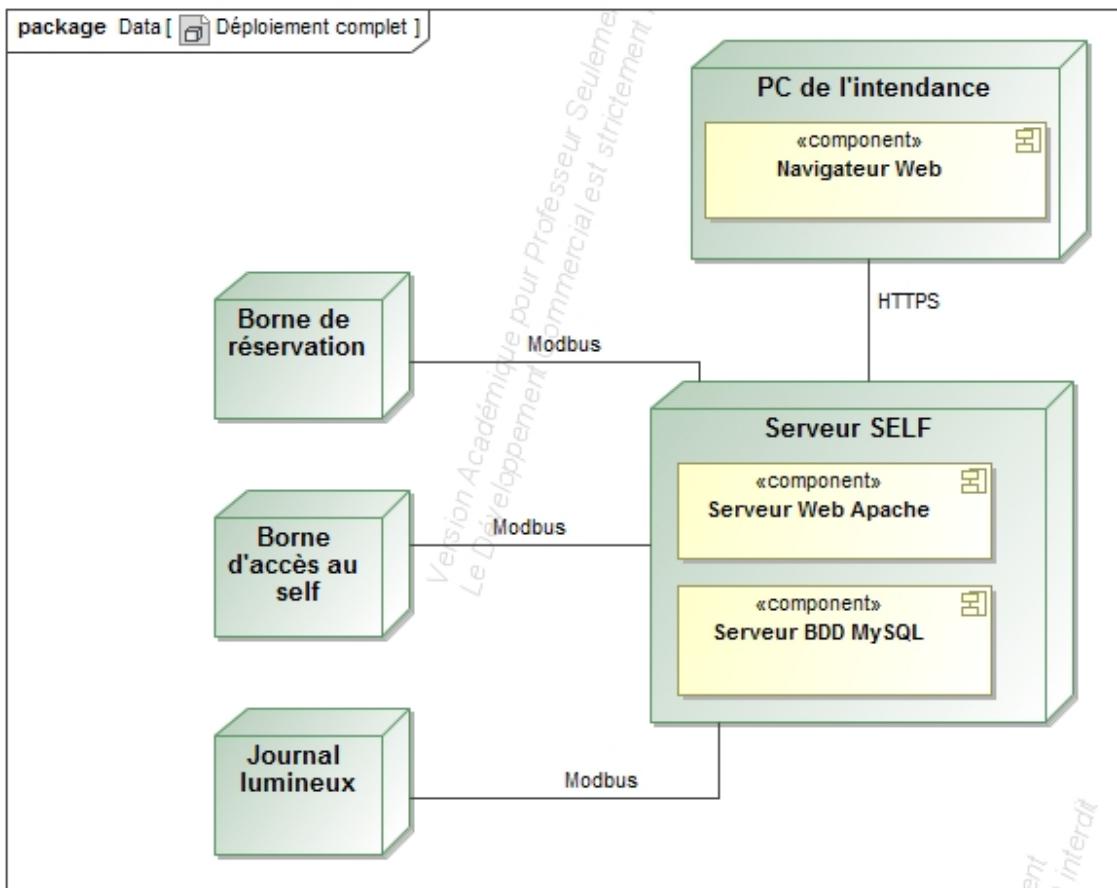
Annexe 3 – UML - Guide de mise en place du diagramme de déploiement

La procédure

Pour dessiner le diagramme de déploiement, suivre les étapes suivantes :

- Faire la liste du matériel nécessaire au fonctionnement du système informatique : ici les ordinateurs sont présents.
- Ajouter les associations (traits continus sans flèche) entre les matériels.
- Ajouter sur les associations le nom des protocoles utilisés.

Un exemple de diagramme de déploiement



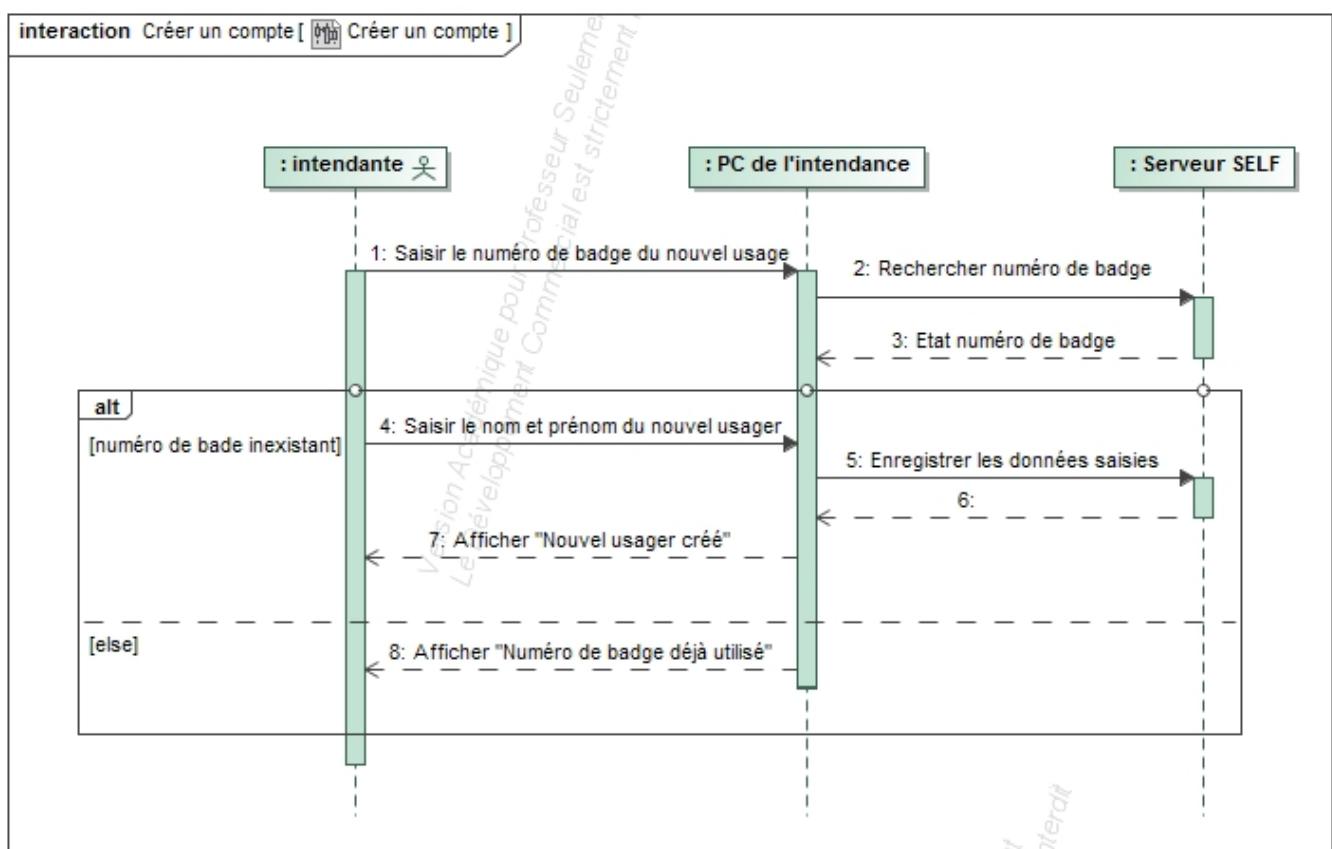
Annexe 4 – UML - Guide de mise en place du diagramme de séquence "système" :

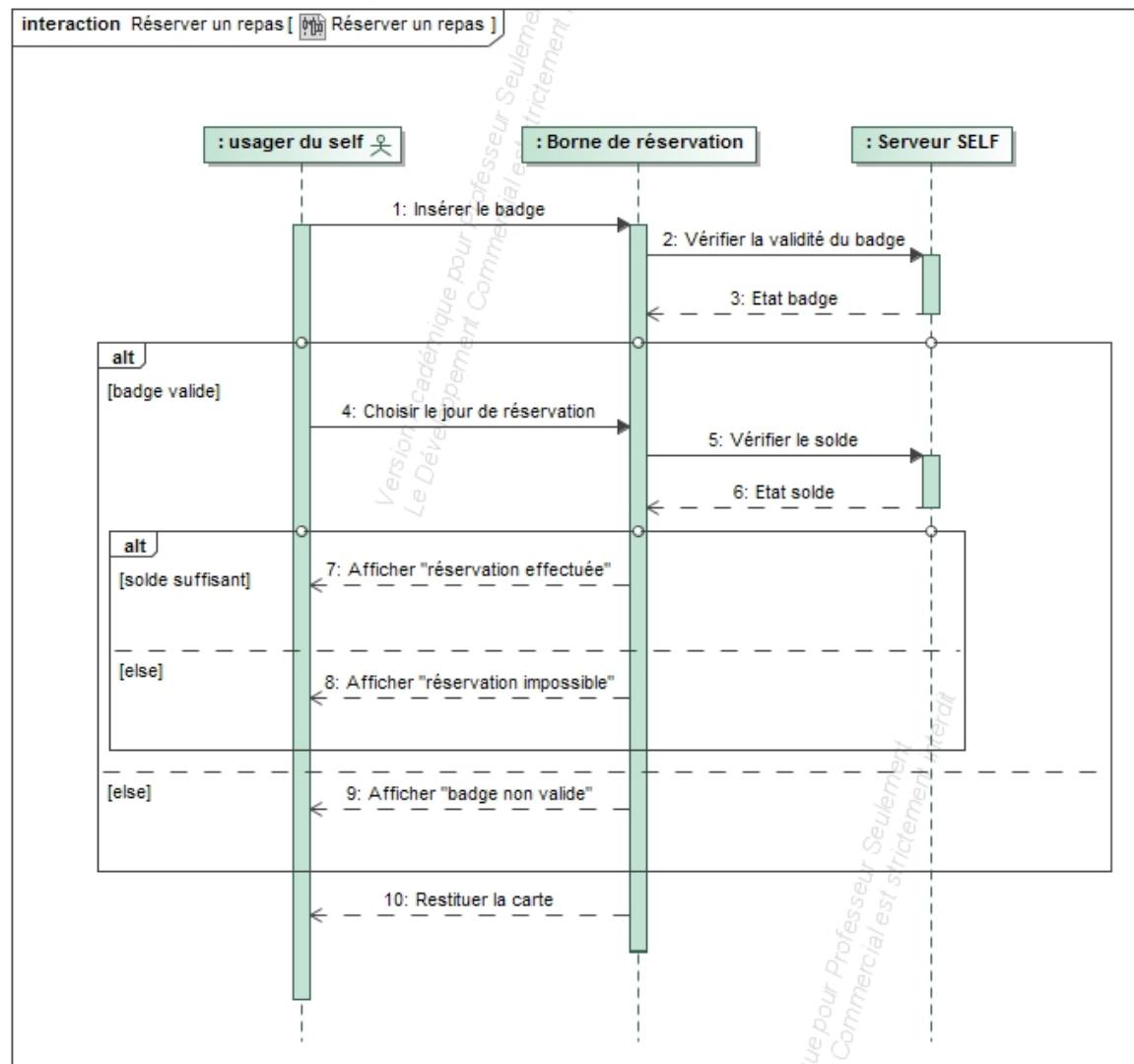
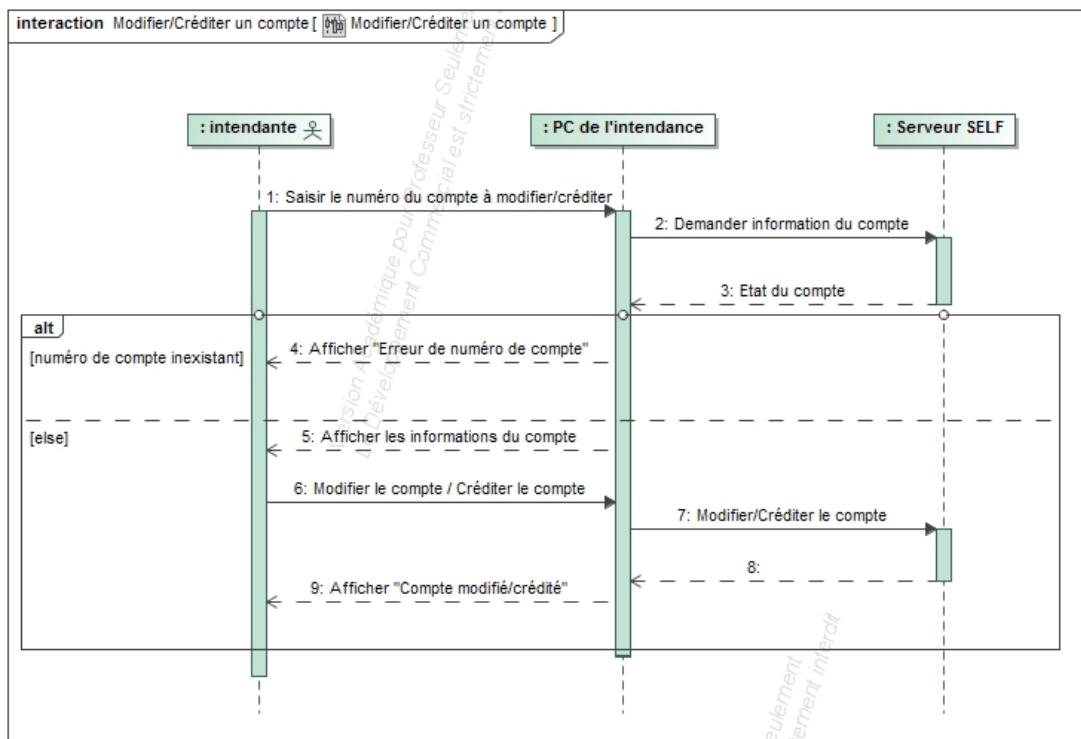
La procédure

Pour dessiner le diagramme de séquence « système », suivre les étapes suivantes :

- Présenter en haut du diagramme les acteurs du cas d'utilisation et/ou les matériels du diagramme de déploiement. L'utilisateur du système est placé à gauche.
- Ajouter chronologiquement les informations, les messages et les réponses envoyées d'un objet vers l'autre. Les réponses sont en pointillés.
- Utiliser des structures algorithmiques :
 - LOOP = boucle
 - ALT = si/sinon
 - OPT = si
 - $[b > 1]$ est une condition

Des exemples de diagrammes de séquence





Glossaire

Attribut

Un attribut est une variable ou un objet composant une classe.

Classe

Une classe sert de plan pour créer des objets, elle est composée d'attributs et de méthodes.

Événement

Un événement est une méthode appelée automatiquement.

Fonction

Une fonction est un sous-programme, une suite d'instructions, de calculs.

Une fonction est caractérisée par les () à l'intérieur desquelles on place *éventuellement* les variables d'entrée nécessaires au traitement à effectuer.

Lors de sa définition, le nom de la fonction est précédée par le type de la variable résultat.

Méthode

Une méthode est une fonction composant une classe (on l'appelle aussi opération).

Objet

Un objet est l'instance d'une classe, il est associé à une zone de données, c'est à dire à des valeurs stockées en mémoire.

Un objet permet d'appeler une méthode déclarée dans sa classe : l'objet "lance" l'opération.

Un . ou une -> permet à un objet d'accéder à une de ses propriétés (attributs) ou de lancer une de ses méthodes : repérée par les () .

Type

Le type d'une variable permet de spécifier son format et sa taille : soit un entier (int sur 32 bits, short sur 16 bits, char sur 8 bits), soit un réel (double sur 64 bits et float sur 32 bits) ou un booléen(bool).

Variable

Une variable est une zone de données : sa taille est donnée par son type.



Module Système 2

Commande d'un système d'éclairage de scène



Table des matières

Objectifs, matériels et ressources.....	2
TD1 – Cahier des charges et analyse UML.....	3
TD2 – Etude de la norme et de la documentation du matériel.....	5
TP – Défi 1 – Envoi d'une trame fixe en mode console : client TCP.....	6
TD3 – Analyse des relevés Wireshark : TCP et UDP.....	8
TD4 – MaClasse	9
TD5 – MaClasse et Chiffrement : composition	10
TP – Défi 2 – classe DMXTCP en mode console.....	12
TP – Défi 3 – interface console complète - 4 modes -	13
TP – Défi 4 – console virtuelle simple : interface graphique.....	14
TP – Défi 5 – console virtuelle spécialisée SPOTEX15 : interface graphique et gestion des événement souris..	16
TD6 – classe spécialisée Spotex15.....	18
Annexe 1 : norme DMX.....	20
Annexe 2 : SPOTEX15.....	22
Annexe 3 : RAD C + + Builder.....	23
Annexe 4 : Relevés Wireshark.....	29
Annexe 5 : Passerelle TCP / UDP.....	31

Objectifs, matériels et ressources

Objectifs

Installation d'un système d'éclairage de scène et son logiciel de commande via un boîtier Ethernet/DMX. Découverte de la norme DMX et du matériel d'éclairage de scène : étude de documents constructeurs. Modélisation UML (cas d'utilisation, déploiement, séquence) du système. Création, à partir d'un diagramme de classes, d'un logiciel de pilotage adapté à un matériel DMX particulier. Mise en place d'une application client/serveur en TCP/IP intégrant la gestion de la souris, utilisation du sniffer réseau Wireshark.

- Tableau en langage C, notion de trame.
- Diagramme de classe, codage d'une classe
- Etude de documents constructeurs
- Codage d'un client TCP
- Wireshark
- Environnement RAD (Rapid Application Development)
- Interface graphique
- Normes DMX, Artnet

Matériels



- Spot motorisé SPOTEX15
- ou Scanner NINO
- ou Rampe à 4 faisceaux
- ou Spot 64 leds
- ou Laser Club500RGBII/III
- ENTTEC Open DMX Ethernet



Ressources

- norme_dmx512_r3.pdf
- LaserClub500RGBII/rgb3.pdf
- RampeBeam-4x10qc_manuel-V1.pdf
- ScanerNINO_manuel-v1.0.pdf
- Spot_64LEDCC.pdf
- SPOTEX-15.pdf
- SPOTEX-15canauxModifiés2019.pdf
- ENTTEC70305.pdf
- Images Modules système.zip

Codes sources

- SNClientTCP.cpp
- SNClientTCP.h

Logiciels

- C++ Builder
- serveurTCPClientEnttec.exe

Annexes

- norme DMX
- SPOTEX15
- RADC++Builder
- Relevés WireShark

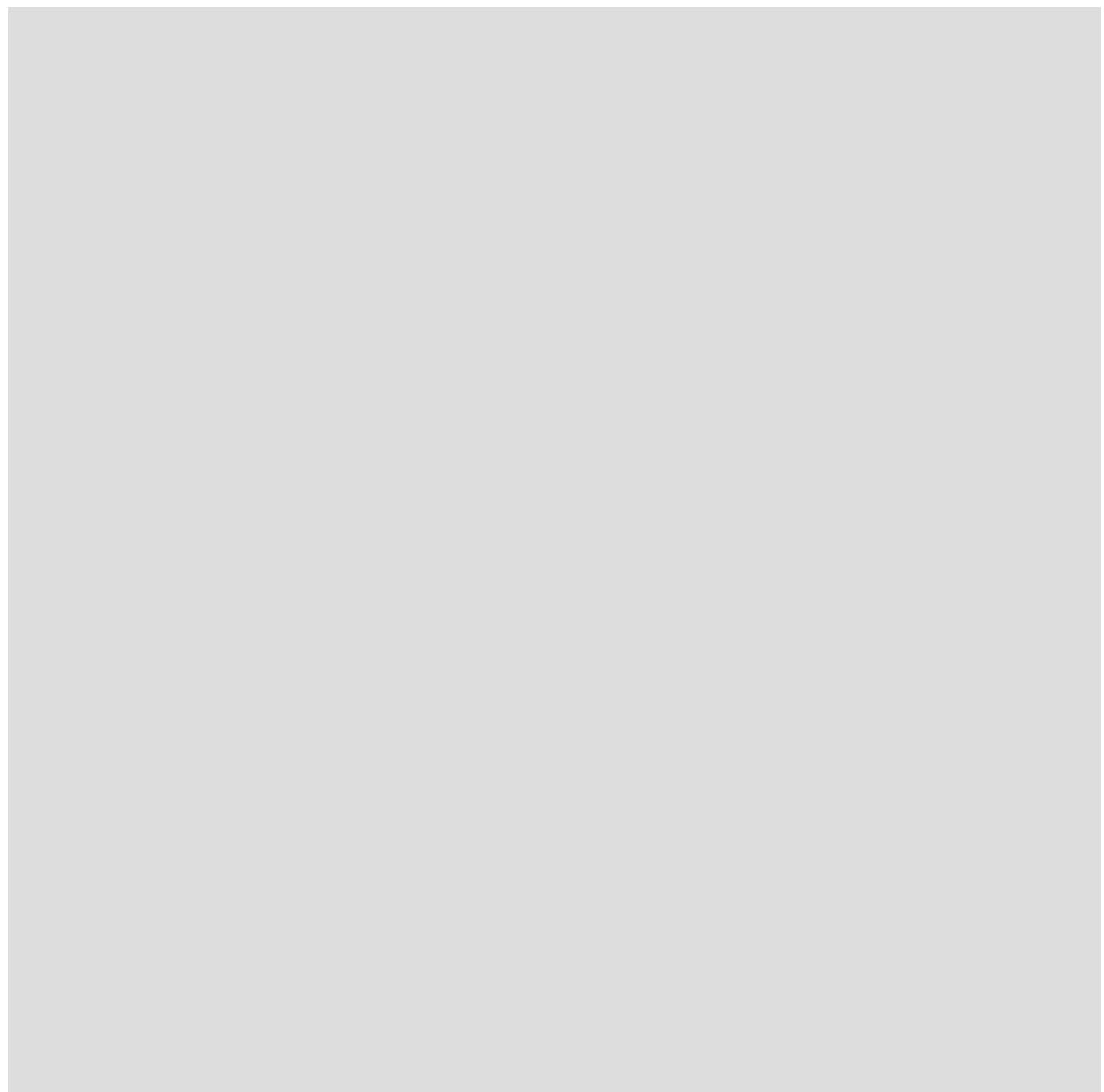
TD1 – Cahier des charges et analyse UML

Le cahier des charges

Un système d'éclairage de scène basé sur plusieurs spots à leds ou motorisés, rampes, scanners et lasers est piloté via un boîtier Ethernet/DMX. Le pilotage se fait via une console virtuelle DMX installée sur un PC ou sur une tablette. L'éclairagiste doit pouvoir piloter facilement chaque matériel DMX en direct, mais aussi programmer des scènes qui seront rejouées en mode automatique. Une interface WEB sécurisée permettra de lancer les scènes sur tous les « univers » interconnectés par un bus Artnet. Un ensemble de caméras permettra de fournir des vidéos des différentes scènes.

Les diagrammes de cas d'utilisation, de déploiement et de séquence

Dessiner le diagramme de cas d'utilisation :





Dessiner le diagramme de déploiement :



Dessiner le diagramme de séquence d'un point de vue système :

TD2 – Etude de la norme et de la documentation du matériel

?

A partir de la norme DMX donnée en annexe, donner la composition d'une trame DMX.

?

Quelles sont les durées maximum et minimum d'une trame DMX.

?

D'après la documentation du SPOTEX15 donnée en annexe (version 5 canaux), donner la composition de la trame permettant les effets suivants :

PAN à mis parcours, TILT au maximum, clignotement rapide d'une fleur rose.

PAN au minimum, TILT à mis parcours, motif "éclairs" Arc-en-ciel avec 50% de lumière.

En étudiant la documentation de chacun des appareils DMX disponibles sur le réseau, compléter le tableau :

Appareil DMX	Nombre de canaux

?

En étudiant la documentation d'un des matériels DMX utilisés (SCANNER Nino, SPOT64LED, ou RAMPE), remplir la fiche suivante :

APPAREIL :		
Canal	Valeur	Effet

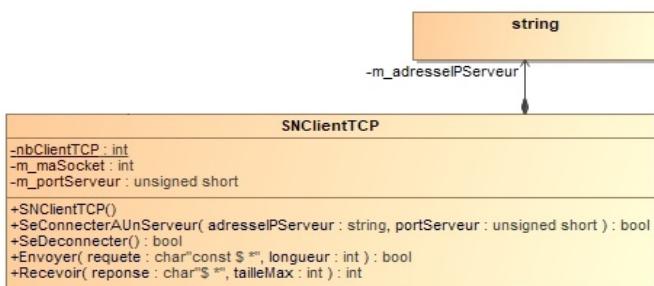
TP – Défi 1 – Envoi d'une trame fixe en mode console : client TCP

Le boîtier Ethernet/DMX est piloté en UDP, un serveur TCP/IP (serveurTCPClientEnttec.exe) permet de convertir les trames TCP en trame UDP pour le boîtier : c'est aussi un client UDP.

? Quelles sont les différences entre un client et un serveur ?

? Quelle est la différence entre UDP et TCP ?

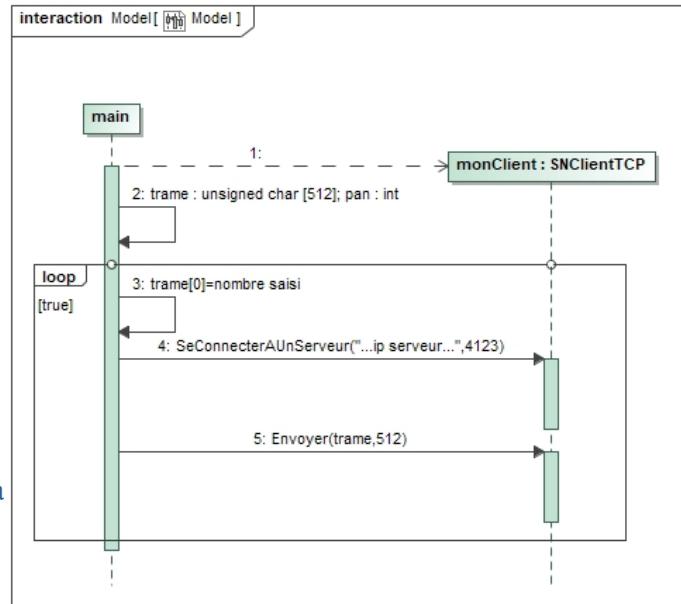
La classe SNClientTCP est donnée :



? Coder le diagramme de séquence donné, afin d'envoyer la trame de votre choix.

? Votre code :

Vérifier les effets lumineux.





Analyser à l'aide de Wireshark la trame TCP envoyée au serveur TCP, expliquer les 6 trames en détail, préciser les adresses mac et les ports. Le filtre à utiliser est :

`ip.addr == 172.20.182.229 || (ip.addr == 172.20.100.245 && tcp) || (ip.addr == 172.20.101.1 && tcp)`

172.20.182.229 est l'adresse du ENTTEC // à modifier

172.20.100.245 est l'adresse du serveur TCP / Client UDP // à modifier

172.20.101.1 est l'adresse du client TCP (votre logiciel) // à modifier



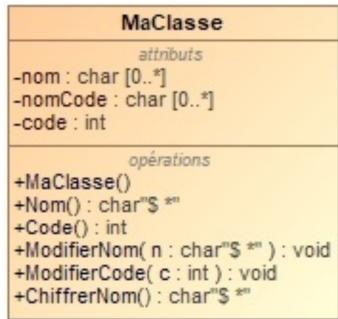
Analyser à l'aide de Wireshark la trame UDP envoyée par le serveur TCP (passerelle TCP/UDP) au boîtier ENTTEC, expliquer les 6 trames en détail, préciser les adresses mac et les ports.

TD3 – Analyse des relevés Wireshark : TCP et UDP

En se basant sur les relevés Wireshark donnés en annexe, proposer un diagramme UML de séquence visant à décrire (lors de l'envoi d'un effet) les échanges entre le client TCP, la passerelle TCP/UDP, le boîtier Ethernet/DMX, et le spot.

TD4 – MaClasse

Relier par des flèches les éléments du code de la déclaration de MaClasse (.h) et le diagramme de classe.



```

class MaClasse
{
    private:
        char nom[10];
        char nomCode[10];
        int code;

    public:
        MaClasse();
        char* Nom();
        int Code();
        void ModifierNom(char* n);
        void ModifierCode(int c);
        char* ChiffrerNom();
}

```

Pour chaque méthode ci-contre (.cpp), donner :

Nom méthode	Type retour	Nb arguments	Types arguments
-------------	-------------	--------------	-----------------

A quoi sert le mot clé **return** ?

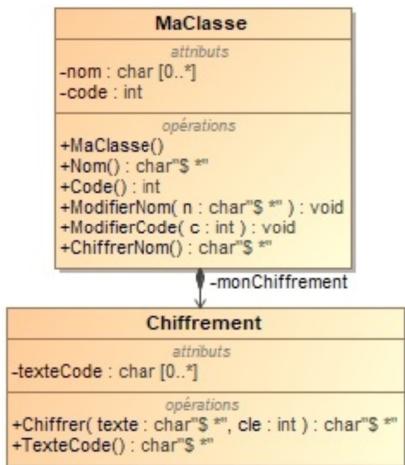
Qu'est ce qu'un constructeur ?

Quel est l'effet de la fonction **strcpy** ?

Proposer le code du programme principal : création de l'objet, appel des méthodes **ModifierNom()**, **ModifierCode()**, **ChiffrerNom()** qui retourne un tableau affichable (avec **cout**).

TD5 – MaClasse et Chiffrement : composition

Relier par des flèches les éléments du code de la déclaration de **MaClasse (.h)** et le diagramme de classe.



```

class MaClasse
{
    private:
        char nom[10];
        int code;
        Chiffrement monChiffrement;

    public:
        MaClasse();
        char* Nom();
        int Code();
        void ModifierNom(char* n);
        void ModifierCode(int c);
        char* ChiffrerNom();
};

class Chiffrement
{
    private:
        char texteCode[10];

    public:
        char* TexteCode();
        char* Chiffrer(char* texte, int cle);
};
  
```

Pour chaque méthode ci-dessous (.cpp), donner :

Nom méthode	Type retour	Nb arguments	Types arguments
-------------	-------------	--------------	-----------------

<pre> char* Chiffrement::Chiffrer(char* texte, int cle) { if(!cle) return texte; char cle8bits=cle%256; for(int i=0;i<strlen(texte);i++) { texteCode[i]=texte[i]^cle8bits; } texteCode[strlen(texte)]=0; return texteCode; } char* Chiffrement::TexteCode() { return texteCode; } </pre>	<pre> MaClasse::MaClasse() { strcpy(nom,"-"); code=0; } char* MaClasse::Nom() { return nom; } int MaClasse::Code() { return code; } void MaClasse::ModifierNom(char* n) { strcpy(nom,n); } void MaClasse::ModifierCode(int c) { code=c; } char* MaClasse::ChiffrerNom() { return monChiffrement.Chiffrer(nom,code); } </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Repérer (entourer et flécher) les objets et les méthodes appelés.

?

Proposer le code du programme principal permettant de tester la classe Chiffrement seule.

```
int main()
{    //créer un objet 'c' de la classe Chiffrement
```

```
//Appeler la méthode Chiffrer() avec l'objet 'c' pour chiffrer "azerty" avec la clé 54321
```

```
//Afficher le texte chiffré
```

```
}
```

?

Proposer le code du programme principal permettant d'afficher le mot "secret" chiffré avec la clé 13 en utilisant la classe MaClasse.

TP – Défi 2 – classe DMXTCP en mode console

A partir du diagramme de classe donné, coder la déclaration de la classe DMXTCP (.h).

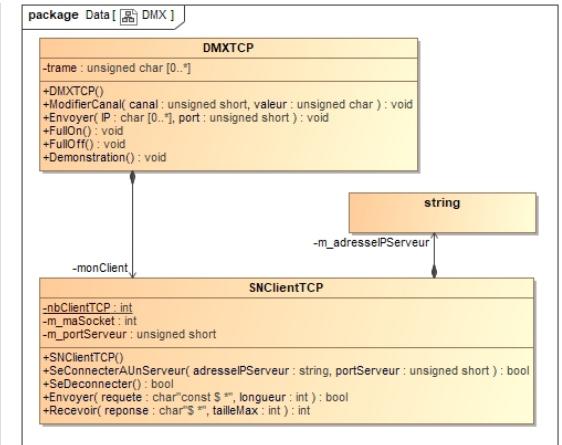


Votre code :

```
class DMXTCP {
public:
    -trame : unsigned char [0..]
    +DMXTCP()
    +ModifierCanal( canal : unsigned short, valeur : unsigned char ) : void
    +Envoyer( IP : char [0..], port : unsigned short ) : void
    +FullOn() : void
    +FullOff() : void
    +Demonstration() : void
};

class SNCientTCP {
public:
    -nbClientTCP : int
    -m_maSocket : int
    -m_portServeur : unsigned short
    +SNCientTCP()
    +SeConnecterAUUnServeur( adresseIPServeur : string, portServeur : unsigned short ) : bool
    +SeDeconnecter() : bool
    +Envoyer( requeste : char const $ "", longueur : int ) : bool
    +Recevoir( reponse : char $ "", tailleMax : int ) : int
};

string m_adresseIPServeur;
```



Coder la définition des méthodes (.cpp) :

Constructeur : initialiser les 512 octets de trame[] à 0 (boucle for).

ModifierCanal : l'argument valeur est stockée dans la case [canal-1] de la trame.

Envoyer : se connecter au serveur, puis envoyer la trame de 512 octets.

Les méthodes FullOff(), FullOn() et Demonstration() seront codées ultérieurement.



Votre code :

```
class DMXTCP {
public:
    -trame : unsigned char [0..]
    +DMXTCP()
    +ModifierCanal( canal : unsigned short, valeur : unsigned char ) : void
    +Envoyer( IP : char [0..], port : unsigned short ) : void
    +FullOn() : void
    +FullOff() : void
    +Demonstration() : void
};

class SNCientTCP {
public:
    -nbClientTCP : int
    -m_maSocket : int
    -m_portServeur : unsigned short
    +SNCientTCP()
    +SeConnecterAUUnServeur( adresseIPServeur : string, portServeur : unsigned short ) : bool
    +SeDeconnecter() : bool
    +Envoyer( requeste : char const $ "", longueur : int ) : bool
    +Recevoir( reponse : char $ "", tailleMax : int ) : int
};

string m_adresseIPServeur;

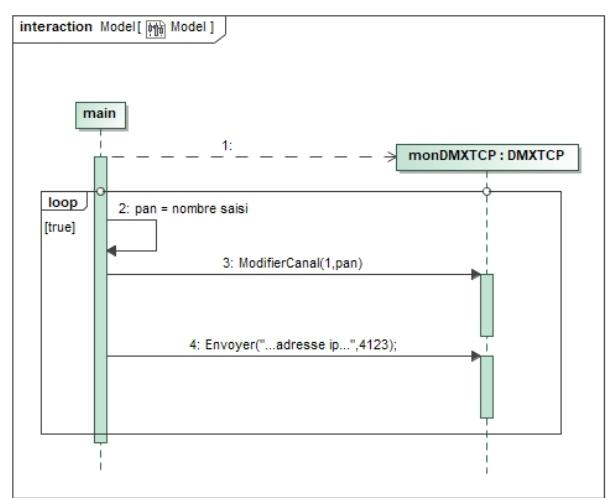
DMXTCP monDMXTCP;
```

Coder le programme principal correspondant au diagramme de séquence donné : la valeur du PAN sera saisie au clavier. Vérifier les effets de votre code sur le SPOTEX15.



Votre code :

```
main
    1: --> monDMXTCP : DMXTCP
loop [true]
    2: pan = nombre saisi
    3: ModifierCanal(1,pan)
    4: Envoyer("...adresse ip...",4123);
```



Ajouter la possibilité de saisir et de modifier la valeur du TILT. Vérifier les effets de votre code sur le SPOTEX15.

TP – Défi 3 – interface console complète - 4 modes -

 Coder les méthodes FullOff() (les 512 octets sont mis à 0) et FullOn() (les 512 octets sont mis à 255).



Le code de FullOff() :

 Coder le programme principal permettant de tester ces méthodes. Vérifier les effets de votre code sur le SPOTEX15.

 Coder la méthode Demonstration() permettant d'envoyer 512 octets aléatoires :

strand(time(0)); permet d'initialiser la fonction random (aléatoire).

rand() renvoie un nombre aléatoire (à stocké dans chaque octet de la trame).



Votre code :

 Coder le programme principal permettant de tester cette méthode. Vérifier les effets de votre code sur le SPOTEX15.

 Coder l'interface console complète, en ajoutant le menu suivant :

MANUEL (1), FULLON (2), FULLOFF (3), DEMO (4)

MANUEL permet de saisir PAN et TILT,

DEMO permet de rentrer dans une boucle infinie while(true){...}

cadencée par la fonction Sleep(1000) pour une temporisation de 1000ms.



Le code du programme principal complet :

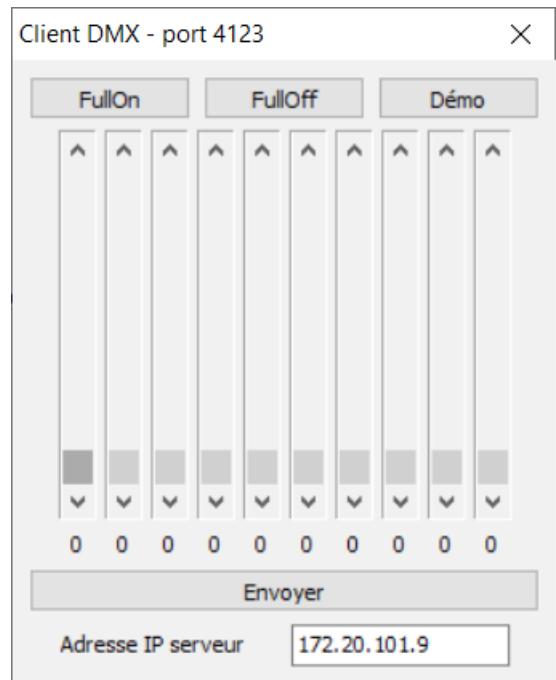
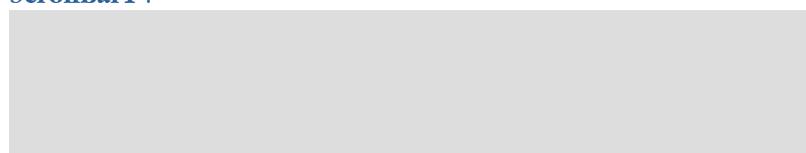
TP – Défi 4 – console virtuelle simple : interface graphique

Créer une nouvelle application sous C++ Builder : l'annexe décrit l'interface du RAD C++ Builder. L'interface désirée est donnée ci-dessous.

Placer Buttons, Labels, ScrollBars (mode horizontal) et Edit dans l'interface. Dans le code source Unit1.h, ajouter un objet monDMX de la classe DMXTCP, dans les attributs privés de la classe TForm1 (la classe qui gère l'IHM - Interface Homme Machine), penser aux #include. Sauvegarder et compiler le projet.

Générer l'événement OnChange de l'objet ScrollBar. Dans le code de cet événement : affecter à la propriété **Caption** de chaque **Label** : (255 - position du ScrollBar) -par défaut la valeur maximum du ScrollBar est en bas-. Pour accéder à une propriété taper **Label->** (pour le Label1), **ScrollBar1->** (pour le ScrollBar1) : **un menu déroulant s'affiche.** Sauvegarder et compiler le projet.

Le code complet (avec entête) de l'événement **OnChange** de l'objet **ScrollBar1** :

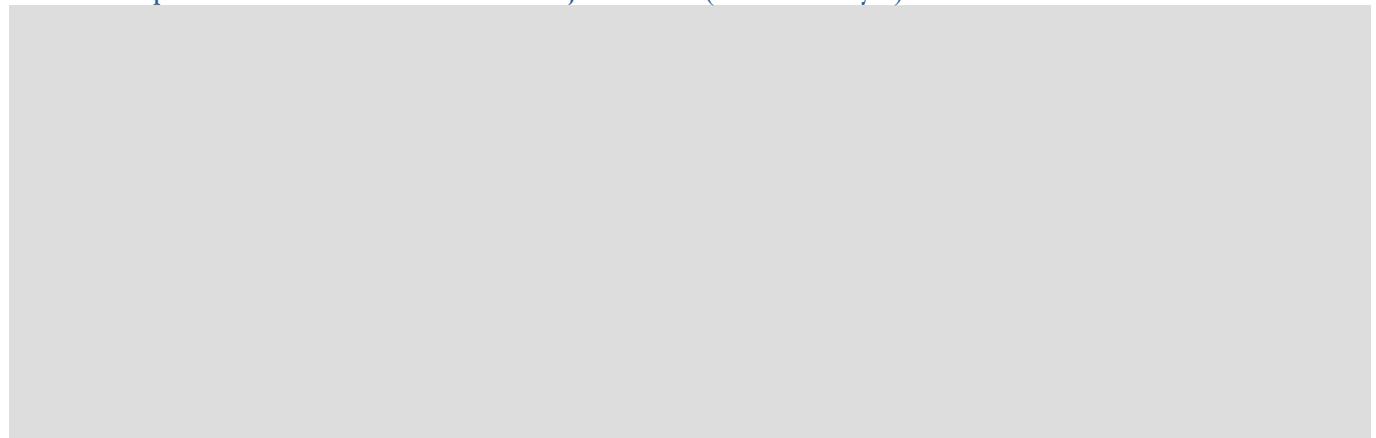


Utiliser l'objet monDMX afin de lancer la méthode ModifierCanal : passer en argument le numéro du canal et la valeur (255 - ScrollBar1 ->Position, pour le premier canal). **monDMX. permet d'accéder aux méthodes de l'objet monDMX.** Sauvegarder et compiler le projet.

A chaque modification du code, il faut construire le projet (**SHIFT F9**) et résoudre les erreurs. Dans le cas contraire, les menus déroulants ne s'affichent plus, le codage est ralenti et perd en efficacité, les erreurs s'accumulent...

Générer l'événement OnClick du bouton Envoyer. Utiliser l'objet monDMX afin de lancer la méthode Envoyer en passant en argument l'adresse IP saisie dans la zone Edit (une conversion est nécessaire, il faut saisir à la place de l'adresse : **AnsiString(Edit1 ->Text).c_str(), puis le numéro du port (4123). Tester les fonctionnalités du logiciel que vous venez de créer.**

Le code complet de l'événement **OnClick** de l'objet Button4 (bouton Envoyer) :



Modes FullOff, FullOn :

Coder les événements associés aux boutons FullOn / FullOff en appelant les bonnes méthodes à l'aide de l'objet monDMX. Tester les fonctionnalités du logiciel.

Le code complet de l'événement **OnClick** de l'objet Button1 (bouton FullOff) :

Mode Démonstration :

Ajouter à l'interface un objet **TTimer**. Dans ses propriétés : **Enabled=false**. Dans l'événement **OnTimer** appeler la méthode **Demonstration()** de l'objet monDMX. Appeler ensuite la méthode **Envoyer()** de l'objet monDMX, comme précédemment.

Le code complet de l'événement **OnTimer** de l'objet Timer1 :

Codage du bouton Démo : Activer le timer en fonction de la valeur du Caption du bouton :

Si la propriété Caption du bouton est "Démo" : la propriété Enabled du timer = true, le Caption du bouton devient "Stop".

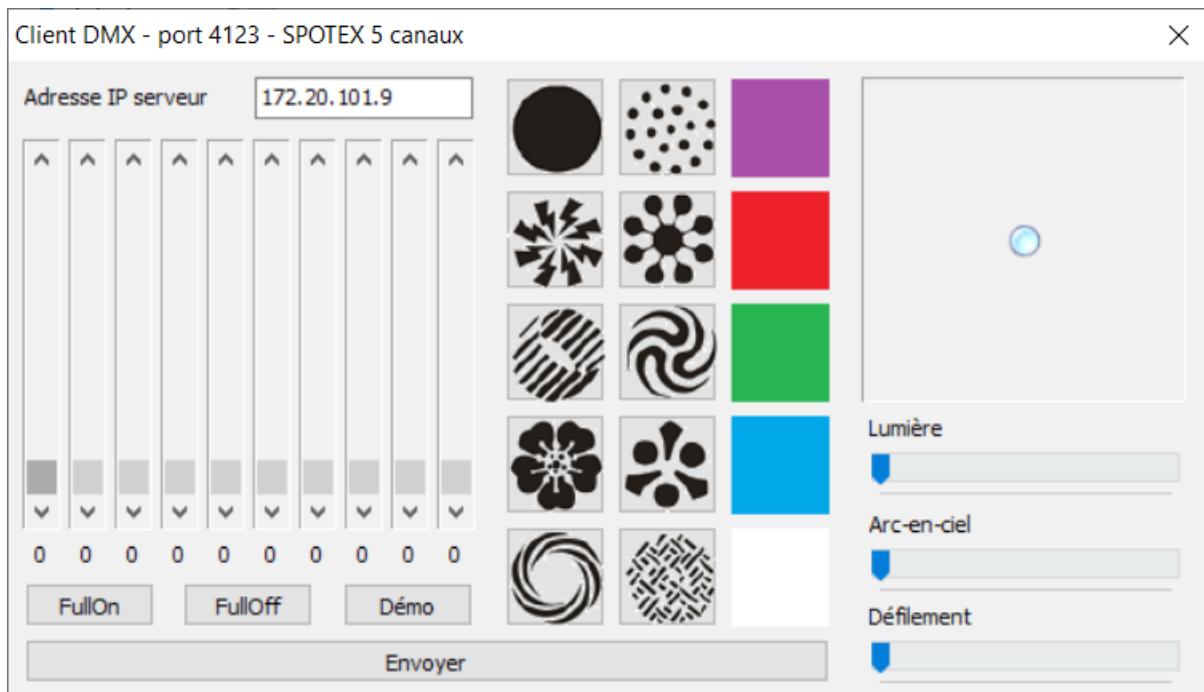
Si la propriété Caption du bouton est "Stop" : la propriété Enabled du timer = false, le Caption du bouton devient "Démo".

Remarque : les objets **Captions** sont de type **UnicodeString**, ils acceptent la comparaison directe à une chaîne de caractères à l'aide de l'opérateur **==** (contrairement aux tableaux de caractères qui nécessitent l'utilisation de la fonction **strcmp()**).

Le code complet de l'événement **OnClick** de l'objet Button3 (bouton Démo) :

TP – Défi 5 – console virtuelle spécialisée SPOTEX15 : interface graphique et gestion des événements souris

Il est demandé de modifier l'interface « console virtuelle » codée et testée lors du précédent défi, afin d'y ajouter des fonctionnalités de commande rapides pour le SPOTEX15.



Les boutons pourvus d'images sont des **SpeedButtons** (propriété **Glyph** pour l'image .bmp), les curseurs -Lumière, Arc-en-ciel, Défilement- sont des **TrackBars**, le "trackBall" virtuel est une image (**TImage**) dans un **Bevel**. Chaque **SpeedButton** doit modifier l'attribut **Position** des **ScrollBars** en fonction de l'effet demandé : le code doit être placé dans l'événement **OnClick** du bouton. Le déplacement des **TrackBars** doit également modifier l'attribut **Position** des **ScrollBars** en fonction de l'effet demandé : le code doit être placé dans l'événement **OnChange** du **TrackBar**.

Coder et tester les fonctionnalités des boutons et des TrackBars.

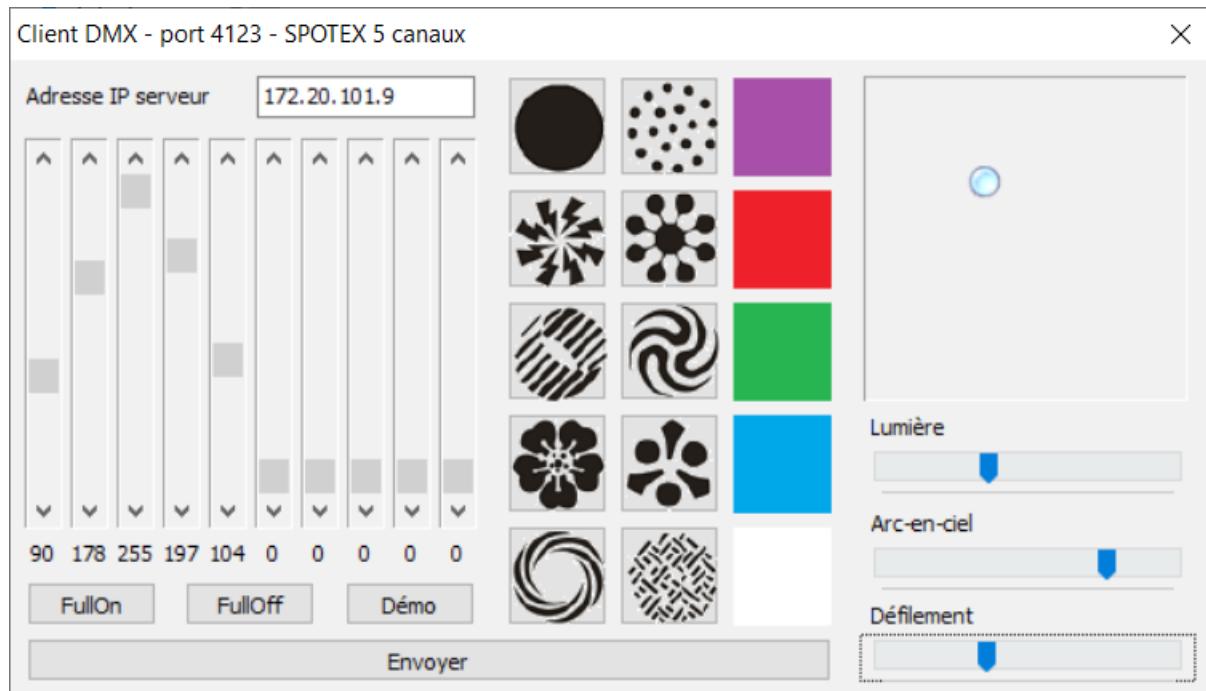
Le code complet de l'événement **OnClick** correspondant au bouton « rouge » :

Le code complet de l'événement **OnChange** de l'objet **TrackBar1** (Lumière) :

Déplacement du "trackBall" :

Ajouter à TForm1 l'attribut click, booléen. Les événements du TImage à utiliser sont :

OnMouseDown : click est positionné à true. **OnMouseUp** : click est positionné à false. **OnMouseMove** : L'événement fournit les coordonnées X et Y du déplacement de la souris sur l'image, il faut utiliser ces 2 valeurs pour déplacer l'image en conséquence et modifier la position des ScrollBars correspondant au PAN et au TILT (à condition que click soit true).



Plusieurs tests doivent être ajoutés pour que l'image ne sorte pas du Bevel.

Tester les fonctionnalités du logiciel spécialisé que vous venez de créer.

Le code complet de l'événement OnMouseMove :

Remarque : Créez un exécutable indépendant de C++ Builder. Cliquez sur le menu Project > Options :

Dans l'onglet C++ Linker, décocher : Link with Dynamic RTL.

Dans l'onglet Packages / Runtime Packages, décocher : Link with runtime packages.

TD6 – classe spécialisée Spotex15

La classe Spotex15 est composée d'un objet monDMXTCP de la classe DMXTCP. Cette classe permet de lancer des effets lumineux directement sans se soucier de la documentation du SPOTEX15.

Voici un exemple de programme principal permettant de tester la nouvelle classe :

```
int main()
{
    Spotex15 monSpotex15;
    monSpotex15.ModifierPAN(128);
    monSpotex15.ModifierTILT(0);
    monSpotex15.ModifierLuminosite(50);           // % de luminosité
    monSpotex15.ModifierMotif(2);                 // numéro du motif de 1 à 10
    monSpotex15.ModifierCouleur(14);              // numéro de la teinte de 0 à 17 :
                                                // 0 = pas de lumière, 1 = violet, 2 = rouge ...
    monSpotex15.Envoyer("172.20.100.6",4123);    // Spotex15::Envoyer() :
                                                // appel la méthode DMXTCP::Envoyer()
    return 0;
}
```

Dessiner le diagramme de classe de Spotex15 : elle comportera 6 attributs.

Donner le code de la déclaration de la classe.



Donner la définition de chaque méthode de la classe Spotex15.



Coder et tester les fonctionnalités de la classe Spotex15 : les numéros de motifs et de couleurs pourront respectivement être placés dans les **énumérations** Motifs et Couleurs.

Annexe 1 : norme DMX 1

DRAFT Standard, BSR E1.11, Entertainment Technology – USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories

1 9 Data Protocol

2 9.1 Format

3 Data transmitted shall be in asynchronous serial format. DMX512 slots shall be transmitted sequentially
4 beginning with slot 0 and ending with a slot 513. Prior to the first data
5 slot being transmitted, a Reset Sequence shall be transmitted – a BREAK, followed by a MARK AFTER BREAK,
6 and a START Code. Valid DMX512 slot values under a NULL START Code shall be 0 to 255 decimal.

7 9.2 Slot format

8 The data transmission format for each data value transmitted shall be as follows:

Table 6 - Data slot format	
Bit Position	Description
1	Start Bit, Low or SPACE
2 through 9	Slot Value Data Bits, Least Significant Bit to Most Significant Bit Positive logic
10, 11	Stop Bits, High or MARK
12	Parity
13	Not transmitted

14 9.3 Break

15 The BREAK (Timing Diagram, Designation #1) shall be defined as a high-to-low transition followed by a low-to-high transition. The BREAK indicates
16 the start of a new packet.

17 9.4 Mark After Break

18 The duration of the MARK separating the BREAK and the START Code (Timing Diagram, Designation #2) shall
19 be not less than 1 microsecond and not greater than 1 second. All DMX512 transmitters shall produce a MARK
20 AFTER BREAK of no less than 8 microseconds. All receivers shall recognize an 8 microsecond MARK AFTER
21 BREAK.

22 9.5 START Code

23 Note: The 1986 version of this standard specified a 4 microsecond MARK AFTER BREAK. The 1990
24 version of the standard changed that value to 8 microseconds, but added an option for receivers capable
25 of recognizing the 4 microsecond MARK AFTER BREAK to be identified as having that capability. Some
26 transmitters may still be in use that generate the shorter 4 microsecond MARK AFTER BREAK, and they
27 may not work with equipment built to this standard.

28 9.6 NULL START Code

29 The START Code is the first byte following a MARK AFTER BREAK. The START Code identifies the function
30 of subsequent data in that packet.

31 9.7 NULL START Code

32 The NULL START Code (a NULL byte – all zeros) identifies subsequent data as sequential 8-bit information.
33

34 41

42

43

44

45

2 9.5.2 Other START Codes

- 1 In order to provide for future expansion and flexibility, DMX512 makes provision for 255 additional non NULL START Codes (' through 255 decimal, 01 through FF hexadecimal), henceforth referred to as Alternate START Codes. Where it is required to send proprietary information over a DMX512 data link, a packet starting with a registered Alternate START Code shall be used.
- 2 Annex D states the requirements for transmitting, processing, and receiving Alternate START Codes.
- 3 Several Alternate START Codes are reserved. See Annex E.
- 4 See Annex F for Alternate START Code Registration Policies.
- 5 12
- 6 13
- 7 14
- 8 15
- 9 16
- 10 17
- 11 18
- 12 19
- 13 20
- 14 21
- 15 22
- 16 23
- 17 24
- 18 25
- 19 26
- 20 27
- 21 28
- 22 29
- 23 30
- 24 31
- 25 32
- 26 33
- 27 34
- 28 35
- 29 36
- 30 37
- 31 38
- 32 39
- 33 40
- 34 41
- 35 42
- 36 43
- 37 44
- 38 45

– 19 –

© USITT ESTA – This document is a work in progress, and may be duplicated only for the purposes of finalizing this Report. It may not be published in part or in whole or be duplicated for-profit, or sold in any manner.
CP/1988-1031r3 **DRAFT**

Printed Oct 25, 2000 10:25 p
– 20 –
© USITT ESTA – This document is a work in progress, and may be duplicated only for the purposes of finalizing this Report. It may not be published in part or in whole or be duplicated for-profit, or sold in any manner.
CP/1988-1031r3 **DRAFT**

Annexe 1 : norme DMX

DRAFT Standard, BSR E1.11, Entertainment Technology – USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories

1 9.11 Dimmer class data

2 Valid dimmer levels shall be 0 to 255 (0 FF hexdecimal) representing dimmer control input. 0 shall
 3 represent a dimmer output of OFF or minimum and 255 shall represent an output of FULL. A dimmer shall
 4 respond to increasing the DMX512 slot value for 0 to 255 by fading from its minimum level (off) to its maximum
 5 level (full). The exact relationship between DMX512 slot values and dimmer output is beyond the scope of this
 6 Standard.

7 8 9.12 Timing Diagram - output of transmitting USART

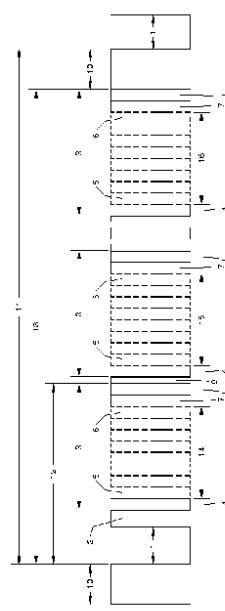


Figure Key

- 9 1 - 'SPACE' for BREAK
- 10 2 - 'MARK' After BREAK (MAB)
- 11 3 - Slot Time
- 12 4 - START Bit
- 13 5 - LEAST SIGNIFICANT Data Bit
- 14 6 - MOST SIGNIFICANT Data Bit
- 15 7 - STOP Bit
- 16 8 - STOP Bit
- 17 9 - 'MARK' Time Between slots
- 18 10 - 'MARK' Before BREAK (MBB)
- 19 11 - BREAK to BREAK Time
- 20 12 - RESET Sequence (BREAK, MAB, START Code)
- 21 13 - DMX512 Packet
- 22 14 - START CODE (Slot 0 Data)
- 23 15 - SLOT 1 DATA
- 24 16 - SLOT nn DATA (Maximum 512)

Figure 4 - Timing Diagram

- 21 -

© USITT ESTA - This document is a work in progress, and may be duplicated only for the purposes of finalizing this Report. It may not be published in part or in whole or be duplicated for-profit or sold in any manner.
 CP1998-1031r3 **DRAFT**

Printed Oct 25, 2000 10:25 p
 Printed Oct 25, 2000 10:25 p

- 22 -

© USITT ESTA - This document is a work in progress, and may be duplicated only for the purposes of finalizing this Report. It may not be published in part or in whole or be duplicated for-profit or sold in any manner.
 CP1998-1031r3 **DRAFT**

Printed Oct 25, 2000 10:25 p

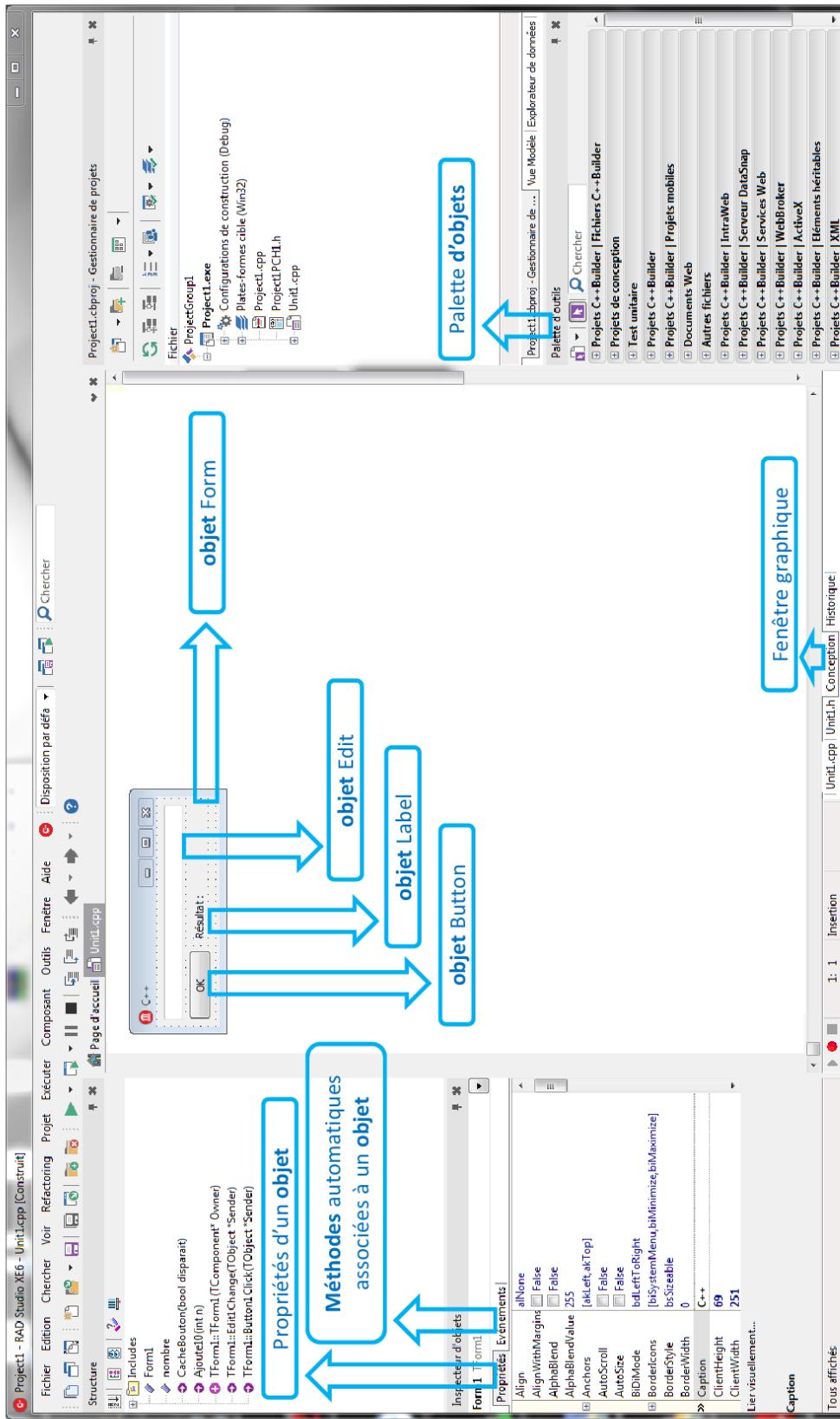
Annexe 2 : SPOTEX15

Documentation corrigée des spots SPOTEX-15

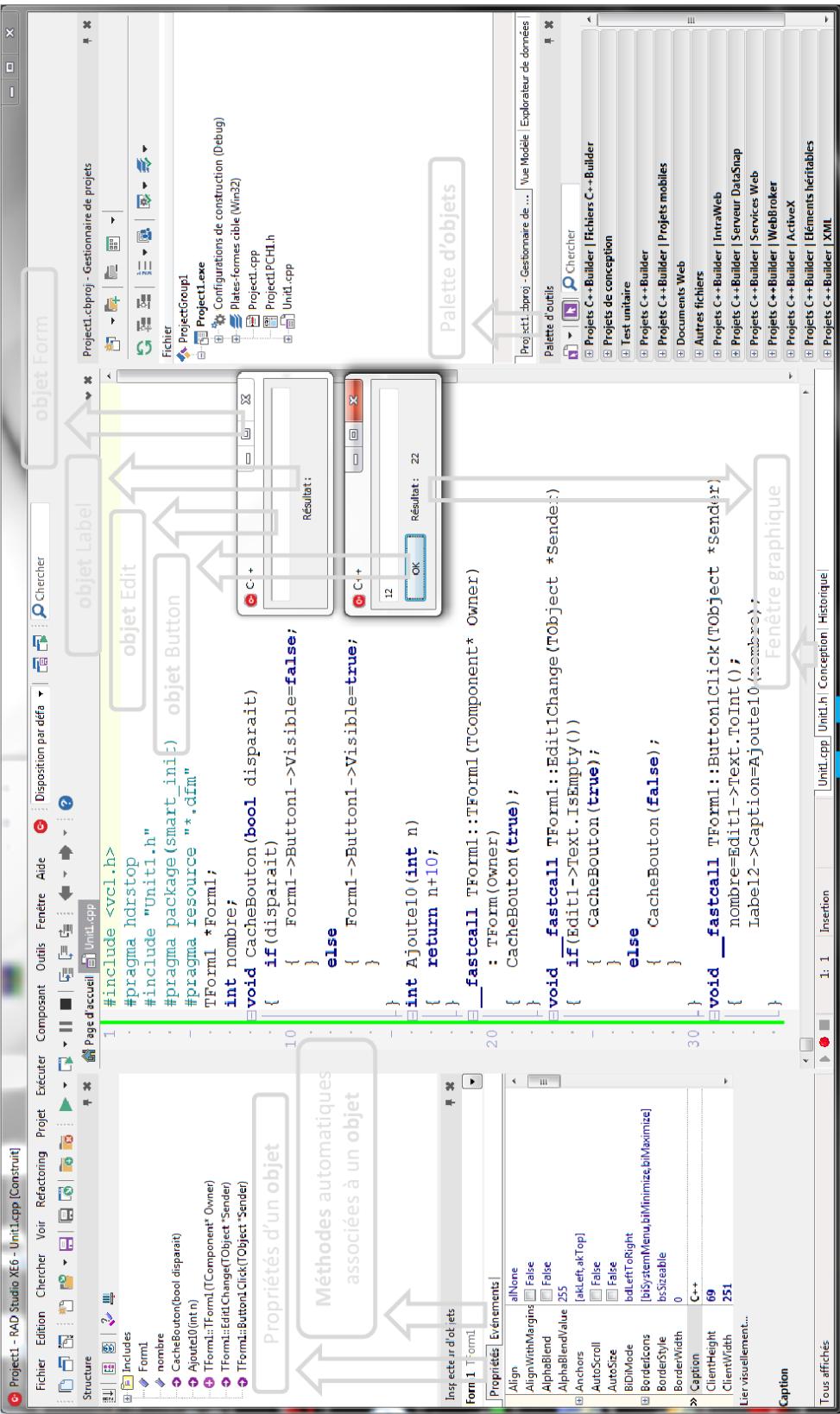


Version 5 canaux			Version 6 canaux		
canal	valeur	effet	canal	valeur	effet
1	0-255	Pan sens horaire	1	0-255	Pan sens anti-horaire
2	0-255	Tilt	2	0-255	Tilt
3	0-7 8-134 135-239 ≥ 240	Pas de lumière Obturateur 100-0% lumière Stroboscope (lent→rapide) Lumière maximum	3	0-7 8-134 135-239 ≥ 240	Pas de lumière Obturateur 100-0% lumière Stroboscope (lent→rapide) Lumière maximum
4	≤ 7 8-21 22-35 36-49 50-63 64-77 78-91 92-105 106-119 120-133 134-147 148-161 162-175 176-189 190-203 204-217 218-231 232-255	Pas de lumière Violet Rouge Vert Bleu foncé Blanc Bleu clair Rose Jaune Vert clair Bleu eau Bleu Bleu blanc Bleu blanc clair Rose claire Rose rouge Orange Arc-en-ciel	5	≤ 7 8-21 22-35 36-49 50-63 64-77 78-91 92-105 106-119 120-133 134-147 148-161 162-175 176-189 190-203 204-217 218-231 232-255	Pas de lumière Violet Rouge Vert Bleu foncé Blanc Bleu clair Rose Jaune Vert clair Bleu eau Bleu Bleu blanc Bleu blanc clair Rose claire Rose rouge Orange Arc-en-ciel
5	0-127 128-255	Motif (gobo) Variation des motifs (lent→rapide)	6	0-127 128-255	Motif (gobo) Variation des motifs (lent→rapide)

Le RAD C++ Builder Xe6



Le RAD C++ Builder Xe6



Une classe sert de plan pour créer des objets, elle est composée d'attributs et de méthodes

Déclaration de la classe

Définition des méthodes et des événements

Annexe 3 : RAD C++ Builder 3

Le RAD C++ Builder Xe6

Définition des méthodes et des événements de la classe TForm1

```

Page d'accueil [Unit1.cpp] Unit1.h
1 #include <vccl.h>
2 #pragma hdrstop
3 #include "Unit1.h"
4 #pragma package (smart_init)
5 #pragma resource "* .dfm"
6 TForm *Form1;
7 int nombre;
8 void CacheBouton(bool disparait)
9 {
10     if(disparait)
11     {
12         Form1->Button1->Visible=false;
13     }
14     else
15     {
16         Form1->Button1->Visible=true;
17     }
18 }
19 __fastcall TForm1::TForm1(TComponent* Owner)
20 {
21     Ajoute10(0);
22     CacheBouton(true);
23 }
24 void __fastcall TForm1::Edit1Change(TObject *Sender)
25 {
26     if(Edit1->Text.IsEmpty())
27     {
28         CacheBouton(true);
29     }
30     else
31     {
32         CacheBouton(false);
33     }
34 }
35 void __fastcall TForm1::Button1Click(TObject *Sender)
36 {
37     nombre=Edit1->Text.ToInt();
38     Label1->Caption=Ajoute10(nombre);
39 }

```

Déclaration de la classe TForm1

```

Page d'accueil [Unit1.h]
1 #ifndef UNIT1H
2 #define UNIT1H
3 #include <System.Classes.hpp>
4 #include <VCL.Controls.hpp>
5 #include <VCL.Statics.hpp>
6 #include <VCL.Forms.hpp>
7 class TForm : public TForm
8 {
9     __published: // composants gérés par l'IDE
10     TButton *Button1;
11     TEdit *Edit1;
12     TLabel *Label1;
13     TLabel *Label2;
14     void __fastcall Edit1Change(TObject *Sender);
15     void __fastcall Button1Click(TObject *Sender);
16     private: // Déclarations utilisées
17     public: // Déclarations utilisées
18     __fastcall TForm(TComponent* Owner);
19 };
20 extern PACKAGE TForm1 *Form1;
21 #endif

```

Bibliothèques de classes et fonctions

Attributs

Constructeur : méthode particulière appelée automatiquement à la création d'un objet

Méthodes

Une **variable** est une zone de données : sa taille est donnée par son **type**
 Une **fonction** est un sous-programme, suite d'instructions, de calculs
 Une **méthode** est une **fonction** composant une **classe**
 Un **attribut** est une **variable** ou un **objet** composant une **classe**
 Un événement est une **méthode** appelée automatiquement
 Un **objet** appelle une **méthode** déclarée dans sa **classe**

Une classe sert de plan pour créer des **objets**,
 elle est composée d'attributs et de méthodes

Annexe 3 : RAD C++ Builder 4

Le RAD C++ Builder Xe6

Définition des méthodes et des événements de la classe TForm1

```

1 #include <vcl.h>
2 #pragma hdrstop
3 #include "Unit1.h"
4 #pragma package(smart init)
5 #pragma resource "*.*"
6
7 TForm1 *Form1;
8
9 void CacheBouton(bool disparait)
10 {
11     if(disparait)
12     {
13         Form1->Button1->Visible=false;
14     }
15     else
16     {
17         Form1->Button1->Visible=true;
18     }
19 }
20
21 int Ajoute10(int n)
22 {
23     return n+10;
24 }
25
26 fastcall TForm1::TForm1(TComponent* Owner)
27 : TForm(Owner)
28 {
29     CacheBouton(true);
30 }
31
32 void __fastcall TForm1::Edit1Change(TObject *Sender)
33 {
34     if(Edit1->Text.IsEmpty())
35     {
36         CacheBouton(true);
37     }
38     else
39     {
40         CacheBouton(false);
41     }
42 }
43
44 void __fastcall TForm1::Button1Click(TObject *Sender)
45 {
46     nombre=Edit1->Text.ToInt();
47     Label2->Caption=Ajoute10(nombre);
48 }
49
50 
```

Classe TForm1

Déclaration de la variable nombre de type int (entier)

Définition de la fonction CacheBouton

Définition de la fonction Ajoute10

Appel de la fonction CacheBouton

Début de la fonction

Fin de la fonction

Création de l'objet Form1

Résultat:

Résultat: 22

OK

Définition du constructeur de TForm1

Définition de l'événement Edit1Change

Définition de l'événement Button1Click

Appel de la fonction Ajoute10

Une variable est une zone de données : sa taille est donnée par son type
 Une fonction est un sous-programme, suite d'instructions, de calculs
 Une méthode est une fonction composant une classe
 Un attribut est une variable ou un objet composant une classe
 Un événement est une méthode appelée automatiquement
 Un objet appelle une méthode déclarée dans sa classe

Une classe sert de plan pour créer des objets,
 elle est composée d'attributs et de méthodes

Annexe 3 : RAD C++ Builder 5

Le RAD C++ Builder Xe6

Définition des méthodes et des événements de la classe TForm1

```

Page d'accueil \Tform1.cpp
1 #include <vccl.h>
- #pragma hdrstop
- #include "Unit1.h"
- #pragma package(smart_init)
- #pragma resource "*.\dfm"
- TForm1 *Form1;
int nombre;
void CacheButton(bool disparait)
{
    if (disparait)
        Form1->Button1->Visible=false;
    else
        Form1->Button1->Visible=true;
}
Aucune variable en résultat;
Variable int en résultat (entier)
Variable int
en argument
Retourne le résultat
Structure alternative
si - sinon :
if(condition)
{ traitement;
}
else
{ traitement;
}

```

* * *

```

1 #include <vccl.h>
- #pragma hdrstop
- #include "Unit1.h"
- #pragma package(smart_init)
- #pragma resource "*.\dfm"
- TForm1 *Form1;
int nombre;
void Ajoute10(int n)
{
    return n+10;
}
int Ajoute10(int n)
{
    return n+10;
}
fastcall TForm1::TComponent* Owner)
{
    CacheButton(true);
}
void __fastcall TForm1::Edit1Change(Tobject *Sender)
{
    if(Edit1->Text.IsEmpty())
        CacheButton(true);
    else
        CacheButton(false);
}
void __fastcall TForm1::Button1Click(Tobject *Sender)
{
    nombre=Edit1->Text.ToInt();
    Label2->Caption=Ajoute10(nombre);
}

```

Appel de la fonction CacheButton avec en entrée la valeur faux

Appel de la fonction Ajoute10 avec en entrée la variable nombre, le résultat est stocké dans Label2->Caption

Une fonction est caractérisée par les () à l'intérieur desquelles on place les variables d'entrée nécessaires au traitement à effectuer.
Lors de sa définition, le nom de la fonction est précédée par le type de la variable résultat

Une variable est une zone de données : sa taille est donnée par son type
Une fonction est un sous-programme, suite d'instructions, de calculs
Une méthode est une fonction composant une classe
Un attribut est une variable ou un objet composant une classe
Un événement est une méthode appellée automatiquement
Un objet appelle une méthode déclarée dans sa classe

Une classe sert de plan pour créer des objets, elle est composée d'attributs et de méthodes

Annexe 3 : RAD C++ Builder 6

Le RAD C++ Builder Xe6

Définition des méthodes et des événements de la classe TForm1

```

Page d'accueil Unit.cpp
1: #include <vcl.h>
2: #pragma hdrstop
3: #include "Unit1.h"
4: #pragma package(smart_init)
5: #pragma resource "*.dfm"
6: TForm1 *Form1;
7: int nombre;
8:
9: void CacheBouton(bool disparait)
10: {
11:     if(disparait)
12:         Form1->Button1->Visible=false;
13:     else
14:         Form1->Button1->Visible=true;
15: }
16:
17: int Ajoute10(int n)
18: {
19:     return n+10;
20: }
21:
22: __fastcall TForm1::TForm1(TComponent* Owner)
23: {
24:     TForm(Owner)
25:     CacheBouton(true);
26: }
27:
28: void __fastcall TForm1::Edit1Change(TObject *Sender)
29: {
30:     if(Edit1->Text.IsEmpty())
31:     {
32:         CacheBouton(true);
33:     }
34:     else
35:     {
36:         CacheBouton(false);
37:     }
38: }
39:
40: void __fastcall TForm1::Button1Click(TObject *Sender)
41: {
42:     nombre=Edit1->Text.ToInt();
43:     Label2->Caption=Ajoute10(nombre);
44: }

```

Un . ou une -> permet à un objet d'accéder à une de ses propriétés (attributs) ou de lancer une de ses méthodes : repérée par les ()

Accès à l'attribut Visible de l'objet Button1, attribut de l'objet Form1

Méthodes auto-générées (_fastcall) qui correspondent à des événements

Définition de la méthode Edit1Change de la classe TForm1

Appel de la méthode IsEmpty() de l'attribut Text de l'objet Edit1

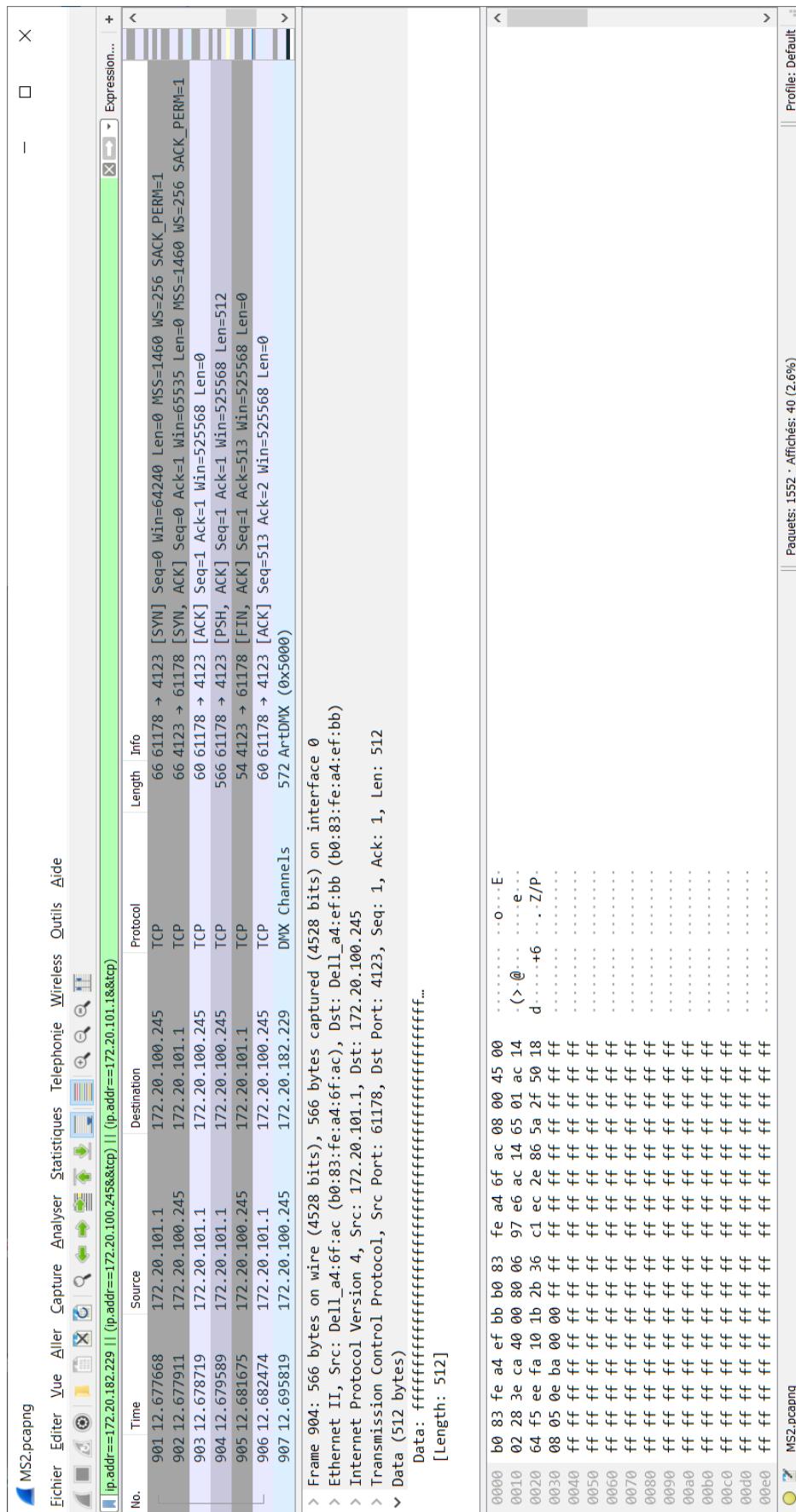
Appel de la méthode ToInt() de l'attribut Text de l'objet Edit1, stockage du résultat dans nombre

Accès à l'attribut Caption de l'objet Label2

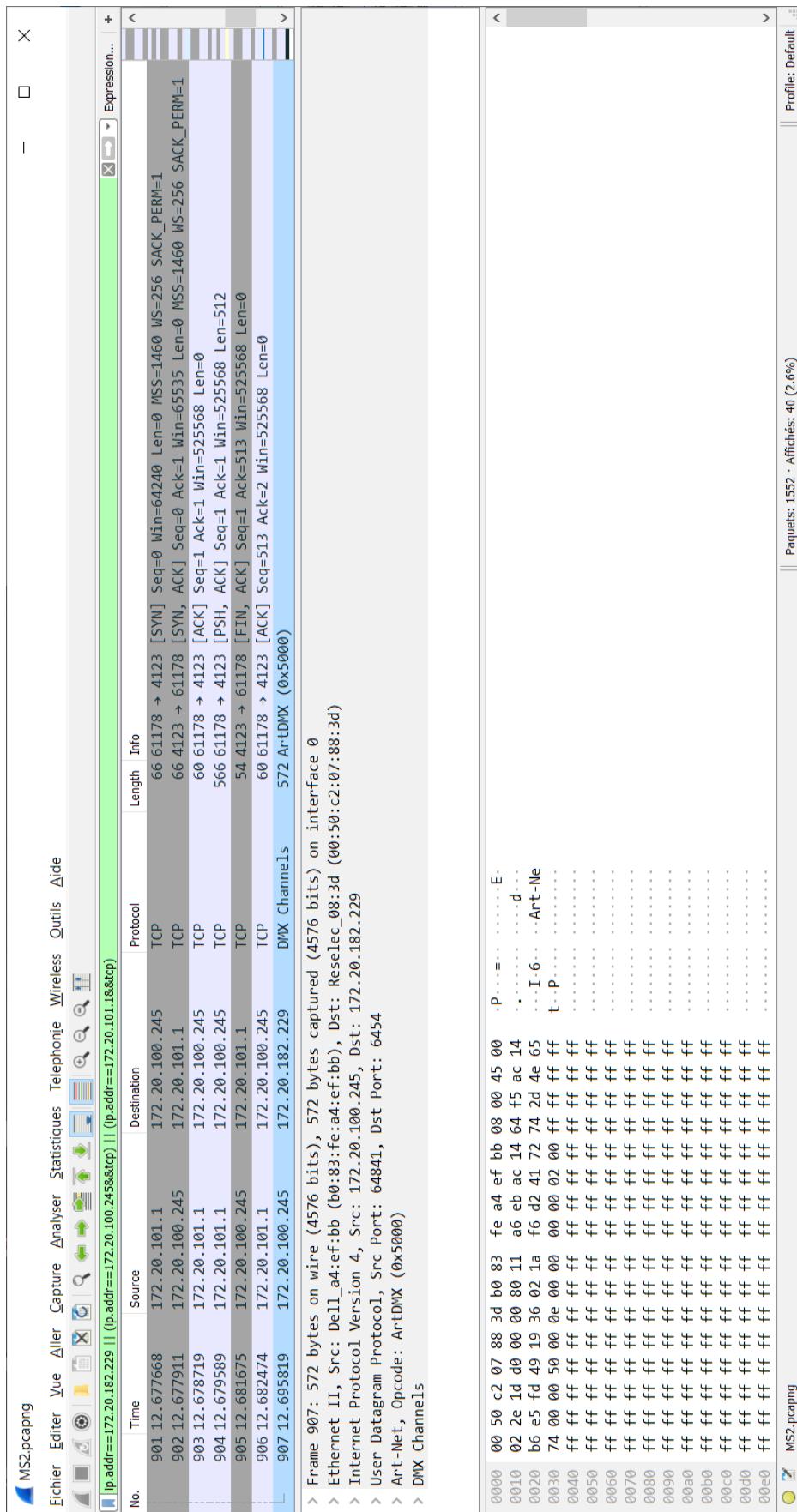
Une fonction est caractérisée par les () à l'intérieur desquelles on place les variables d'entrée nécessaires au traitement à effectuer.

Lors de sa définition, le nom de la fonction est précédée par le type de la variable résultat

Annexe 4 : Relevés Wireshark 1



Annexe 4 : Relevés Wireshark 2



Annexe 5 : Passerelle TCP / UDP

The image shows two terminal windows side-by-side. Both windows have a title bar 'C:\serveurTCPClientEnttec.exe' and a black background with white text.

Left Terminal Window:

```
initialisation de la DDL Winsock2
Numero de la socket : 324
Creation de la socket d'ecoute : OK.
172.20.182.229 6454 0 1000 test.txt
DMX ENTTEC
Mode Demonstration ou Serveur TCP:4123 (D ou S) :
```

Right Terminal Window:

```
initialisation de la DDL Winsock2
Numero de la socket : 304
Creation de la socket d'ecoute : OK.
172.20.182.229 6454 0 1000 test.txt
DMX ENTTEC
Mode Demonstration ou Serveur TCP:4123 (D ou S) : S
Association a un port : OK.
Mise en ecoute : OK.
```


The image shows two Notepad windows side-by-side.

Left Notepad Window: titled 'test.txt - Bloc-notes', contains binary data in hex format.

```
41 72 74 2D 4E 65 74 00 00 50 00 0E 00 00 00 00 02 00
1: B4 BA 4D B4 06 E7 8C 7F 9F AC 20 0D 63 04 70 75 FB A9 32 49
21: B8 A5 66 C9 A7 16 8D 49 F0 41 2F E0 74 DA 8C 52 A7 92 A1 40
41: AB E2 5F 8E AE 17 27 15 85 9E 89 8B 89 92 3E AF B2 84 63 88
61: F2 89 F8 00 D5 8A 99 ED 56 C8 7B 20 04 AE AA E2 56 DB 81 CB
81: 4B 70 6A B4 45 97 D7 DF 28 54 81 65 AE 90 E0 E1 0E 63 EB 5B
101: CD A1 12 77 E4 A9 78 DF 13 6A 75 EE 85 B6 4C 9D A5 CB 29 B3
121: 40 1F 5F 29 DB 6C DF DB 57 FC FB 76 C4 36 5D 9E 82 6A 43 5E
141: 9E D9 FD D5 6B 08 A8 1F 60 47 2C 82 32 7C 74 DA 32 35 ED 46
161: E9 EE 42 14 F1 9F 55 EB 16 89 81 37 AE 42 0F DA 36 FF F6 11
181: 2E 22 D8 A4 36 04 39 51 6A 04 02 7B FA D0 35 0F 0E F3 F2 B7
201: D9 A0 AB 43 FC BD A4 54 1D 33 AE 4A C8 75 24 70 85 4F 26 67
221: 3B F1 15 D0 C5 3B 4F 3E D2 4B 2C 5A 02 45 39 58 3E 62 B4 9B
241: 58 3B 48 A4 E4 58 09 40 56 F4 B3 F2 5D 13 1F 9F 7F C7 07 78
261: F6 BE FB 2D C9 10 A3 50 2E 48 36 07 20 B0 3B F6 3D E2 80 6B
281: E2 8C 58 CC 89 83 18 3E 62 0B D4 97 30 65 56 86 67 9A 60 01
301: 84 87 24 F0 AC 2A 00 18 88 28 7E 49 60 AE 8C CB 76 58 F8 07
321: A5 A0 2F 72 3B 58 37 BF 13 7B EA 45 F7 38 74 AE 34 40 0F E2
341: 7F 4D FE 32 08 F3 C8 C9 DC AB B7 50 81 1D 90 E1 CC AE 94 2F
361: 05 4A B7 F3 3F 73 1F 99 F1 8A E0 2C D9 5D F7 7F 12 F2 84 9D
381: 74 93 4C 73 2C 1B 6F BC 9D 7F 66 2D 76 9F 40 E3 14 4F 1D 09
401: 1A A1 EB CE 4E 76 60 86 B7 44 38 1F F5 29 B1 C7 E0 31 43 DB
421: 65 E4 A4 13 9D 14 FD EC 2E E5 66 57 E7 1D A7 A0 97 AE 32 A0
441: 2E 81 5C 27 17 85 DC A4 D5 F8 86 16 DC F5 48 38 B2 3F 65 E8
461: 46 4C F1 D9 91 95 8E 7A 6B 1D 65 26 AF 40 67 8C 93 BE C6 61
481: 42 07 AE 45 BB C9 48 F4 EE 88 F2 AE 14 9D B8 E8 4E A2 16 34
501: 82 DC F1 6E 73 1A D3 28 20 ED 66 58
```

Right Notepad Window: titled 'config.txt - Bloc-notes', contains configuration parameters.

```
IP 172.20.180.229
Port 6454
Univers 0
tempo(ms) 1000
fichierTest test.txt
```


The image shows a terminal window titled 'C:\serveurTCPClientEnttec.exe' with a black background and white text.

```
Numero sock : 324
Nb octets emis : 530
Last error : 0
41 72 74 2D 4E 65 74 00 00 50 00 0E 00 00 00 00 02 00
1: FE C9 F4 FB 98 0B 14 B8 87 A0 8A 0F 8E 09 7E 80 67 AD 10 06
21: CF 51 FE B4 D5 DB 9A 8E 65 48 EB AA 10 EA 23 87 1D E7 41 AE
41: D6 EF 07 87 0C 3D 1E 7E EC D1 88 56 06 EB 47 CD 9B CC 4C 71
61: 27 A8 A2 C2 2F DB 89 4D BE BB 78 79 A0 23 03 99 89 41 7E
81: 4C 15 F6 78 F4 FD 88 56 02 DA 89 34 86 1E 07 F8 F6 50 F5 B8
101: 3E 87 DA 4F 09 2A B8 3A 4F E6 3F 88 1F F0 33 22 AE 7D 1B 88
121: A1 7E 2B 8E 97 A1 74 85 FB 84 4A CD 08 3C F6 6A 1D 06 9D E5
141: 43 4C 70 65 FC DC 16 0E 4F 64 56 9A 11 31 09 37 75 28 3F A0
161: D6 05 B1 7E C4 D3 67 01 4E AC AA BA 8F B3 2C BC 7C 4F 76 F4
181: ED A9 93 C5 E9 68 FB E7 36 AC C2 81 21 49 01 85 8A 45 85 4D
201: 34 94 B7 74 48 12 CB F6 37 C2 2D 8A A0 46 29 40 C2 9E D9 57
221: F2 25 54 61 71 C9 14 9A 71 8A A8 85 66 36 9C C8 8F 63 A6 4A
241: C4 AD 12 8A 81 31 6F 43 33 4A 78 BE C3 8B 4C 77 61 FD C1 73
261: 92 9A 29 E1 7C 09 31 70 71 9C 13 F0 BB 88 FD AC A7 63 BF 06
281: D5 E3 B9 54 BE D5 04 FB 86 5C E0 CA 04 6C 5E 99 10 87 4E 23
301: 0C B6 6B 20 B5 CA C6 A6 2C D4 70 23 F2 6B 52 FE FD D3 26
321: 7B 60 49 57 E0 FA A0 E6 B7 26 CC 99 76 A9 07 13 4A EC 45 32
341: 25 7C DA AE 0A BF 6A EF 98 F9 11 2E D6 7F D4 D0 3E 3A 48 FF
361: 0C 5F 81 8B C8 6C 3D 00 11 66 52 07 AC 52 55 02 CE F2 8C E3
381: A7 C2 16 4E 2F 32 59 EF 35 25 AC 1D A5 A9 46 B0 19 F9 67 1F
401: A3 B0 C5 DE D9 50 39 F7 22 F3 AA E7 41 50 3A BB 1F F1 B0 6B
421: DD B2 26 77 1A 7E F4 C0 81 46 DB A9 97 42 76 6E C4 6C DE C4
441: 9F 39 9D B2 7E 9A D5 AF 3E 33 B1 FD 4F DB 0D 45 04 54 61 73
461: 1A 4C F3 D5 86 94 3D 6F 84 9F 9C A1 DC 3E 3B FC 77 9A 40 61
481: 23 72 35 5C A3 96 B8 BF FB A4 67 82 FA 03 06 DB 08 1D F3 99
501: 31 E1 CC C7 72 78 5E 7E 44 46 D1 09
```



Module Système 3

Caméras IP de surveillance



Table des matières

Objectifs, matériels et ressources.....	2
TD 1 - Analyse UML.....	3
Défi 1 - Prise en main du matériel et analyse des trames.....	4
TD 2 - Analyse des trames échangées entre le navigateur Web et le serveur Web de la caméra IP avec Wireshark.....	5
Défi 2 - Prise d'une photo avec la caméra IP Heden.....	7
TD 3 - La classe CameraHedenNoire.....	9
Défi 3 - Codage de la classe CameraHedenNoire.....	10
TD 4 - La classe ConfigurationCamera.....	12
Défi 4 - Codage de la classe ConfigurationCamera.....	14
TD 5 - Les classes CameraHedenNoire et ConfigurationCamera.....	15
Défi 5 - Les classes CameraHedenNoire et ConfigurationCamera.....	16
Défi 6 - Déplacement de la caméra IP.....	17

Objectifs, matériels et ressources

Objectifs

domaine	compétences	où	acquis
C++	<ul style="list-style-type: none"> Écriture d'une méthode ou d'une fonction à partir d'un diagramme de séquence : <ul style="list-style-type: none"> Respect de la chronologie des événements 	Défi 2 TD 4	
	<ul style="list-style-type: none"> Création d'un objet 	Défi 2 TD 4	
	<ul style="list-style-type: none"> Appel de la méthode d'un objet 	Défi 2 TD 4	
	<ul style="list-style-type: none"> Interpréter une boucle (loop) 	Défi 2	
	<ul style="list-style-type: none"> Écriture d'une classe à partir de son diagramme de classe : 	TD 3,4,5	
	<ul style="list-style-type: none"> Écriture des attributs et des méthodes selon leur visibilité 	TD 3,4,5	
	<ul style="list-style-type: none"> Gestion d'une composition 	TD 5	
	<ul style="list-style-type: none"> Consultation de la documentation d'une fonction, d'une méthode ou d'une classe 	TD 3	
	<ul style="list-style-type: none"> Lecture et écriture dans un fichier binaire 	Défi 2	
	<ul style="list-style-type: none"> Lecture d'un fichier texte 	TD 4	
UML	<ul style="list-style-type: none"> Savoir analyser un cahier des charge et <ul style="list-style-type: none"> dessiner le diagramme de cas d'utilisation dessiner le diagramme de déploiement dessiner le diagramme de séquence système Savoir compléter un diagramme de séquence à partir d'un cahier des charges 	TD 1 TD 1 TD 1 TD 4	

Matériels

- Camera IP Heden



Ressources

La documentation de la caméra IP Heden.

Les sites de référence des librairies standards du C/C++ : www.cplusplus.com ou cppreference.com.

TD 1 - Analyse UML

Cahier des charges

Une société de sécurité souhaite créer une nouvelle application de surveillance des locaux qui affiche le flux vidéos de nouvelles caméras IP Heden qu'ils viennent d'acquérir. Cette application sera installée au poste de contrôle des agents de sécurité.

Cette application détectera automatiquement tout mouvement inhabituel et déclenchera une alarme lumineuse au poste de sécurité. Dans le même temps, un email et/ou un MMS sera envoyé aux agents de sécurité avec la capture d'image. S'ils ne réagissent pas dans les 30 secondes (en confirmant ou non l'intrusion), une alarme sonore devra retentir.

Analyse du cahier des charges

?

Dessiner le diagramme d'activité :

?

Dessiner le diagramme de déploiement :

?

Dessiner le diagramme de séquence d'un point de vue système :

Défi 1 - Prise en main du matériel et analyse des trames

Objectifs

Le premier objectif de cette partie est de découvrir la caméra IP Heden VisionCam : son branchement, éventuellement son paramétrage et son pilotage avec un navigateur web.

Le second objectif est d'analyser les trames échangées entre la caméra et le navigateur web lors de la prise de photo. Ceci nous permettra d'envoyer la même trame avec notre future application.

Raccordement et vérification du fonctionnement de la caméra IP

- Raccorder la caméra IP au secteur ainsi qu'au réseau local avec les câbles qui conviennent.
- Vérifier si les informations indiquées sur la caméra sont correctes en tapant son adresse IP et son numéro dans un navigateur web.
- Si la caméra n'est pas visible sur le réseau, utiliser les logiciels VisionCam ou IPCamTools pour la reparamétrer.
- Explorer les possibilités offertes par cette caméra IP : flux vidéo, déplacement, prise de photo, etc.

Serveur et protocole

■ ? S'il est possible de se connecter à la caméra IP en utilisant un navigateur Web, en déduire quel type de serveur est disponible sur la caméra IP.

■ ? En déduire le protocole de communication qui est utilisé entre navigateur Web et la caméra IP. Donner la signification de ce sigle et son contexte d'utilisation.

Le protocole :

La signification :

Le contexte d'utilisation :

■ ? En utilisant Internet, trouver à quelle couche du modèle OSI ce protocole appartient.

Analyse de trames

Dans la suite de ce module, vous créerez une application qui prendra des photos avec la caméra IP. Pour y arriver, il sera très utile d'analyser les trames échangées entre le navigateur web et le serveur web de la caméra.

- Utiliser Wireshark pour analyser les trames échangées entre le navigateur web et la caméra IP lors d'une prise de photo (avant de lancer Wireshark, arrêter le flux vidéo ; démarrer Wireshark puis rafraîchir la page contenant la prise de photo).
- Pour filtrer les HTTP échangées entre le navigateur web et la caméra IP, ajouter le filtre : `http && ip.addr == adresseIPDeVotreCaméraHeden`.

Le TD suivant permet d'analyser plus en détail les trames échangées entre le navigateur web et la caméra IP.

TD 2 - Analyse des trames échangées entre le navigateur Web et le serveur Web de la caméra IP avec Wireshark

L'objectif de ce TD est d'analyser les trames échangées entre le navigateur web et le serveur web de la caméra Heden.

Les 2 trames HTTP échangées

En utilisant le relevé ci-contre, en déduire l'adresse IP du navigateur web et celle du serveur web de la caméra IP.

@IP du poste avec navigateur :

@IP du serveur web de la caméra IP :

Donner le numéro de port du serveur web de la caméra IP.

Préciser le nom du protocole utilisé par HTTP : TCP ou UDP ?

La requête HTTP envoyée par le navigateur web

La première des 2 trames est la requête HTTP émise par le navigateur web. Elle commence par une commande composée d'une méthode, d'une ressource et d'une version.

Donner la méthode, la ressource et la version utilisée dans cette requête HTTP

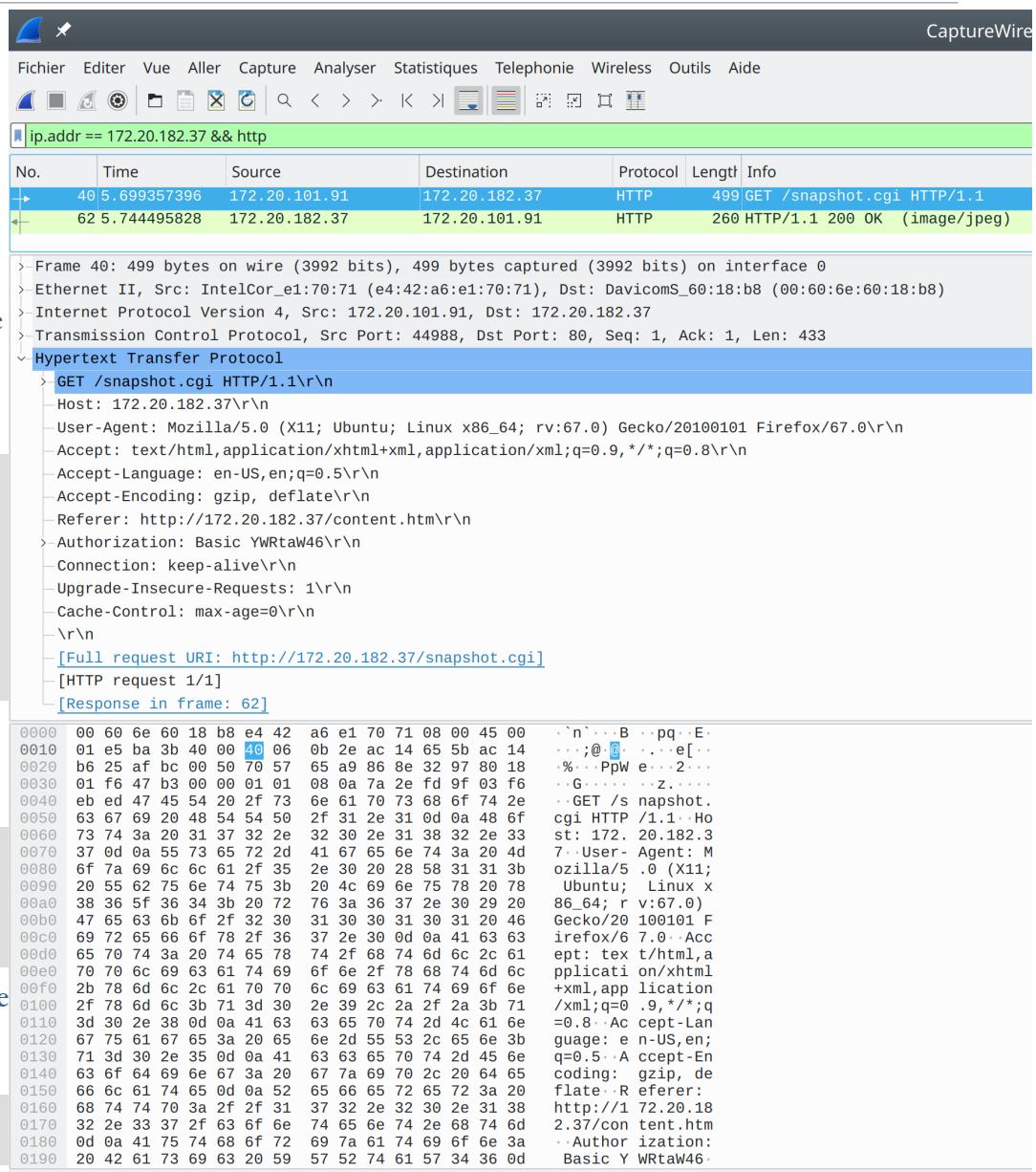
Méthode :

Ressource :

Version :

Donner l'explication de cette commande.

Après la commande, on trouve des options toutes séparées par ‘\r\n’. Donner la signification des options suivantes :



Host : 172.20.182.37

Authorization: Basic YWRtaW46

Connection : keep-alive

A RETENIR: Votre future application devra impérativement envoyer la commande et les trois options précédentes au serveur web de la caméra IP. Ce n'est qu'ainsi que vous recevrez en réponse la photo demandée.

La réponse HTTP envoyée par le serveur web

Voici ci-contre le détail de la réponse du serveur Web.

La réponse du serveur web commence par l'état du traitement de la requête HTTP

Dans cet exemple, l'état est :

HTTP/1.1 200 OK

Donner la signification de cet état.

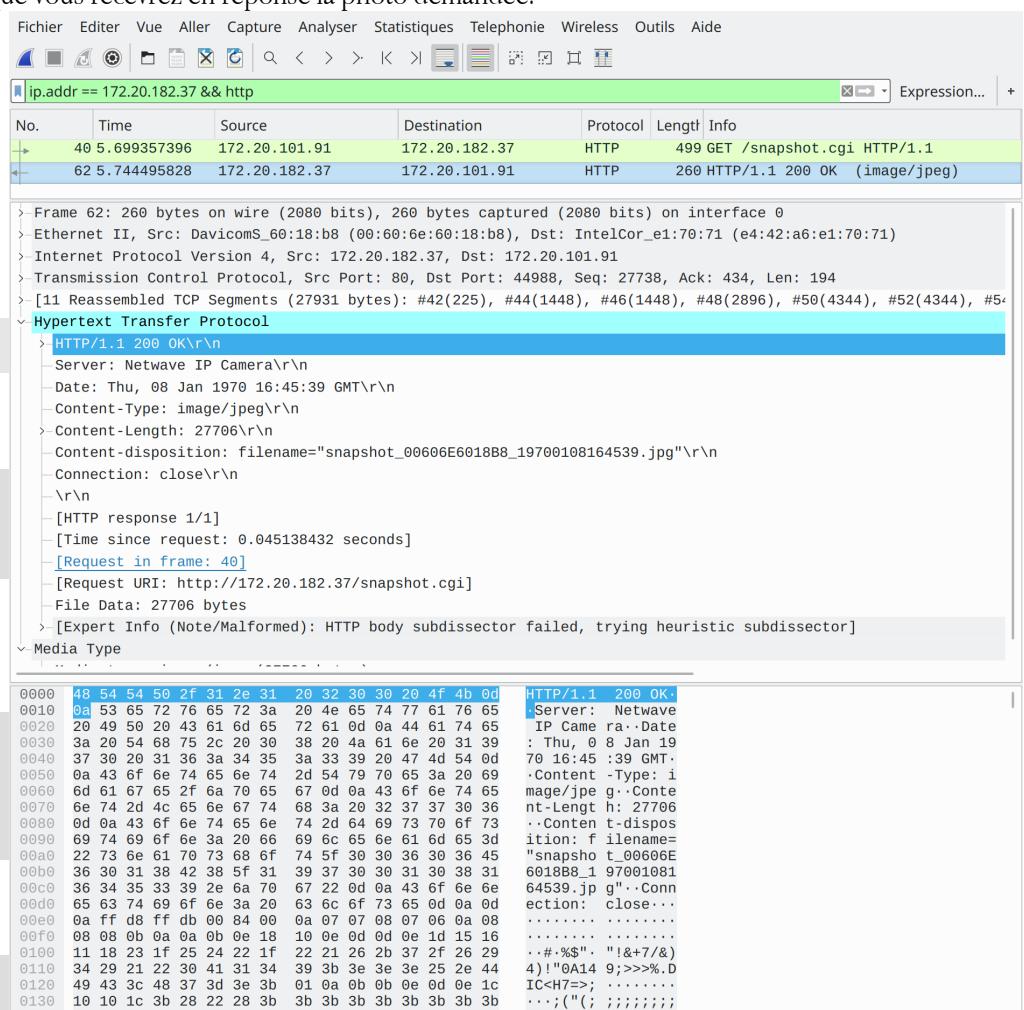
Suivent ensuite des options

Donner la signification de l'option Content-type.

Donner la signification de l'option Content-Length.

Donner la signification de l'option Content-disposition.

Après l'entête HTTP, indiquer ce que va contenir cette réponse du serveur.

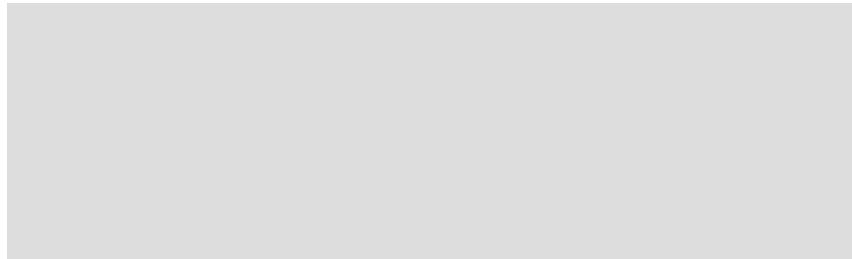


Défi 2 - Prise d'une photo avec la caméra IP Heden

Créer un projet en mode console avec Embarcadero. Enregistrer votre projet dans un répertoire nommé MS3-Defi2.

Envoi de la trame HTTP au serveur

Traduire le diagramme de séquence suivant en C++ :



Écrire le code précédent dans votre projet. Compiler et exécuter le projet.

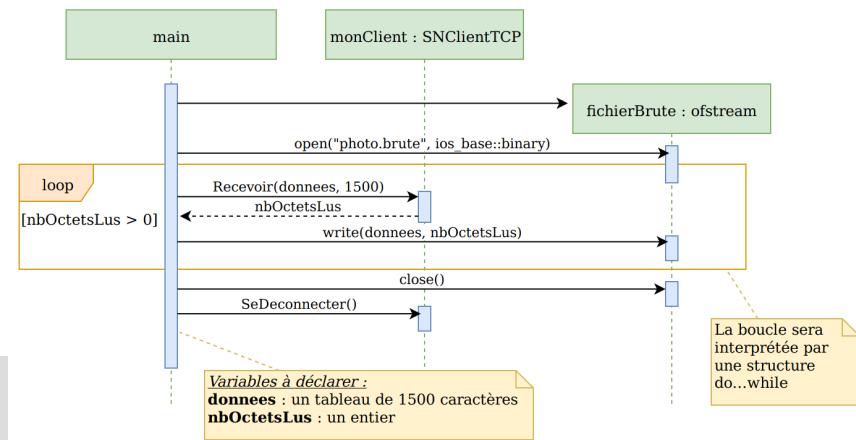
Analyser les trames avec Wireshark et vérifier que la requête a été correctement envoyée.

Réception de la réponse du serveur

La réception de la réponse du serveur est illustrée avec le diagramme de séquence ci-dessous :

Tous les octets de réponse provenant du serveur seront enregistrés dans un fichier nommé « photo.brute ».

Interpréter en C++ le diagramme de séquence :



Ajouter le code précédent dans la main.

Tester votre programme. Vérifier la création du fichier *photo.brute*.

Cette image est-elle visualisable ? Expliquer en ouvrant ce fichier avec éditeur hexadécimal (comme HexEdit).

Extraction de l'image

Il est nécessaire de supprimer l'entête HTTP de du fichier *photo.brute* pour que l'image soit au bon format.
L'objectif de cette partie va être de créer un nouveau fichier *photo.jpg* en copiant à l'intérieur tous les octets du fichier *photo.brute* sauf l'entête HTTP.

Le pseudo-code suivant explique le principe de l'extraction de l'image :

```
Créer un objet fichierLecture de la classe ifstream
Avec l'objet fichierLecture, ouvrir le fichier « photo.brute »
Créer un objet fichierEcriture de la classe ofstream
Avec l'objet fichierEcriture, ouvrir le fichier « photo.jpg »
Déclarer un booléen nommé copie. Lui attribuer la valeur faux.
Déclarer un caractère nommé octetLu.
FAIRE
    Lire un octet du fichierLecture et stocker sa valeur dans octetLu
    SI octetLu vaut 0xFF
        ALORS modifier copie à vrai
    FIN SI
    SI copie vaut vrai
        ALORS écrire octetLu dans fichierEcriture
    FIN SI
TANT QUE
Fermer fichierEcriture
Fermer fichierLecture
```

Traduire le code précédent en C++

- ⌨ Ajouter le code précédent dans la méthode PrendrePhoto().
- ⌨ Vérifier que le fichier photo.jpg est correctement créé et que son format est correct.

TD 3 - La classe CameraHedenNoire

Objectif

L'objectif de ce défi est de créer la classe CameraHedenNoire. Cette classe permettra de prendre des photos avec n'importe quelle caméra Heden noire connectée au réseau local. Cette classe sera écrite en réutilisant le code écrit dans le programme principal précédent.

Présentation de la classe CameraHedenNoire

Analyse du diagramme de classe

? Donner le nombre d'attributs de cette classe et la visibilité de ces attributs.

? Donner le nombre de méthodes que possède cette classe et leur visibilité.

? Donner le nom que porte la méthode CameraHedenNoire().

CameraHedenNoire
- adresseIP: char [20]
- port : unsigned short
- identifiant : char [30]
- motDePasse : char [30]
+ CameraHedenNoire()
+ PrendrePhoto(nomFichier : char[]) : bool
+ Deplacer(commande : int) : void

Écriture du fichier d'entête (CameraHedenNoire.h)

? Écrire en C++ la déclaration de cette classe (CameraHedenNoire.h) :

Écriture du constructeur de la classe CameraHedenNoire et test

Le constructeur de la classe CameraHedenNoire initialisera les attributs de la classe aux valeurs de la caméra que vous avez utilisée dans le défi 1.

? En C++, initialiser les attributs de la classe. Vous utiliserez la fonction [strcpy\(\)](#), pour modifier les chaînes de caractères.

? Que suffit-il de faire pour tester un constructeur ? Écrire le code C++ qui y correspond.

Défi 3 - Codage de la classe CameraHedenNoire

- 💻 Créer un projet en mode console avec Embarcadero. Enregistrer votre nouveau projet dans le répertoire MS3.

La déclaration de la classe

- 💻 Ajouter une nouvelle classe à votre projet. Nommer cette classe CameraHedenNoire. Écrire la déclaration de la classe précédente dans le fichier d'entête (.h).

Le constructeur

- 💻 Ajouter la définition du constructeur de la classe CameraHedenNoire dans le fichier source (.cpp).

 Pour tester le constructeur, créer un objet nommé 'cam' de la classe CameraHedenNoire

- 💻 Ajouter la déclaration de l'objet 'cam' de la classe CameraHedenNoire dans le programme principal.

- 💻 Vérifier que la compilation et l'exécution de votre programme se passe correctement. Corriger les éventuelles erreurs.

La méthode PrendrePhoto()

- 💻 Ajouter le code du programme principal du défi 2 dans la méthode *PrendrePhoto()*.
- 💻 L'image sera enregistrée dans un fichier dont le nom est passé en paramètre de la méthode *PrendrePhoto(char * nomFichier)*.
- 💻 Vérifier que le fichier *photo.jpg* est correctement créé et que son format est correct.

Adapter la requête en fonction de l'adresse IP et du port

- 💻 Modifier la requête créée dans la méthode *PrendrePhoto()* pour qu'elle s'adapte à l'attribut *adresseIP* et *port* de la classe CameraHedenNoire. Utiliser la fonction *sprintf()*.
- 💻 Tester.

Adapter la requête en fonction de l'identifiant et du mot de passe

L'identifiant et le mot de passe de l'utilisateur sont transmis à la caméra IP comme suit :

Étape 1. Concaténation du nom d'utilisateur avec le mot de passe séparés par ' : '. Par exemple, si le nom d'utilisateur est admin et le mot de passe est admin, on obtient « admin:admin ».

Étape 2. Codage de cette chaîne de caractères en base 64. Le site <https://www.base64encode.org/> permet de coder et décoder en base 64. Le résultat de l'exemple précédent est : « YWRtaW46YWRtaW4= »

Étape 1 : Concaténation

 En C++, créer une nouvelle chaîne de 50 caractères maximum nommée *concat*. En utilisant la fonction *sprintf()*, effectuer la concaténation de l'étape 1 dans *concat*.

Étape 2 : Codage base 64

La fonction *void codage64(const char * mot, char * motCode)* permet d'effectuer le codage en base 64 d'une chaîne de caractères. Le premier paramètre est la chaîne de caractères que l'on souhaite coder en base 64. Le second paramètre est la

chaîne de caractères dans laquelle la fonction stockera la chaîne codée.

Voici un exemple de son utilisation :

```
char monTexte[50] = "Le BTS SNIR";
char texteB64[50];
codage64(monTexte, texteB64);
cout << texteB64 << endl;
```

Expliquer le code précédent donné en exemple :

En C++, créer une nouvelle chaîne de 50 caractères maximum nommée *code* qui contiendra le codage en base 64 de *concat*. En utilisant la fonction *codage64(const char * mot, char * motCode)*, coder en base 64 la chaîne de caractères *concat* et mettre le résultat dans *code*.

💻 Ajouter ce code dans la méthode *PrendrePhoto()*. Modifier *requete* en conséquence.

💻 Tester votre code en renseignant un nom d'utilisateur et un mot de passe correct puis erroné. Vérifier les trames envoyées par le serveur avec Wireshark.

💡 Expliquer comment gérer dans *PrendrePhoto()* une réponse du serveur indiquant un erreur.

💡 Proposer une solution en C++.

💻 Appliquer cette solution dans votre projet.

TD 4 - La classe ConfigurationCamera

Objectifs

On souhaite maintenant que la configuration de la caméra utilisée ne soit pas « écrite en dure » dans le code source de notre projet mais dans un fichier de configuration. Ce fichier de configuration (config.txt) aura le format suivant :

```
IP 172.20.182.37
PORT 80
UTILISATEUR admin
MDP -                                // si aucun mot de passe
```

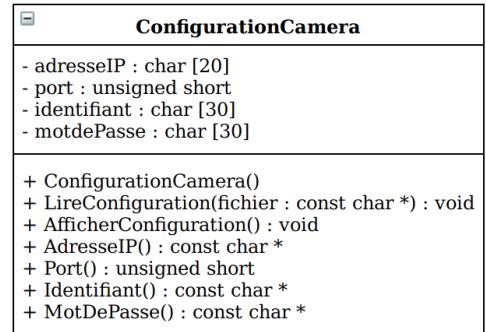
C'est la classe ConfigurationCamera qui sera chargée de lire le fichier de configuration et d'extraire les valeurs dans des variables.

La classe ConfigurationCamera

La classe ConfigurationCamera est représentée dans le diagramme de classe ci-dessous.

La déclaration de la classe

? À partir de ce diagramme de classe, écrire la déclaration de la classe ConfigurationCamera.



Le constructeur

? Rappeler le rôle d'un constructeur :

Le constructeur de cette classe initialise adresseIP avec « 0.0.0.0 », port à 0, identifiant avec « admin » et motDePasse avec « ».

? Écrire en C++ la définition du constructeur. Utiliser la fonction strcpy() pour initialiser les chaînes de caractères.

La méthode AfficherConfiguration()

La méthode AfficherConfiguration() à pour rôle d'afficher avec l'objet cout les 4 attributs de la classe.

? Indiquer le nom du fichier à inclure pour utiliser l'objet cout.

? Écrire en C++ la définition de cette méthode.

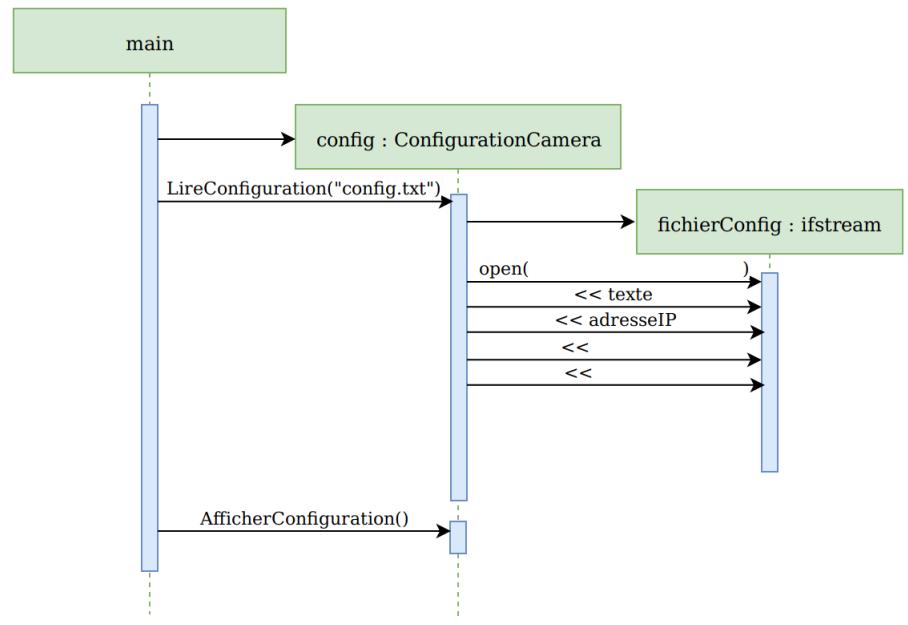
La méthode LireConfiguration()

La méthode LireConfiguration() doit :

1. Ouvrir en lecture seule le fichier de configuration dont le nom est passé en paramètre.
2. Lire chaque ligne du fichier de configuration une par une et affecter la valeur lue à l'attribut de la classe correspondant.
3. Fermer le fichier de configuration.

? Compléter le diagramme de séquence suivant :

? Traduire en C++ la méthode LireConfiguration() de la classe ConfigurationCamera.



? Traduire en C++ la fonction principale main.

Les méthodes 4 méthodes d'accès : AdresseIP(), Port(), Identifiant() et MotDePasse()

? Donner la définition des 4 méthodes d'accès. Expliquer leur intérêt.

Défi 4 - Codage de la classe ConfigurationCamera

Création d'un nouveau projet

Vous allez créer un nouveau projet pour créer la classe ConfigurationCamera et la tester. Une fois les tests effectués, cette nouvelle classe sera importer dans le projet MS3.

- █ Créez un nouveau projet Embarcadero en mode console. Lui donner le nom MS3-Defi4.

Déclaration de la classe ConfigurationCamera

- █ Pour créer la classe ConfigurationCamera, ajouter une nouvelle *Unité* dans votre projet. Renommer le fichier en ConfigurationCamera.cpp
- █ Dans le fichier ConfigurationCamera.h, écrire en C++ la déclaration de la classe faite dans le TD précédent.

Définition du constructeur

- █ En reprenant le TD précédent, ajouter la définition du constructeur dans le fichier ConfigurationCamera.cpp
- █ Dans la fonction principale main(), ajouter la déclaration d'un objet nommé 'conf' de la classe ConfigurationCamera.
- █ Compiler et vérifier que votre programme s'exécute sans erreur.

Définition de la méthode AfficherConfiguration()

- █ En reprenant le TD précédent, ajouter la définition de la méthode AfficherConfiguration(). Faire les inclusions nécessaires.
- █ Dans la fonction principale main(), appeler la méthode AfficherConfiguration()
- █ Compiler et vérifier que l'affichage dans le terminal est correct (les attributs doivent avoir la valeur imposée par le constructeur).

Définition de la méthode LireConfiguration()

- █ En reprenant le TD précédent, ajouter la définition de la méthode LireConfiguration().
- █ Modifier le programme principale comme dans le diagramme de séquence du TD précédent.
- █ Compiler et tester que la lecture du fichier de configuration se fait correctement.

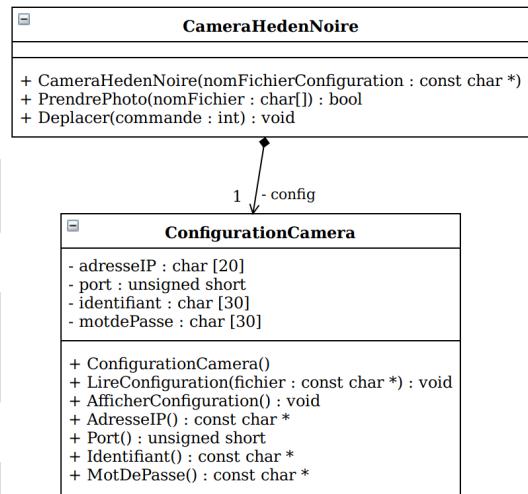
Les 4 méthodes d'accès

- █ En reprenant le TD précédent, ajouter la définition des 4 méthodes d'accès.
- █ Les tester toutes les 4 dans le programme principal en affichant la valeur qu'elles renvoient.
- █ Vérifier que les valeurs affichées sont correctes.

TD 5 - Les classes CameraHedenNoire et ConfigurationCamera

Pour la version finale de ce projet, les 2 classes CameraHedenNoire et ConfigurationCamera seront liées comme le montre clairement le diagramme de classe ci-contre.

? Donner le nom du lien entre ces 2 classes.



? Expliquer ce que ce lien ajoute dans la classe CameraHedenNoire.

? Écrire en C++ la déclaration de la classe CameraHedenNoire.

Le constructeur de la classe CameraHedenNoire doit lire le fichier de configuration grâce à la méthode *LireConfiguration* de l'objet *config*. Le nom du fichier de configuration est passé en argument du constructeur.

? Écrire en C++ le code du constructeur de la classe CameraHedenNoire :

? Dans le programme principal, créer un objet nommé *cam* de la classe CameraHedenNoire. Fournir en argument le nom du fichier de configuration que vous avez créé, par exemple *config.txt*.

```

int main()
{
}
  
```

Défi 5 - Les classes CameraHedenNoire et ConfigurationCamera

- Repartez du projet créé dans le répertoire MS3.
- Copier-coller les fichiers sources de la classe ConfigurationCamera dans le répertoire MS3 puis ajouter ces fichiers dans votre projet.
- Modifier la classe CameraHedenNoire comme indiquée dans le TD précédent.
- Modifier le programme principal pour créer un objet *cam* en précisant le nom du fichier de configuration.

Les appels aux anciens attributs adresseIP, port, identifiant et motDePasse sont obsolètes. Ces attributs étant maintenant dans la classe ConfigurationCamera, il est possible d'y accéder grâce à leurs méthodes d'accès. Par exemple, le code

```
adresseIP
```

sera changé par

```
config.AdresseIP();
```

Il en est de même pour les 3 autres.

- Modifier en conséquence la méthode PrendrePhoto().
- Vérifier que la prise de photo fonctionne toujours correctement.

Défi 6 - Déplacement de la caméra IP

Objectif

L'objectif de cette partie est d'envoyer les bonnes trames au serveur de la caméra IP pour qu'elle se déplace : à droite, à gauche, en haut et en bas !

Analyse des trames

La première étapes consiste à analyser les trames qui sont échangées entre le navigateur web et la caméra IP lorsqu'on souhaite la déplacer.

 En utilisant Wireshark, faire la liste des trames envoyées pour déplacer la caméra IP à droite, à gauche, en haut et en bas.

 À partir de cette analyse, indiquer quelle(s) est(sont) la(les) différence(s) entre les différentes trames.

 Écrire en C++ la définition de la méthode Deplacer(commande : int) : void de la classe CameraHedenNoire.

 Ajouter la définition de la méthode Deplacer() dans votre projet.

 Tester cette méthode en l'appelant dans la fonction principale et vérifier son bon fonctionnement.



Module Système 4

Supervision d'un véhicule sur bus CAN



Table des matières

Objectifs, matériels et ressources.....	2
TD1 – Cahier des charges et analyse UML.....	3
TD2 – Etude de la norme et de la documentation du matériel.....	5
TP – Défi 1 – demande d'une trame moteur ou état des feux en mode console : client TCP.....	7
TP – Défi 2 – la classe CombineC4.....	9
TP – Défi 3 – calcul de la vitesse, du régime et extraction des feux.....	11
TP – Défi 4 – classe CompteurAiguille : création d'une image.....	14
TP – Défi 5 – tracé de l'aiguille.....	16
TP – Défi 6 – interface graphique finale.....	19
Annexe 1 : norme Bus CAN.....	20
Annexe 2 : notice MT-CAN-LIN-BSI1.....	28
Annexe 3: diagramme de classe Exxotest complet.....	30

Objectifs, matériels et ressources

Objectifs

Étudier la messagerie CAN de la maquette Exxotest (Citröen C4) afin de remplacer le compteur à aiguilles par un affichage graphique. L'affichage sera réalisé sur un ordinateur, l'affichage sur tablette graphique est envisageable dans un second temps. Le code devra être construit autour de 2 classes dont le diagramme de classe est fourni.

- Tableau en langage C, notion de trame
- Extraction de données et traitement de bits
- Diagramme de classe, codage d'une classe
- Codage d'un client TCP
- Environnement RAD (Rapid Application Development)
- Interface graphique
- Bus CAN

Matériels



- Module VSCOM CAN-USB
- Maquette MT-CAN-LIN-BSI (Exxotest)



Codes sources

- SNImage.h
- SNImage.obj
- SNClientTCP.h
- SNClientTCP.cpp

Logiciels

- ServeurExxotestVSCOMdec2014.exe
- SimulationExxotestVSCOM.exe
- C++ Builder
- regmodify.vbs et scripVSCOM.jpg
- OscilloCan.exe

Ressources

- boschCAN2SPEC.PDF
- Notice MT-CAN-LIN-BSI.pdf
- Schéma-Général-MT-CAN-LIN-BSI.pdf
- voyants.zip

Annexes

- extrait du site :
[https://fr.wikipedia.org/wiki/Bus_de_données_CAN
#Couche_physique](https://fr.wikipedia.org/wiki/Bus_de_données_CAN#Couche_physique)
- extrait documentation Maquette MT-CAN-LIN-BSI (Exxotest)
- diagramme de classe Exxotest complet

TD1 – Cahier des charges et analyse UML

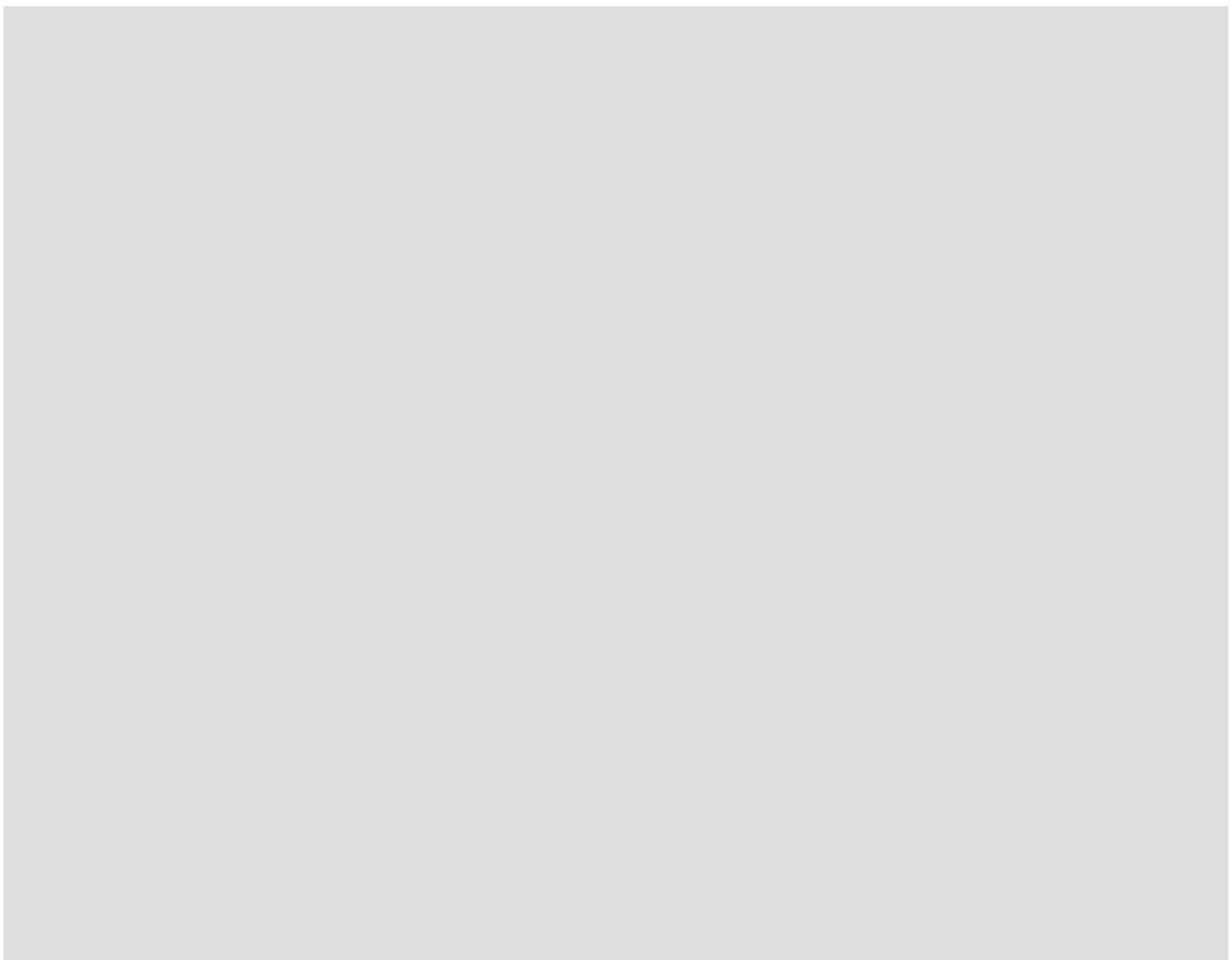
Le cahier des charges

On nous demande de remplacer le compteur analogique à aiguille (combiné) d'une automobile par un affichage digital sur tablette. Les informations envoyées au combiné de la Citroën C4 sont véhiculées sur un bus CAN Low Speed cadencé à 125 kbits/s. Pour récupérer les informations transitant sur le bus CAN nous utiliserons un module VS-COM permettant la conversion CAN-USB. Un logiciel serveur TCP/IP sera chargé d'envoyer les trames CAN aux postes clients en faisant la demande. C'est sur le poste client (PC ou tablette) que le compteur digital sera affiché en temps réel : informations des feux et des clignotants, de la vitesse du véhicule et du régime du moteur.



Les diagrammes de cas d'utilisation, de déploiement et de séquence

Dessiner le diagramme de cas d'utilisation :





Dessiner le diagramme de déploiement :



Dessiner le diagramme de séquence d'un point de vue système :

TD2 – Étude de la norme et de la documentation du matériel

A partir de la documentation fournie en annexe, donner les principales caractéristiques du bus CAN.



Présentation matérielle :



Topologie :



Vitesse :



Longueur du bus :



Encodage :



Niveaux électriques :



Arbitrage et priorité :



Contrôle d'erreurs :



Donner la composition d'une trame CAN et précisant le nombre de bits relatif à chaque champ :



Donner le nombre de bits maximum d'une trame :



Donner le nombre de bits minimum d'une trame :



Donner la durée maximum d'une trame à 125kbits/s :



Donner la durée minimum d'une trame à 125kbits/s :



Combien de liaisons de données composent la maquette Exxotest ?



Parmi ces liaisons : combien y a-t-il de bus CAN ?



Que signifie BSI ?

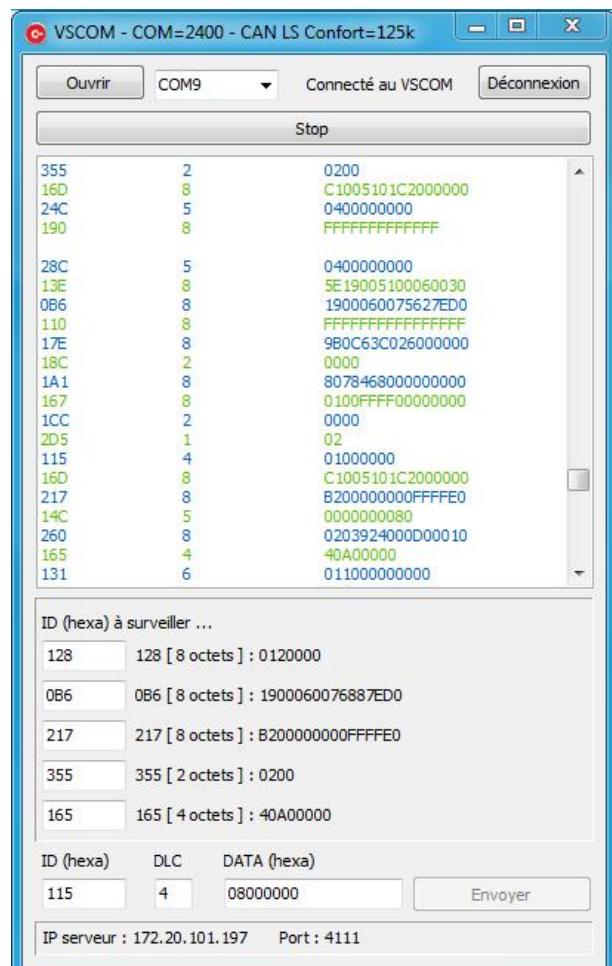
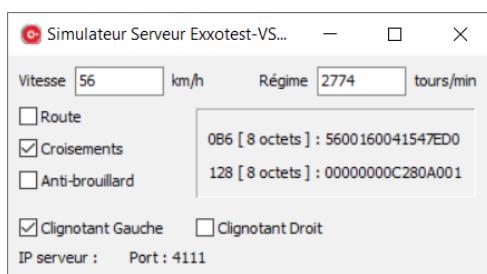
TP – Défi 1 – demande d'une trame moteur ou état des feux en mode console : client TCP

La maquette MT-CAN-LIN-BSI (Exxotest) est un support pédagogique composé d'éléments réels conçu pour l'étude et la compréhension des réseaux de communication utilisés sur les véhicules actuels. Un serveur connecté via une interface USB/CAN affiche toutes les trames circulant sur le bus CAN confort.

Quel est l'intérêt d'utiliser ici un environnement client-serveur ?

Après s'être assuré du bon fonctionnement du serveur connecté via le module VSCOM à la maquette Exxotest, il nous faut coder une application cliente TCP/IP permettant de demander les trames d'identifiants 0x128 et 0x0B6 : pour la trame 0x128 par exemple, il faut envoyer "128" au serveur.

Pour plus de commodité lors des tests, un serveur de simulation : SimulationExcotestVSCOM.exe est disponible.

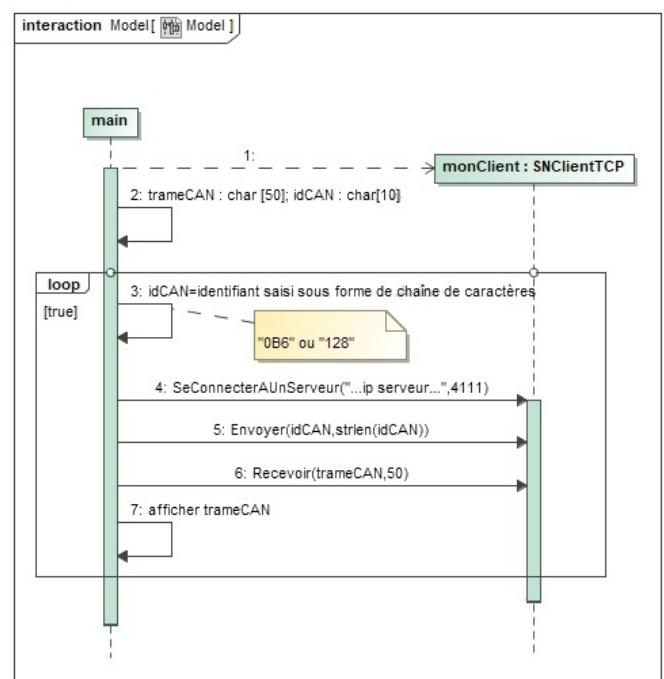
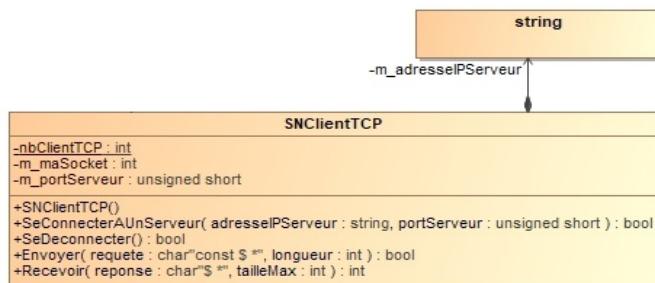


Que signifient les identifiants 128 et 0B6 ?

Trame d'état des voyants des feux :							
Identifiant 0x128							
8 octets de données (DLC=8)							
5 ^{ème} octet : état des feux de signalisation							
7	6	5	4	3	2	1	0
Position	Croise.	Route	AB Av.	AB Ar.	Clign. D	Clign. G	

Trame de vitesse de déplacement et de rotation du moteur :							
Identifiant 0x0B6							
8 octets de données (DLC=8)							
1 ^{er} octet : régime moteur							
Régime moteur \leftrightarrow nombre transmis * 100 / 3.1, exprimé en tr/min.							
3 ^{ème} : vitesse du véhicule							
Vitesse du véhicule \leftrightarrow nombre transmis * 2.55, exprimé en km/h.							

La classe SNClientTCP est donnée :



Coder et tester le programme principal correspondant au diagramme de séquence suivant.

Votre code :

TP – Défi 2 – la classe CombineC4

 A partir du diagramme de classe donné, coder la déclaration de la classe dans `CombineC4.h`.



Votre code, n'écrire que les attributs et méthodes relatives au moteur :

```
CombineC4
-idFeux : char [0..*]
-idMoteur : char [0..*]
-donneeFeux : char [0..*]
-donneeMoteur : char [0..*]
-regime : int
-vitesse : int
-clignoDroit : bool
-clignoGauche : bool
-brouillard : bool
-route : bool
-croisement : bool
-ChaineHexaVersInt( ChaineHexa : char$ "", NbCar : int ) : unsigned int
+CombineC4()
+EnregistrerDonnée( id : char$ "", donnée : char$ "" ) : void
+CalculerRegime() : void
+CalculerVitesse() : void
+ExtraireEtatFeux() : void
+Vitesse() : int
+Regime() : int
+ClignoDroit() : bool
+ClignoGauche() : bool
+Brouillard() : bool
+Route() : bool
+Croisement() : bool
+DonneesCAN( id : char$ "" ) : char$ ""
```

Codage de la définition des méthodes dans `CombineC4.cpp` :



Quelle fonction permet de copier une chaîne de caractères dans une autre ?

Constructeur : initialiser les chaînes `idFeux` et `idMoteur` respectivement à "128" et "0B6".



Votre code :



Quelle fonction permet de comparer 2 chaînes de caractères ?

EnregisterDonnee : si `id` (passé en argument) est égal à `idFeux`, copier `donnee` dans `donneeFeux`. Même chose avec `idMoteur`.



Votre code :

 **DonneesCAN** : en suivant le même principe qu'EnregistreDonnee, la méthode retourne la bonne donnée en testant l'id.



Votre code :

 Méthodes d'accès : les méthodes Vitesse(), Regime(), Route(), ClignoDroit(), ClignoGauche(), Croisement() et Brouillard() sont chargées de retourner simplement l'attribut privé correspondant au nom de la méthode d'accès.



Un exemple de code d'une des méthode d'accès :

 Coder le programme principal afin de vérifier le code déjà produit : avant de coder les méthodes CalculerVitesse(), CalculerRegime() et ExtraireEtatFeux(). Compléter ainsi le programme principal (du défi précédent) en ajoutant la création (l'instanciation) d'un objet monCombine de la classe CombineC4 (au début du main).

Après la réception de la trame, stocker la donnée dans l'objet CombinéC4 en appelant la méthode EnregistrerDonnee : Pour la trame "0B6 [8 octets] : 5600160041547ED0", la donnée est "5600160041547ED0". Il faut donc passer en argument de la méthode EnregistrerDonnee, l'adresse du tableau trameCAN augmentée de 19 cases mémoires : soit trameCAN + 19. Vérifier le bon enregistrement en affichant le résultat de la méthode DonneesCAN.

Il faut bien-sûr essayer les 2 identifiants "128" et "0B6".



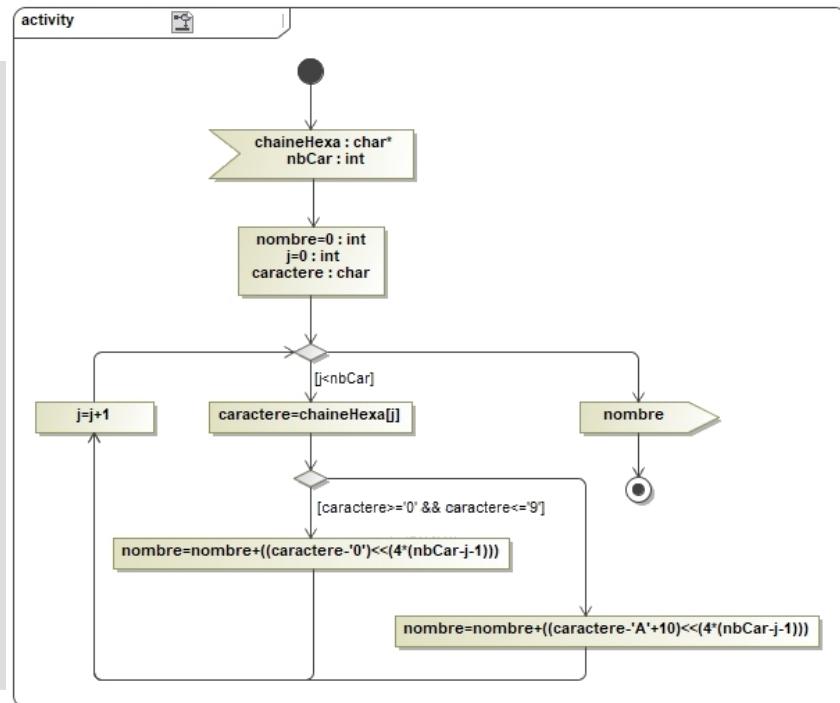
Votre code du programme principal:

TP – Défi 3 – calcul de la vitesse, du régime et extraction des feux

Codage de la définition des méthodes de la classe CombineC4 restantes :

Coder la méthode ChaineHexaVersInt, à partir du diagramme d'activité suivant :

Le code :



La donnée "5600160041547ED0" contient 16 caractères hexadécimaux qui représentent la valeur de 8 octets (un octet est bien représenté par 2 caractères hexadécimaux). la valeur de la vitesse est contenue dans le 3ème octet de la donnée, c'est à dire dans les caractères hexadécimaux d'indice 4 et 5 de la donnée : "16" dans notre exemple. La chaîne "16" est codée en ASCII et représente le nombre hexadécimal 0x16, soit 22 en décimal. D'après la documentation constructeur, la vitesse vaut 2,55 fois ce nombre : soit 56,1km/h.

Donner la valeur des 8 octets de la donnée ASCII "5600250041547ED0" :

Donner la valeur de la vitesse du véhicule correspondant à cette donnée :

La méthode ChaineHexaVersInt permet de convertir en entier une chaîne hexadécimale codée en ASCII : si donneeMoteur vaut "5600250041547ED0", ChaineHexaVersInt(donneeMoteur,2) donne la valeur des 2 premiers caractères hexadécimaux "56", soit le nombre 86. ChaineHexaVersInt(donneeMoteur+2,2) donne la valeur de l'octet codé ASCII suivant puisque nous avons fait un décalage de +2 dans la mémoire (2 caractères hexa ASCII = 1octet)...

Coder la méthode CalculerVitesse chargée de stocker la vitesse dans l'attribut vitesse.



Le code de la méthode CalculerVitesse :

- Modifier le main afin d'appeler la méthode CalculerVitesse() si l'identifiant est "0B6", et d'afficher la valeur de la vitesse (en km/h) en appelant la méthode d'accès adéquate.



Votre code :

Le régime moteur dépend des 2 premiers caractères ASCII de la donnée moteur (le 1er octet). Cet octet doit être multiplié par 100 et divisé par 3,1 d'après la documentation constructeur.

- Coder la méthode CalculerRegime chargée de stocker le régime moteur dans l'attribut regime.

- Modifier le main afin d'appeler la méthode CalculerRegime() si l'identifiant est "0B6", et d'afficher la valeur du régime moteur (en tours/minute) en appelant la méthode d'accès adéquate.

Pour coder la méthode ExtraireEtatFeux : il s'agit de convertir la chaîne ASCII hexa en nombre entier comme précédemment. L'octet concerné est le 5ème octet de la trame d'identifiant "128".



Quelles indices de la chaîne de caractères donneeFeux sont concernés ?

- Dans la méthode ExtraireEtatFeux, extraire le bon octet de la trame en appelant la méthode ChaineHexaVersInt :
unsigned char octet = ChaineHexaVersInt(donneeFeux + ..., 2). Afficher et vérifier la valeur de cet octet.

Il s'agit maintenant d'extraire chaque bit de l'octet. D'après la documentation constructeur, les feux de route sont représentés par le bit 5, les bits allant de 0 (lsb) à 7 (msb) :

L'instruction : route = octet & (0x01 << 5); permet de stocker le bit 5 de la variable 8 bits "octet" dans la variable booléenne "route". En effet si b5 est à 1, route=true, sinon route=false :

octet	=	b7	b6	b5	b4	b3	b2	b1	b0
<u>0x01<<5</u>	=	0	0	1	0	0	0	0	0
&	=	0	0	b5	0	0	0	0	0

Donner les instructions permettant de stocker l'état du clignotant droit (bit 2), du clignotant gauche (bit 1), de l'anti-brouillard (bit 3) et des feux de croisement (bit 6) :

Terminer le code de la méthode ExtraireEtatFeux. Dans le programme principal : si l'identifiant est "128", extraire l'état des feux, puis afficher leur état en testant successivement les 5 méthodes d'accès (Route(), ClignoDroit()...).

Scrutation automatique de l'état du véhicule : l'utilisateur ne saisie plus l'identifiant, mais ce dernier vaudra tantôt "0B6" tantôt "128".

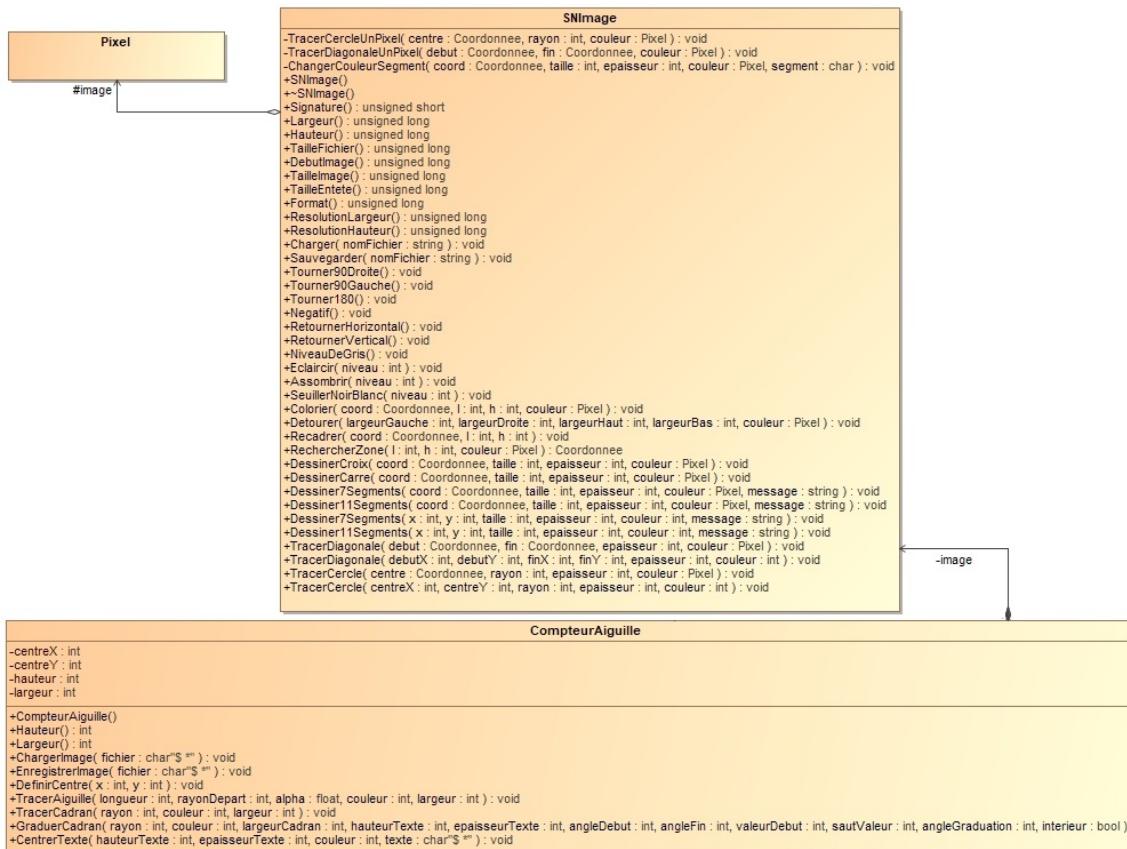
A chaque passage dans la boucle infinie du programme principal, si idCAN vaut "128" : copier "0B6" dans idCAN. Dans le cas contraire copier "128" dans idCAN. Quelque soit l'identifiant, calculer la vitesse et le régime, extraire l'état des feux, puis afficher toutes les informations du combiné. En fin de boucle attendre 1s (utiliser la fonction Sleep()). La fonction clrscr() permet d'effacer le contenu de la console.

Le code de votre programme principal permettant une scrutation automatique de l'état du véhicule :

TP – Défi 4 – classe CompteurAiguille : création d'une image

Cette partie vise à dessiner dans une image le compteur de vitesse.

Nous utiliserons la classe SNImage qui permet d'ouvrir une image bmp et d'y tracer des formes ou bien du texte. Le diagramme de classe suivant est repris en annexe, en version complète et élargie. Pour tester la classe CompteurAiguille, il est nécessaire de créer un nouveau projet en mode console.



Coder la déclaration de la classe dans CompteurAiguille.h

Le code (sans la méthode GraduerCadran) :

```

class CompteurAiguille {
public:
    CompteurAiguille();
    ~CompteurAiguille();
    int Hauteur();
    int Largeur();
    void ChargerImage(char*$\0);
    void EnregistrerImage(char*$\0);
    void DefinirCentre(int x, int y);
    void TracerAiguille(int longueur, int rayonDepart, float alpha, int couleur, int largeur);
    void TracerCadran(int rayon, int couleur, int largeur);
    void GraduerCadran(int rayon, int couleur, int largeurCadran, int hauteurTexte, int epaisseurTexte, int angleDebut, int angleFin, int valeurDebut, int sautValeur, int angleGraduation, bool interieur);
    void CentrerTexte(int hauteurTexte, int epaisseurTexte, int couleur, char$texte$);
private:
    int centreX;
    int centreY;
    int hauteur;
    int largeur;
};
  
```

Codage de la définition des méthodes dans CompteurAiguille.cpp :

-  Constructeur : mettre à 0 tous les attributs de type int.
-  Méthodes d'accès : coder les méthodes Hauteur() et Largeur() qui permettent d'accéder aux attributs hauteur et largeur.
-  ChargerImage : appeler la méthode Charger() avec l'objet image. stocker la hauteur et la largeur de l'image dans les attributs de CompteurAiguille correspondants.
-  EnregistrerImage : appeler la méthode Sauvegarder avec l'objet image.
-  DefinirCentre : stocker les 2 arguments x et y dans les attributs de CompteurAiguille correspondants.
-  TracerCadran : appeler la méthode TracerCercle avec l'objet image. La couleur est composée des 3 octets bleu, vert et rouge : ainsi 0x0000ff est du rouge, 0xff0000 du bleu, 0x000000 du noir, 0xffffffff du blanc, 0x505050 du gris...
-  Tester les méthodes précédentes en chargeant une image blanche de 600x600, en affichant la hauteur et la largeur stockées dans les attributs de l'objet créé.
-  Dessiner un cercle centré au milieu de l'image puis sauvegarder l'image sous un autre nom.



Le code du programme principal :

TP – Défi 5 – tracé de l'aiguille

Codage de la définition des méthodes de la classe CompteurAiguille restantes :

 Coder la méthode TracerAiguille:

L'argument rayonDepart permet de faire partir l'aiguille à partir d'un rayon de départ au lieu de partir du centre du cadran. Si rayonDepart=0, l'aiguille partira du centre défini par les attributs centreX et centreY de la classe. Créer 4 entiers permettant de stocker les coordonnées du début et de la fin de l'aiguille : Xdebut, Ydebut, Xfin et Yfin.

 En vous aidant d'un croquis, donner l'expression de Xfin et Yfin en fonction des coordonnées du centre, de la longueur et de l'angle alpha : il faudra utiliser le sinus et le cosinus de l'angle. L'argument alpha sera exprimé en degré, il faudra donc dans le sinus ou le cosinus multiplier alpha par pi et diviser le tout par 180 afin d'obtenir la valeur de l'angle en radian. Remarque : le premier pixel en haut à gauche a les coordonnées (0,0).

 Donner l'expression de Xdebut et Ydebut en fonction des coordonnées du centre, du rayon de départ et de l'angle alpha.

Appeler ensuite la méthode TracerDiagonale avec l'objet image.

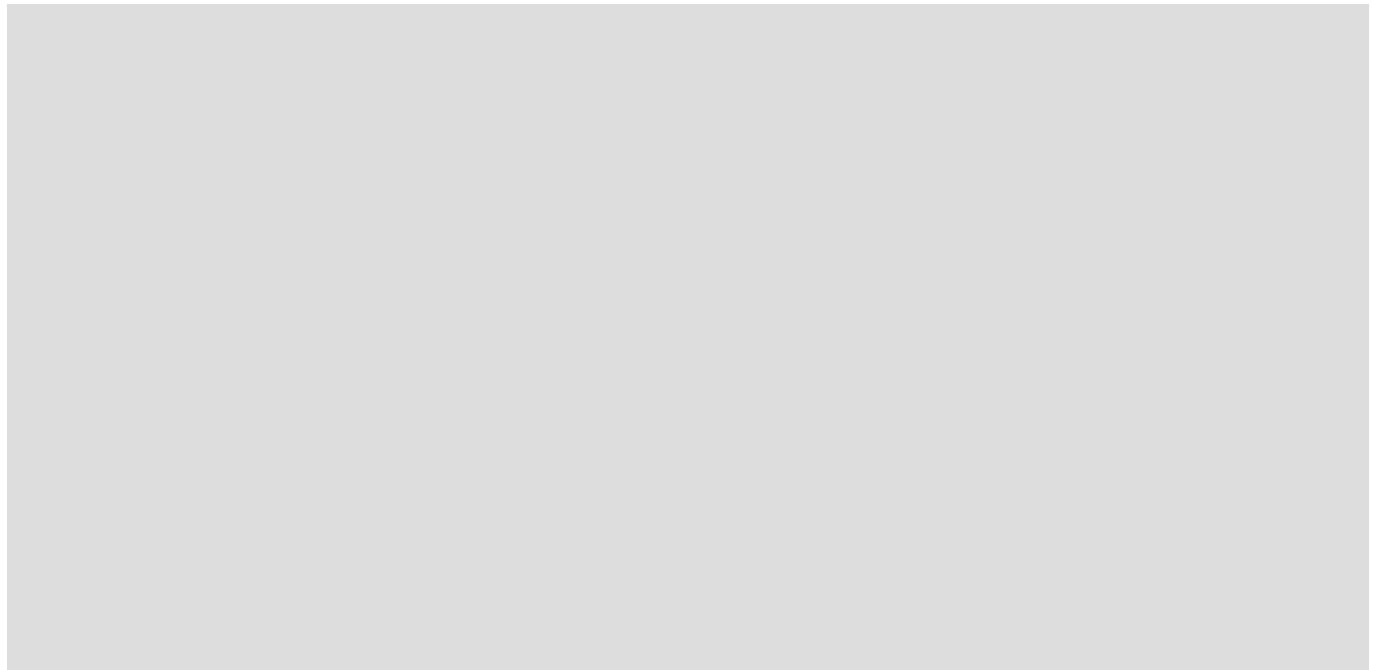
 Le code de la méthode TracerAiguille :

Coder la méthode CentreTexte :

Il faut déterminer les coordonnées du début du texte en fonction de sa longueur (strlen) mais aussi de la hauteur du texte (la largeur d'un caractère est la moitié de sa hauteur). Appeler ensuite la méthode Dessiner11Segments avec l'objet image.



Le code de la méthode CentreTexte :



Pour la suite, on donne :

```
void CompteurAiguille::GraduerCadran(int rayon, int couleur, int largeurCadran, int hauteurTexte,
    int epaisseurTexte, int angleDebut, int angleFin, int valeurDebut, int sautValeur, int
    angleGraduation, bool interieur)
{
    int marge, centrage, nbValeur=(angleFin-angleDebut)/angleGraduation;
    if(intérieur) marge=-hauteurTexte-largeurCadran;
    else marge=hauteurTexte+largeurCadran;
    if(angleFin>angleDebut)
        for(int i=angleDebut;i<=angleFin;i+=angleGraduation)
    {
        char texte[10];
        sprintf(texte,"%d",valeurDebut);
        valeurDebut+=sautValeur;
        centrage=hauteurTexte*strlen(texte)/2-hauteurTexte/4;
        int Y=centreY-(rayon+marge)*sin(i*M_PI/180);
        int X=centreX+(rayon+marge)*cos(i*M_PI/180);
        image.Dessiner7Segments(X-centrage,Y,hauteurTexte,epaisseurTexte,couleur,texte);
        TracerAiguille(rayon-8*largeurCadran,rayon-15*largeurCadran,i,couleur,largeurCadran);
    }
    else
        for(int i=angleDebut;i>=angleFin;i-=angleGraduation)
    {
        char texte[10];
        sprintf(texte,"%d",valeurDebut);
        valeurDebut+=sautValeur;
        centrage=hauteurTexte*strlen(texte)/2-hauteurTexte/4;
        int Y=centreY-(rayon+marge)*sin(i*M_PI/180);
        int X=centreX+(rayon+marge)*cos(i*M_PI/180);
        image.Dessiner7Segments(X-centrage,Y,hauteurTexte,epaisseurTexte,couleur,texte);
        TracerAiguille(rayon-8*largeurCadran,rayon-15*largeurCadran,i,couleur,largeurCadran);
    }
}
```

 Coder le programme principal permettant de tracer le cadran ainsi que l'aiguille de la vitesse (une variable entière vitesse sera créée et initialisée à une valeur permettant les tests de la classe CompteurAiguille).
Appeler successivement les méthodes ChargerImage, DefinirCentre, TracerCadran, GraduerCadran.

 Donner l'expression de l'angle alpha en fonction de la vitesse : pour trouver la relation linéaire $\alpha = a \cdot \text{vitesse} + b$, il est nécessaire de choisir 2 points puis d'en déduire les valeurs des constantes a et b (par exemple : pour $\alpha = 90^\circ$, $\text{vitesse} = 70\text{km/h}$...).

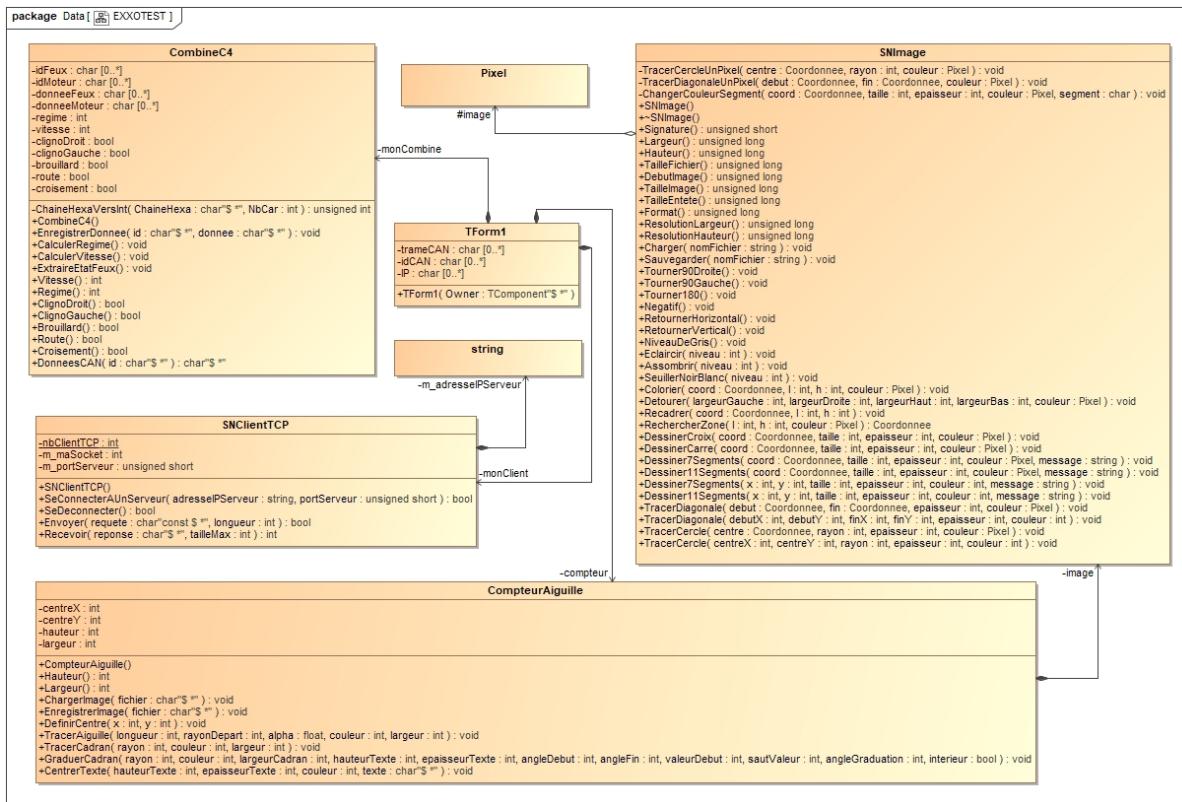
Appeler alors les méthodes TracerAiguille et CentrerTexte afin d'afficher au centre du cadran la valeur de la vitesse (utiliser la fonction sprintf). Ne pas oublier d'enregistrer l'image.

 Le code du programme principal permettant d'obtenir l'image suivante :



TP – Défi 6 – interface graphique finale

Le diagramme complet suivant est repris en annexe sur une page.



Créer une nouvelle application graphique sous C++ Builder. L'IHM comportera 6 images (l'image de fond qui contiendra les 2 cadrants et les images des 5 voyants). Un objet Timer permettra d'envoyer régulièrement les requêtes au serveur.



Ajouter à la classe TForm1 les attributs nécessaires (voir le diagramme de classe).

Dans le constructeur, initialiser la valeur de l'IP. Ajouter la ligne DoubleBuffered=true; afin d'éviter le scintillement de l'image.

Placer dans l'événement OnTimer de l'objet Timer les contenus des 2 programmes principaux ayant permis de tester les classes CombineC4 et CompteurAiguille.

L'état des feux seront stockés dans les attributs "visible" des images des voyants lumineux. A la fin de l'événement, il suffira de charger l'image des combinés dans Image1 en appelant la méthode "LoadFromFile" de l'attribut "Picture" de l'image.

Annexe 1 : norme Bus CAN

12/12/2019

Bus de données CAN — Wikipédia

Historique

Le bus de données CAN est le fruit de la collaboration entre l'Université de Karlsruhe et Bosch.

Il fut d'abord utilisé dans le secteur de l'automobile, mais est actuellement utilisé dans la plupart des industries comme l'aéronautique *via* des protocoles standardisés basés sur le CAN.

Il fut présenté avec Intel en 1985, mais ne fut standardisé par l'ISO qu'au début des années 1991.

En 1992 plusieurs entreprises se sont réunies pour créer le CAN in Automation, une association qui a pour but de promouvoir le CAN.

Couche physique

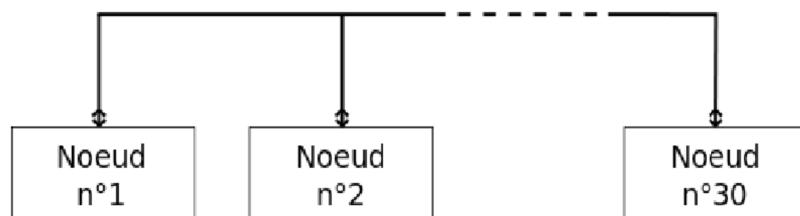
Il existe deux normes pour la couche physique :

- ISO 11898-2 (2003) : CAN « high-speed » (jusqu'à 1Mbits/s),
- ISO 11898-3 (2006) : CAN « low-speed, fault tolerant » (jusqu'à 125kbits/s).

Topologie

CAN est un bus de données série bidirectionnel half-duplex dans le domaine automobile, mais est utilisé en unidirectionnel — simplex — dans l'aéronautique, pour obtenir un comportement déterministe.

Chaque équipement connecté, appelé « nœud », peut communiquer avec tous les autres.



Pour un bus de données CAN « basse vitesse », le nombre de nœuds est limité à 20. Pour un bus de données CAN « haute vitesse », il est limité à 30.

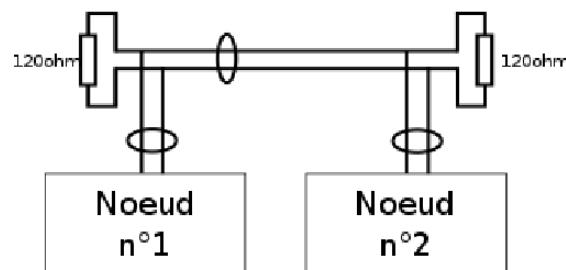
Support

Chaque nœud est connecté au bus par l'intermédiaire d'une paire torsadée (blindée ou non).

Les deux extrémités du bus doivent être rebouclées par des résistances de $120\ \Omega$ (tolérance entre $108\ \Omega$ et $132\ \Omega$).

12/12/2019

Bus de données CAN — Wikipédia



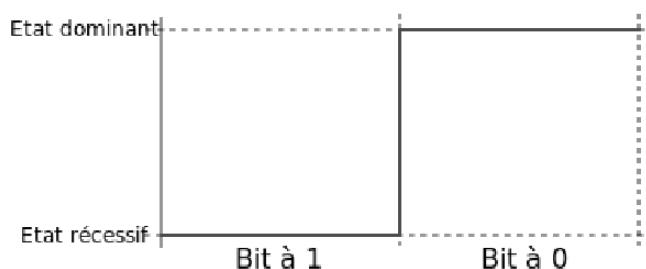
L'accès au bus de données CAN suit la technique CSMA/CR (écoute de chaque station avant de parler mais pas de tour de parole, résolution des collisions par priorité).

La longueur maximale du bus est déterminée par la vitesse utilisée :

Vitesse (kbit/s)	Longueur (m)
1000	30
800	50
500	100
250	250
125	500
62,5	1000
20	2500
10	5000

Encodage des bits

L'encodage utilisé est de type NRZ (non retour à 0) :



États logiques et Niveaux électriques

Les nœuds sont câblés sur le bus par le principe du « OU câblé » du point de vue électrique (« ET câblé » du point de vue logique), ce qui veut dire qu'en cas d'émission simultanée de deux nœuds, la valeur 0 écrase la valeur 1.

On dit donc :

- que l'état logique « 0 » est l'état « dominant »,
- que l'état logique « 1 » est l'état « récessif ».

Les états logiques et les niveaux électriques utilisés entre les deux lignes de la paire différentielle pour le CAN low-speed sont les suivants :

12/12/2019

Bus de données CAN — Wikipédia

Etat logique	$V_{CANH-GND}$	$V_{CANL-GND}$	$V_{CANH-CANL}$
Récessif ou « 1 »	1,75 V	3,25 V	-1,5 V
Dominant ou « 0 »	4 V	1 V	3 V

Les états logiques et les niveaux électriques utilisés entre les deux lignes de la paire différentielle pour le CAN high-speed sont les suivants :

Etat logique	$V_{CANH-GND}$	$V_{CANL-GND}$	$V_{CANH-CANL}$
Récessif ou « 1 »	2,5 V	2,5 V	de 0 à 0,5 V
Dominant ou « 0 »	3,5 V	1,5 V	de 0,9 à 2 V

Temps et vitesse

La durée d'un bit est appelée « Nominal Bit Time ».

Chaque bit est constitué de plusieurs segments cadencés par l'horloge interne de chaque nœud :

- segment de synchronisation,
- segment de propagation,
- segment de phase buffer n° 1,
- segment de phase buffer n° 2.

Time Quantum

Le « Time Quantum » est l'unité de temps construite à partir de la période de l'oscillateur interne de chaque nœud.

La fréquence du bus étant au maximum de 1 MHz et celles des oscillateurs de plusieurs MHz, le « Time Quantum » vaut généralement plusieurs périodes d'horloge (entre 1 et 32 fois).

La durée de chaque segment est la suivante :

Segment	Durée en « Time Quantum »
Synchronisation	1
Propagation	de 1 à 8
Phase buffer n° 1	de 1 à 8
Phase buffer n° 2	de 2 à 8

Ainsi la durée d'un bit peut varier de 5 à 25 « Time Quantum ».

Plus la fréquence de l'horloge interne du nœud est importante, plus la durée du « Time Quantum » pourra être faible, plus les 3 derniers segments compteront de « Time Quantum » et meilleure sera la précision de la synchronisation.

Segment de synchronisation

Le segment de synchronisation est utilisé pour synchroniser les différents nœuds.

La transition de 0 à 1 ou de 1 à 0, effectuée pour le nœud émetteur, doit s'effectuer dans ce segment. Si pour un nœud récepteur cette transition n'a pas lieu dans ce même segment, c'est qu'il est désynchronisé. Il s'agit d'une erreur de phase.

Grâce au bit de transparence, cette vérification peut être faite au moins tous les 5 bits (pour les premiers champs de la trame dans lequel il est utilisé).

Segment de propagation

Le segment de propagation est utilisé pour compenser les phénomènes de propagation sur le bus.

Segments de phase

Les segments de phase sont utilisés pour compenser les erreurs de phase détectées lors des transitions.

La durée de ces segments peut varier en cas de resynchronisation.

Point d'échantillonnage

Le point d'échantillonnage ou « Sample point » est l'instant où on lit la valeur du bit sur le bus. Celui-ci intervient entre les 2 segments de phase.

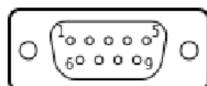
Synchronisation

Il existe 2 types de synchronisation :

- la hard-synchronisation qui consiste à se resynchroniser brutalement dès qu'une transition est détectée dans le segment de synchronisation, notamment lors du SOF (Start Of Frame) d'une nouvelle trame,
- la resynchronisation qui consiste à allonger le segment de phase buffer n°1 ou à diminuer le segment de buffer phase n°2, ce qui a 2 effets :
 - déplacer le point d'échantillonnage,
 - diminuer ou augmenter la durée du bit et ainsi améliorer la synchronisation du bit suivant.

Connecteur

Le brochage sur le bus de données CAN est normalisé et utilise un connecteur DE-9 :



Broche	Description
1	(Réservé)
2	CANL
3	Masse
4	(Réservé)
5	Blindage (optionnel)
6	Masse
7	CANH
8	(Réservé)
9	Alimentation externe (optionnel)

Couche liaison de données

Il existe également 2 standards pour la couche de liaison de données :

- ISO 11898 part A → CAN 2.0A « standard frame format » (identification sur 11bits),
- ISO 11898 part B → CAN 2.0B « extended frame format » (identification sur 29bits).

Il existe plusieurs types de trame :

- Trame de données,
- Trame de requête,
- Trame d'erreur,
- Trame de surcharge.

Entre 2 trames, les émetteurs doivent respecter une pause (période d'inter-trame) équivalente à la durée de 3 bits pendant laquelle le bus est maintenu à l'état récessif.

Trame de données

La trame de données sert à envoyer des informations aux autres nœuds.

Une trame de données se compose de 7 champs différents :

- Le début de trame ou SOF (Start Of Frame) matérialisé par 1 bit dominant,
- Le champ d'arbitrage (identificateur) composé de 12 ou 32 bits,
- Le champ de commande (ou de contrôle) composé de 6 bits,
- Le champ de données composé de 0 à 64 bits (de 0 à 8 octets),
- Le champ de CRC composé de 16 bits,
- Le champ d'acquittement composé de 2 bits,
- La fin de trame ou EOF (End of Frame) matérialisée par 7 bits récessifs.

Champ d'arbitrage	Champ de commande	Champ de données	Champ de CRC	ACK	EOF
1 bit	12 ou 32 bits	6 bits	de 0 à 64 bits	16 bits	2 bits

12/12/2019

Bus de données CAN — Wikipédia

Dans chaque champ de la trame, les bits sont transmis du plus fort au plus faible.

Champ d'arbitrage

Le champ d'arbitrage est composé de 11 bits d'identification pour CAN 2.0A et 29 bits pour CAN 2.0B suivis par le bit RTR (Remote Transmission Request) qui est dominant.

Ce champ sert d'identifiant pour la donnée transportée dans le champ de données.

Les 11 bits de CAN 2.0A autorisent $2^{11} = 2048$ combinaisons.

Les 29 bits de CAN 2.0B autorisent $2^{29} = 536\,870\,912$ combinaisons.

Champ de commande

Le champ de commande est composé de 6 bits.

Le bit de poids fort est utilisé pour différencier le type de trame :

- Dans le cas d'une trame standard (sur 11 bits), le bit de poids fort est dominant,
- Dans le cas d'une trame étendue (sur 29 bits), le bit de poids fort est récessif,

Le bit suivant n'est pas utilisé.

Les 4 bits de poids faibles appelés DLC (Data Length Code) représentent le nombre d'octets du champ de données (PAYLOAD) embarqué.

Ce nombre d'octets peut varier de 0 à 8, soit 9 valeurs stockées avec les 4 bits du champ DLC. Les valeurs DLC supérieures à 9 ne seraient donc pas utilisées (de 9 à 15).

Champ de données

Le champ de données peut varier de 0 à 8 octets.

Dans le cas d'une trame de requête le champ de données est vide.

Champ de CRC

Le champ est composé de 15 bits de CRC (Cyclic Redundancy Check) et d'un bit dit délimiteur (« CRC delimiter ») toujours récessif.

Le CRC est calculé à partir de l'ensemble des champs transmis jusque-là (c'est-à-dire le SOF, le champ d'arbitrage, le champ de commande et le champ de données; les bits de transparence ne sont pas pris en compte). L'ensemble constitue le polynôme $f(x)$.

L'algorithme consiste dans un premier temps à multiplier $f(x)$ par 2^{15} .

Ensuite le polynôme $f(x)$ est divisé (modulo 2) par le polynôme $g(x)=x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+x^0$.

Une fois les divisions successives effectuées, le reste constitue la séquence de CRC.

La distance de Hamming de l'algorithme utilisé est de 6, ce qui signifie que 5 erreurs au maximum sont détectables.

12/12/2019

Bus de données CAN — Wikipédia

Grâce à ce système de détection, le taux d'erreur enregistré est très faible (inférieur à $4,6 \cdot 10^{-11}$). De plus, le réseau est capable de différencier les erreurs ponctuelles des erreurs redondantes. Ainsi, tout périphérique défaillant peut être déconnecté du réseau afin de limiter les perturbations. Le réseau entre alors en mode « dégradé ».

Champ d'acquittement ACK

Le champ est composé d'un bit d'acquittement ACK (ACKnowledge) et d'un bit dit délimiteur (« ACKnowledge delimiter ») toujours récessif.

Tous les récepteurs qui ont bien reçu le message doivent l'acquitter en émettant un bit dominant pendant la durée du bit ACK, ce qui permet au nœud émetteur de savoir qu'au moins un des nœuds récepteurs a reçu le message.

Si un nœud récepteur n'a pas ou mal reçu le message, il ne peut pas se servir de ce mécanisme pour signaler l'erreur, puisqu'il suffit qu'une station réceptrice envoie un bit dominant pour masquer tous les bits récessifs. Pour signaler le dysfonctionnement, il doit émettre une trame d'erreur.

Trame de requête

La trame de requête sert à demander une donnée à un autre nœud. Elle est similaire à la trame de données hormis :

- Le champ de données est vide,
- Le bit RTR du champ d'arbitrage est récessif,
- Les 4 bits DLC du champ de commande correspondent au nombre d'octets attendus dans la réponse.

À noter que le fait que le bit RTR soit récessif dans le cas d'une trame de requête fait que si une trame de données est émise simultanément avec le même champ d'arbitrage, c'est la trame de données qui est prioritaire.

Bit de transparence

Afin de sécuriser la transmission des messages, la méthode du « bit-stuffing » est utilisée.

Elle consiste, dans le cas où l'on a émis 5 bits de même polarité d'affilée, d'ajouter à la suite un bit de polarité contraire, pour casser des chaînes trop importantes de bits identiques. Cette méthode n'est appliquée que sur les champs SOF, d'arbitrage, de commande, de data et de CRC (délimiteur exclu).

Par exemple, « 1111 1110 » deviendra « 1111 1011 0 ».

Priorité de transmission

Que se passe-t-il si plusieurs nœuds tentent de transmettre simultanément?

Il existe une procédure d'accès au bus à laquelle chaque nœud doit se soumettre :

- lorsqu'il transmet un bit, le nœud doit rester à l'écoute du bus. Dit autrement, après avoir envoyé un bit, il lit le bus et vérifie que le bit lu correspond à celui transmis,

12/12/2019

Bus de données CAN — Wikipédia

- s'il y une différence (forcément sur un bit récessif), c'est qu'un autre nœud l'a écrasé par un bit dominant,
- le nœud doit alors interrompre sa transmission, monitorer le bus pour attendre la fin de la transmission, puis tenter à nouveau d'envoyer son message.

Ainsi une priorité est réalisée grâce au champ d'arbitrage.

Plus celui-ci est petit, plus il contient des bits de poids forts à 0 (dominant), plus il sera prioritaire.

Cette phase de priorisation ou d'arbitrage prend fin au bit RTR.

Trame d'erreur

Dès la détection d'une erreur, le nœud n'attend pas la fin de la trame incriminée, il envoie immédiatement une trame d'erreur pour signaler un problème dans la transmission.

Une trame d'erreur se compose de 2 champs différents :

- Le drapeau d'erreur composé de 6 bits,
- Le délimiteur composé de 8 bits récessifs.

La trame d'erreur peut être :

- « active » si le drapeau d'erreur est composé de 6 bits dominants,
- « passive » si le drapeau d'erreur est composé de 6 bits récessifs.

Erreurs

Un certain nombre d'erreurs sont détectables par les nœuds.

Bit error

Chaque fois qu'un nœud émet un bit sur le bus, il relit le bus et doit retrouver le bit qu'il a écrit. Si sur l'envoi d'un bit récessif, il relit un bit dominant, c'est que celui-ci a été altéré.

Ce mécanisme est identique à celui permettant la priorisation, c'est pourquoi il ne faut pas en tenir compte dans le champ d'arbitrage.

Idem pour le champ d'acquittement, si le bit récessif envoyé par le nœud émetteur devient dominant, c'est simplement qu'un ou plusieurs nœuds récepteur ont confirmé la bonne réception de la trame, ce n'est donc pas une erreur.

Stuff error

Si sur le bus on lit 6 bits de même polarité consécutifs, c'est que le mécanisme du bit de transparence n'a pas été respecté ou qu'un bit a été altéré.

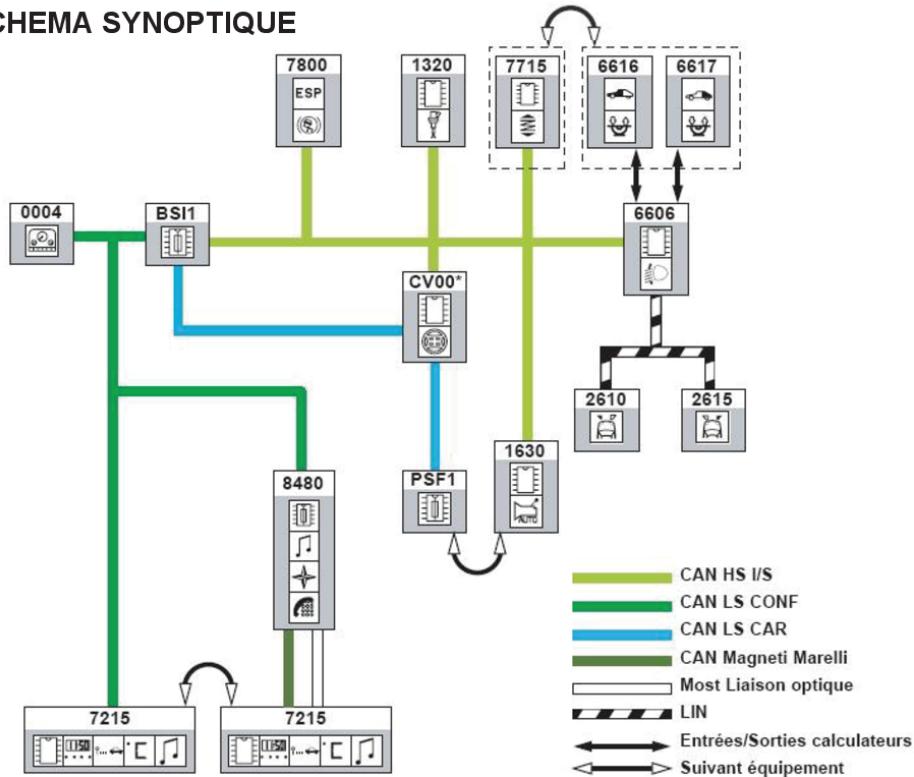
CRC error

Si la valeur de CRC calculée par le nœud récepteur est différente du CRC codé dans la trame par le nœud émetteur, c'est que la trame a été altérée.

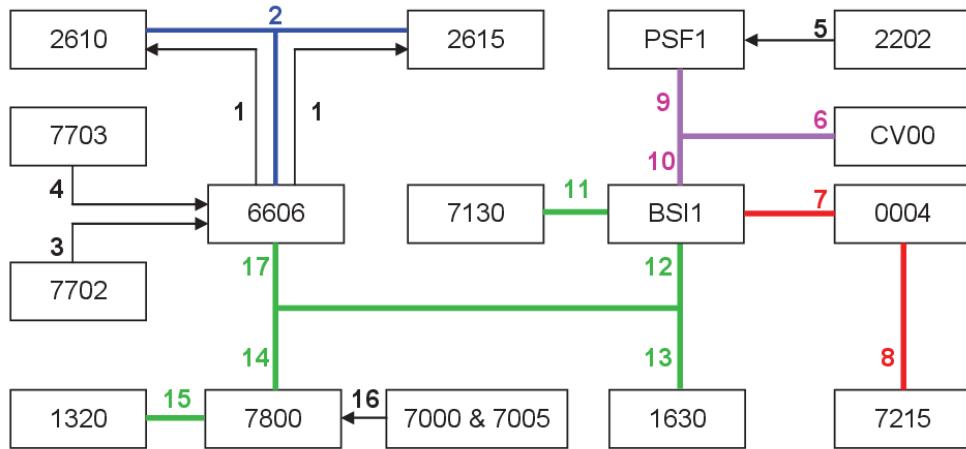
Annexe 2 : notice MT-CAN-LIN-BSI1



2.2. SCHEMA SYNOPTIQUE

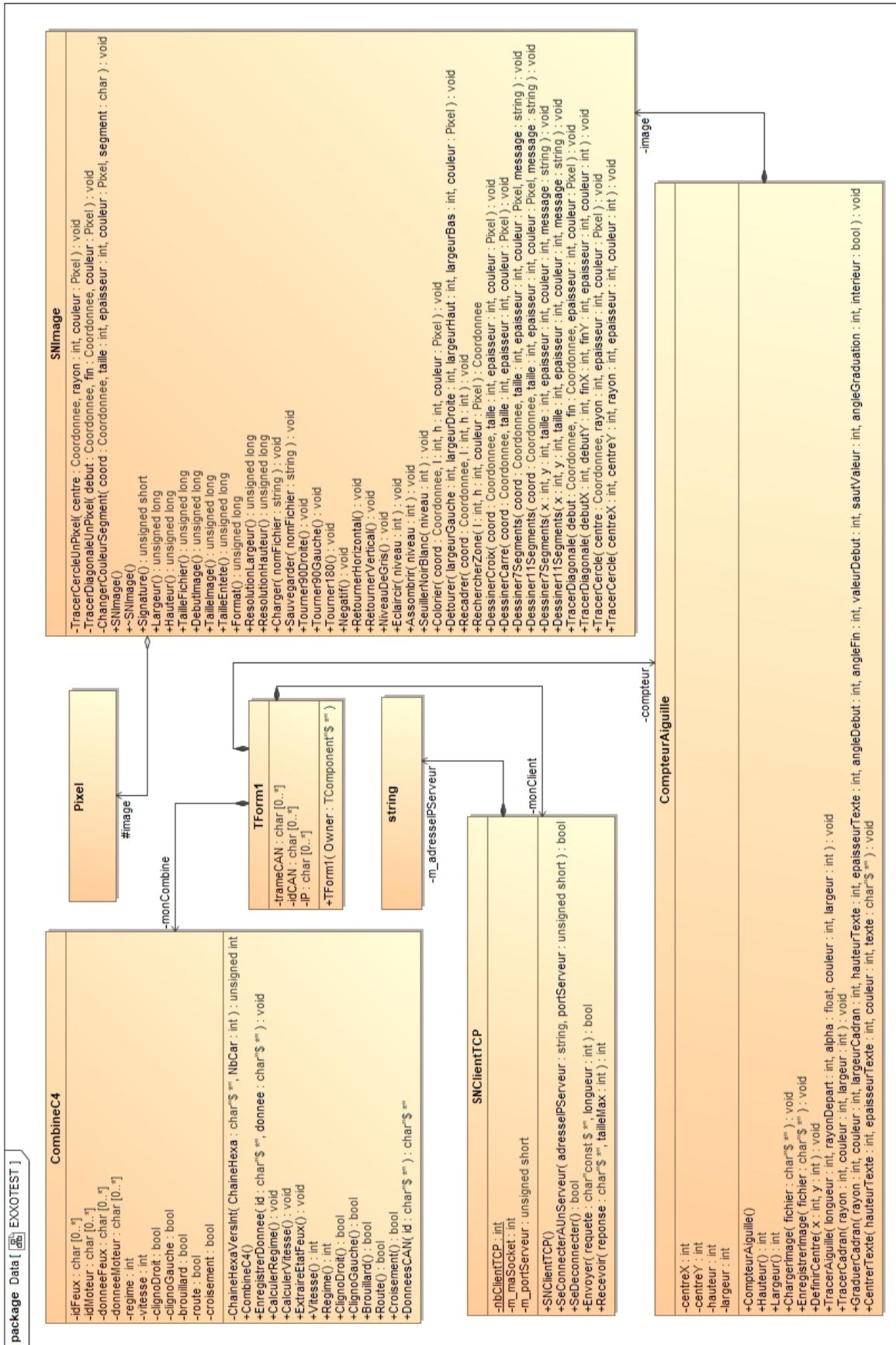


Organes	
BSI1	Boîtier Servitude Intelligent
BSM	Boîtier Servitude Moteur
CV00	Module de commutation sous volant
PSF1	Platine servitude fusible compartiment moteur
0004	Combiné
1320	Calculateur moteur
1630	Calculateur boîte de vitesses automatique
2202	Contacteur de Marche AR (BVM)
2610	Projecteur gauche
2615	Projecteur droit
6606	Boîtier de correction dynamique des projecteurs
6616	Capteur hauteur de caisse avant
6617	Capteur hauteur de caisse arrière
7000 & 7005	Capteur antibloage de roue AVG et AVD
7215	Ecran multifonctions
7130	Capteur d'angle volant
7702	Capteur hauteur de caisse avant
7703	Capteur hauteur de caisse arrière
7715	Calculateur suspension
7800	Calculateur ESP
8410	Autoradio
8480	Emetteur récepteur télématique



N°	Signal	Liaison
1	Commande de correction de site	Filaire
2	Commande de correction d'azimut	LIN
3	Information de hauteur de caisse avant	Filaire
4	Information de hauteur de caisse arrière	Filaire
5	Information Marche AR (BVM)	Filaire
6	Etat des commandes d'éclairage	CAN Car
7	Commande voyants Demande correction d'azimut	CAN Conf
8	Demande activation/inhibition correction d'azimut	CAN Conf
9	Information Marche AR (BVM)	CAN Car
10	Information Marche AR (BVM) / Etat des commandes d'éclairage	CAN Car
11	Information capteur d'angle volant	CAN I/S
12	Information capteur d'angle volant / Information Marche AR (BVM) Etat des commandes d'éclairage / Commande voyants Demande activation/inhibition correction d'azimut	CAN I/S
13	Information Marche AR (BVA)	CAN I/S
14	Info moteur tournant / Vitesse véhicule	CAN I/S
15	Information moteur tournant	CAN I/S
16	Information vitesse de rotation des roues avants	Filaire
17	Information capteur d'angle volant / Information Marche AR Etat des commandes d'éclairage / Commande voyants / Demande correction d'azimut / Info moteur tournant / Vitesse véhicule	CAN I/S

Annexe 3: diagramme de classe Exxotest complet





Module Système 5

Surveillance météorologique



Table des matières

Objectifs, matériels et ressources.....	2
TD 1 - Analyse UML.....	3
TD 2 - Le port série de la station météorologique La Crosse WS2300.....	4
Défi 1 - Création d'un projet C++ sous Linux.....	6
Défi 2 - La classe WS2300 – Initialiser la communication.....	7
TD 3 - Lire des données sur la station météorologique.....	11
Défi 3 - Extraction des données météorologiques.....	14
Défi 4 - Application de surveillance météorologique.....	17
Annexe - Memory MAP FOR WS 2300.....	20

Objectifs, matériels et ressources

Objectifs

Les compétences visées

	Où	Acquis
• C++		
◦ Utiliser une classe dans un programme	Défi 2	
▪ Déclarer un objet d'une classe		
▪ Appeler des méthodes avec un objet	Défi 2	
• Fournir des arguments valides		
• Récupérer éventuellement la valeur de retour	Défi 2	
◦ Écriture de la déclaration d'une classe à partir de son diagramme de classe		
▪ Déclarer des attributs selon leur visibilité	Défi 2	
▪ Déclarer des méthodes selon leur visibilité	Défi 2	
▪ Coder une composition	Défi 2	
◦ Les structures de données		
▪ Déclarer une nouvelle structure	Défi 3	
▪ Accéder aux attributs d'une structure en lecture et en écriture	Défi 3	
◦ Les opérateurs binaires		
▪ l'opérateur binaire ET	TD 3, Défi 3	
▪ les décalages à droite et à gauche	TD 3, Défi 3	
• Numération		
◦ Comprendre et utiliser des masques	TD 3, Défi 3	
◦ Comprendre et utiliser des décalages	TD 3, Défi 3	
• UML		
◦ Savoir analyser un cahier des charges		
▪ Dessiner le diagramme de cas d'utilisation	TD1	
▪ Dessiner le diagramme de déploiement	TD1	

Le matériel

- Station météorologique La Cross WS2300
- Raspberry PI 3



Station météorologique La Crosse WS2300



Les capteurs de la station météorologique



La Raspberry PI 3

Les ressources

- Le plan d'adressage de la station météorologique.
- Le principe de lecture des informations de la station météorologique.
- Les sites de référence des librairies standards du C/C++ : www.cplusplus.com ou cppreference.com.

TD 1 - Analyse UML

Cahier des charges

Une entreprise spécialisée dans le développement des toits végétaux sur Paris souhaite installer des stations météorologiques sur les toits végétaux pour connaître les conditions météorologiques précises de chacun des toits qu'elle gère. Elle souhaite que des alarmes se déclenchent quand les températures franchissent des seuils réglables (gelées, forte chaleur, etc.), que le vent dépasse une vitesse limite ou que le niveau d'eau des pluies montent trop rapidement.

L'entreprise impose l'utilisation de la station météorologique La Crosse WS2300. Cette station est autonome. Il est possible de récupérer les mesures prélevées par cette station météorologique grâce à son port série.

C'est la Raspberry PI 3 qui se chargera de récupérer les mesures prélevées par la station sur son port série. Ensuite, ces informations seront mise à disposition grâce à un serveur TCP qui se trouvera également sur la Raspberry PI 3.



Analyse du cahier des charges

L'objectif de cette partie est, à partir du cahier des charges précédent, de mettre en place les diagrammes UML de première analyse.

? Dessinez à la main le diagramme de cas d'utilisation du projet complet.

? Dessinez à la main le diagramme de déploiement du projet complet.

TD 2 - Le port série de la station météorologique La Crosse WS2300

La vitesse de communication

La documentation de la station météorologique nous indique un débit de 2400 baud 8N1.

? Donner la vitesse de transmission en bit.s⁻¹.

8 :

N :

1 :

? En déduire le nombre de bits transmis sur le port série pour envoyer un octet.

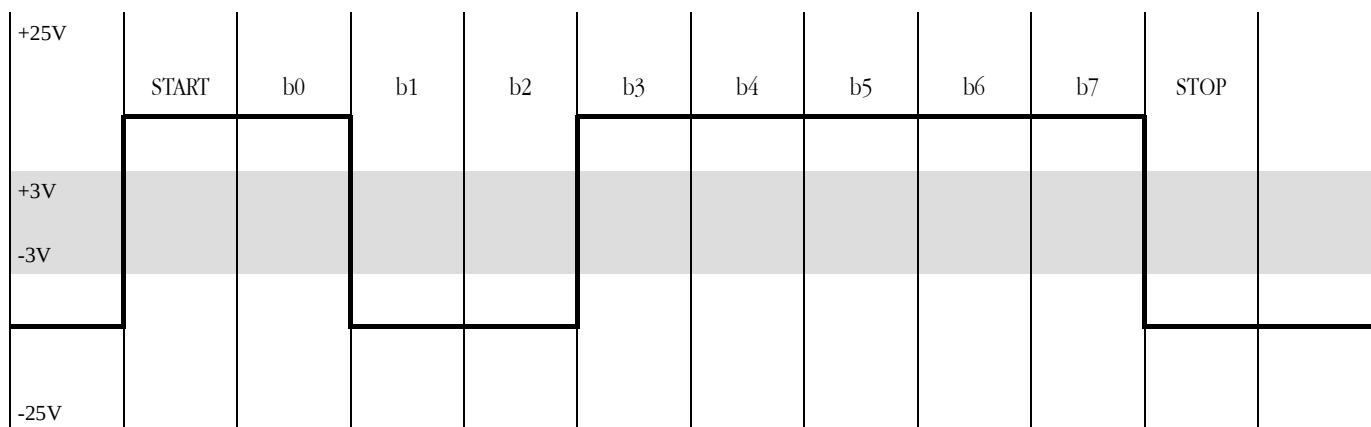
? Calculer la durée de l'émission d'un octet sur le port série.

? Calculer alors le rendement de la liaison série de la station météorologique La Crosse WS2300.

$$\text{rendement} = \frac{\text{nombre de bits de données utiles}}{\text{nombre total de bits transmis}}$$

Les niveaux électriques

Un niveau logique "0" est représenté par une tension de +3 à +25 et un niveau logique "1" par une tension de -3 à -25.



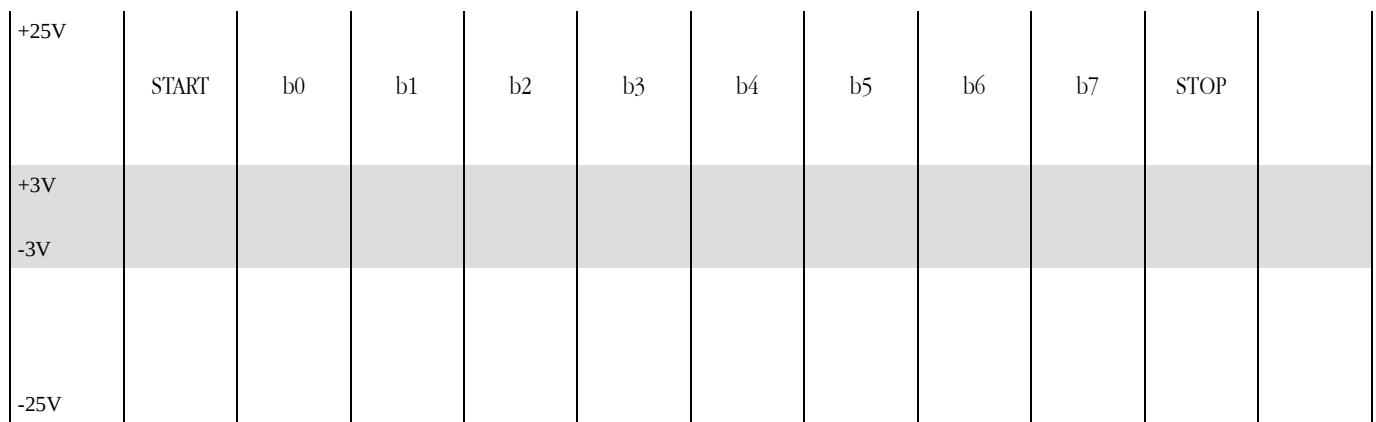
? Écrire sur le chronogramme la valeur binaire de chacun des bits envoyés.



En déduire la valeur en hexadécimal de l'octet transmis.

Cet octet sera le premier à envoyer à la station météorologique. Il permet d'initialiser la communication. Si la station météorologique est disponible, elle répond par un octet valant 0x02.

? Dessiner sur le chronogramme suivant la réponse de la station météorologique. Indiquer la valeur binaire de chacun des bits de la réponse.



Défi 1 - Crédation d'un projet C++ sous Linux

Le développement de l'application qui récupérera les informations météorologiques sur la station WS2300 se fera sur une Raspberry, dotée du système d'exploitation Raspbian (la distribution linux 'Debian' adaptée à la Raspberry).

La première application avec un makefile

- Lancer WinSCP pour naviguer sur les fichiers de la Raspberry. Se loguer avec le compte 'pi' et le mot de passe 'raspberry'.
- Créer un répertoire de travail à votre nom.
- Créer les fichiers main.cpp et makefile dans votre répertoire.
- Écrire un code simple dans le fichier main.cpp qui affiche dans le terminal le texte : « Station Météo WS 2300 ».
- Ouvrir le fichier makefile et écrire les directives de compilation suivantes :

```
all: main.o stationMeteo
main.o: main.cpp
    g++ -Wall -c main.cpp
stationMeteo: main.o
    g++ -Wall main.o -o stationMeteo
clean:
    rm -f main.o stationMeteo
```

- Lancer SmarTTY ou TeraTerm pour obtenir un terminal distant.
- Se déplacer dans votre répertoire. Par exemple, si vous avez créer un répertoire 'toto', taper :

```
cd toto
```

Pour compiler l'ensemble du programme, il suffit de taper :

```
make
```

Si des erreurs apparaissent, les corriger et relancer la compilation. S'il n'y a plus d'erreur de compilation, lancer l'exécutable en tapant :

```
./stationMeteo
```

Ajout de la classe SNPortSerie

Dans ce projet, nous communiquerons avec la station météorologique WS2300 en utilisant un port série virtuel. Vous utiliserez la classe SNPortSerie, disponible dans le répertoire Echange.

- Ajouter les fichiers SNPortSerie.h et SNPortSerie.cpp dans votre répertoire de travail sur la Raspberry.
- Ajouter la directive de compilation de cette classe dans le makefile comme suit :

```
SNPortSerie.o: SNPortSerie.cpp SNPortSerie.h
    g++ -Wall -c SNPortSerie.cpp
```

Dans le fichier makefile, ajouter 'SNPortSerie.o' dans la liste des fichiers de la ligne 'all'.

Dans le fichier makefile, ajouter que l'exécutable final est dépendant de la classe SNPortSerie en le modifiant ainsi :

```
stationMeteo: main.o SNPortSerie.o
    g++ -Wall main.o SNPortSerie.o -o stationMeteo
```

Vérifier que votre compilation fonctionne toujours en tapant la commande :

```
make
```

Ajout de la classe WS2300.

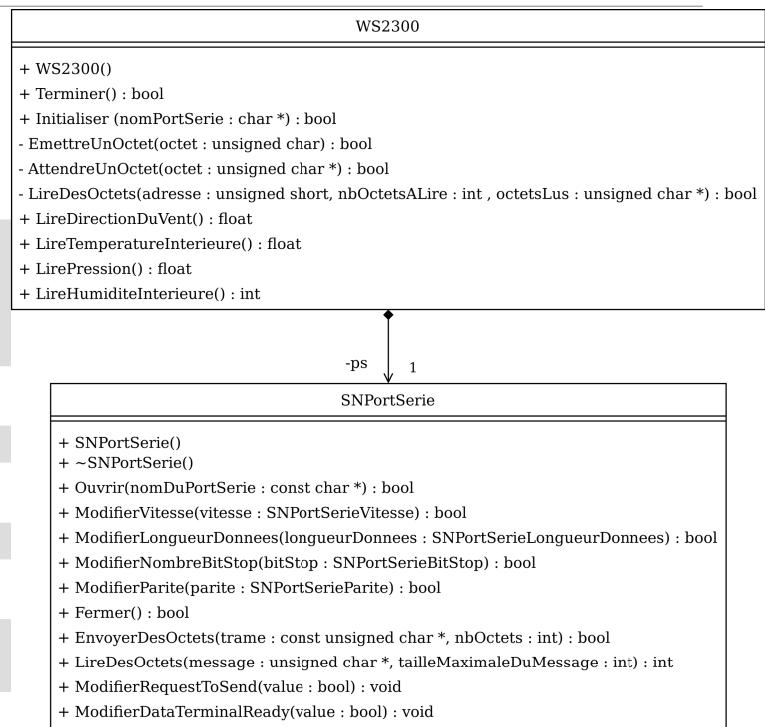
Dans ce projet, vous allez créer une classe nommée WS2300.

- Ajouter les fichiers WS2300.h et WS2300.cpp et modifier le fichier makefile en conséquence. Tester la compilation.

Défi 2 - La classe WS2300 – Initialiser la communication

Analyse du diagramme de classe

Donner le nom du lien entre les classes *WS2300* et *SNPortSerie*. Expliquer ce que ce lien ajoute dans la classe *WS2300*.



Donner le nombre d'attributs privés :

Donner le nombre de méthodes :

Donner le nom des méthodes privées :

Rappeler l'utilité d'un constructeur.

Déclaration de la classe WS2300

A partir du diagramme de classe précédent, écrire en C++ la déclaration de la classe *WS2300*.

Compléter le fichier *WS2300.h* avec la déclaration précédente.

La méthode *Terminer()*

La méthode *Terminer()* doit fermer le port série en utilisant l'objet 'ps'.

Écrire en C++ la définition de la méthode *Terminer()*.

 Ajouter la définition de la méthode *Terminer()* de la classe WS2300 dans le fichier WS2300.cpp.

La méthode *Initialiser()* - première partie

Dans un premier temps, la méthode *Initialiser()* doit ouvrir le port série et le paramétrier suivant la configuration de la station météorologique WS2300 : 2400 baud, 8N1.

 En utilisant l'objet *ps* de la classe *SNPortSerie*, appeler les méthodes qui conviennent pour faire une initialisation complète de la communication sur le port série.

```
bool WS2300::Initialiser(char * nomPortSerie)
{
    bool ret = ps.Ouvrir(nomPortSerie);
    if(!ret)
    {
        return false;
    }

    // Paramétrage de la vitesse : si echec, appeler
    Terminer() et renvoyer false.
    ret = ps.
```

SNPortSerieVitesse
V0_BAUD
V50_BAUD
V75_BAUD
V110_BAUD
V134_BAUD
V150_BAUD
V200_BAUD
V300_BAUD
V600_BAUD
V1200_BAUD
V1800_BAUD
V2400_BAUD
V4800_BAUD
V9600_BAUD
V19200_BAUD
V38400_BAUD
V57600_BAUD
V115200_BAUD
V230400_BAUD

SNPortSerieLongueurDonnees
D5_BITS
D6_BITS
D7_BITS
D8_BITS

SNPortSerieBitStop
S1_BIT
S2_BITS

SNPortSerieParite
PARITE_PAIRE
PARITE_IMPAIRE
PAS_DE_PARITE

```
// Paramétrage du nombre de bits : si echec, appeler Terminer() et renvoyer false.
```

```
// Paramétrage de la parité : si echec, appeler Terminer() et renvoyer false.
```

```
// Paramétrage du bit stop : si echec, appeler Terminer() et renvoyer false.
```

```
    return true;
}
```

 Écrire le code de la méthode *Initialiser()* dans le fichier WS2300.cpp.

Tester votre code en écrivant les instructions suivantes dans le programme principal :

1) Créer un objet ‘station’ de la classe WS2300

 2) Appeler la méthode *Initialiser()* avec l’objet ‘station’. Passer en argument le nom du port série (voir MS1 : potentiomètre du journal lumineux). Afficher la valeur de retour de la méthode *Initialiser()*.

3) Appeler la méthode *Terminer()* avec l’objet ‘station’.

La méthode *Initialiser()* - deuxième partie

L’initialisation se poursuit avec l’envoi d’un octet valant 0x06 à la station météorologique et jusqu’à la réception de l’octet 0x02 (voir le TD 2). Commençons par analyser les méthodes *EmettreUnOctet()* et *AttendreUnOctet()* de la classe WS2300.

La méthode *EmettreUnOctet()*

L'envoi d'un octet à la station météorologique n'est pas classique. Pour preuve, voici le code de la méthode *EmettreUnOctet()* :

```
bool WS2300::EmettreUnOctet(unsigned char octetAEEnvoyer)
{
    ps.ModifierRequestToSend(1);
    ps.ModifierDataTerminalReady(0);
    return ps.EnvoyerDesOctets(&octetAEEnvoyer, 1);
}
```

 Donner l'état des signaux RTS (Request To Send) et DTR (Data Terminal Ready) nécessaire pour l'envoi d'un octet.

 Donner le type des paramètres attendus par la méthode *EnvoyerDesOctets()* de la classe *SNPortSerie*.

 Justifier le ‘&’ devant *octetAEEnvoyer*.

 Ajouter la définition de la méthode *EmettreUnOctet()* de la classe *WS2300* dans le fichier *WS2300.cpp*.

La méthode *AttendreUnOctet()*

La réception d'un octet n'est pas classique non plus. La station météorologique étant ‘capricieuse’, il faut parfois attendre longtemps avant de recevoir une réponse de sa part. Parfois même elle ne répond pas. La méthode *AttendreUnOctet()* doit donc attendre la réponse de la station météorologique et s'arrêter d'attendre au bout d'un certain temps si aucun message n'arrive. Voici la définition en C++ de la méthode *AttendreUnOctet()* :

```
bool WS2300::AttendreUnOctet(unsigned char * octetALire)
{
    int nbBoucle=0;
    int nbOctetsLus=0;

    do
    {
        nbOctetsLus = ps.LireDesOctets(octetALire,1);
        usleep(4000);
        nbBoucle++;
    }while( (nbOctetsLus <= 0) && (nbBoucle <= 50) );

    return nbOctetsLus;
}
```

 Expliquer la condition de sortie de la boucle *do...while* du code précédent.

 Donner approximativement le temps d'exécution d'un tour de boucle sachant que la méthode *LireDesOctets()* de la classe *SNPortSerie* n'est pas bloquante.

 Donner approximativement le temps d'exécution maximum de la méthode *AttendreUnOctet()*.

 Ajouter la définition de la méthode *AttendreUnOctet()* de la classe WS2300 dans le fichier WS2300.cpp.

Voici le cahier des charges de la deuxième partie de la méthode *Initialiser()* :

1. Envoyer l'octet 0x06 (en utilisant la méthode *EmettreUnOctet()*)
2. Lire la réponse de la station (en utilisant la méthode *AttendreUnOctet()*)
3. Si il n'y a pas de réponse ou si la réponse n'est pas l'octet 0x02, alors on renvoie l'octet 0x06. Cependant, on limitera le nombre de tentative d'initialisation à 20. Si ce nombre est atteint, on abandonne l'initialisation et on renvoie la valeur faux.

 A partir de ce cahier des charges, écrire la deuxième partie de la définition de la méthode *Initialiser()* :

```
int nbTentatives = 0;  
unsigned char reponse;
```

 Ajouter ce code à la fin de la méthode *Initialiser()* dans le fichier WS2300.cpp. Tester les modifications que vous avez apportées à la méthode *Initialiser()* en relançant votre programme. Vérifier que l'initialisation de la station se fait correctement.

TD 3 - Lire des données sur la station météorologique

La memory map de la station météorologique (le plan mémoire)

Les données météorologiques de la station WS2300 sont stockées en mémoire. Le plan de la mémoire est donnée en annexe. Les données sont stockées par quartets (1 demi octet, soit 4 bits). Les adresses correspondent à des adresses de quartet.

? En utilisant le memory_map (plan de la mémoire) du WS2300, compléter le tableau suivant :

La donnée à lire	L'adresse en mémoire	Le nombre d'octet(s) à lire
La direction du vent	0x52C	1
La température intérieure		
La vitesse du vent		
L'humidité intérieure		
La pression	0X5E2	
L'heure		
La date		

Le protocole de lecture des données météorologiques

Nous allons illustrer la lecture des données météorologiques avec la direction du vent. Le principe reste strictement identique pour toutes les autres données.

Préparation du message à envoyer

Le message à envoyer à la station météorologique est de 5 octets :

- Les 4 premiers octets contiennent chacun un quartet de l'adresse demandée multiplié par 4 puis additionné de 0x82.
- Le dernier octet contient le nombre d'octet(s) à lire multiplié par 4 puis additionné de 0xC2

Voilà le détail du calcul des 5 octets à envoyer pour la direction du vent :

$$\text{octetsAEnvoyer}[0] = 0 * 4 + 0x82 = 0x82$$

$$\text{octetsAEnvoyer}[1] = 5 * 4 + 0x82 = 0x96$$

$$\text{octetsAEnvoyer}[2] = 2 * 4 + 0x82 = 0x8A$$

$$\text{octetsAEnvoyer}[3] = C * 4 + 0x82 = 0xB2$$

$$\text{octetsAEnvoyer}[4] = 1 * 4 + 0xC2 = 0xC6$$

? Donner les 5 octets à envoyer pour demander la température à la station météorologique :

octetsAEnvoyer[0] =

octetsAEnvoyer[1] =

octetsAEnvoyer[2] =

octetsAEnvoyer[3] =

octetsAEnvoyer[4] =

?

Écrire le code en C++ permettant de générer automatiquement ces 5 octets en fonction de l'adresse et du nombre d'octet demandé :

```
bool WS2300::LireDesOctets(unsigned short adresseALire, int nbOctetsALire, unsigned char * donneesLues)
{
    // Préparation du tableau d'octets à envoyer
    unsigned char octetsAEnvoyer[5];

    // A continuer
}
```

Envoi du message et réception

Il faut maintenant envoyer le message à la station météorologique. A chaque octet émis, la station météorologique répond par un octet. Si elle ne répond pas, c'est qu'il y a une erreur lors de la communication, il faut alors quitter la méthode avec la valeur false.

?

Écrire en C++ l'envoi des 5 octets comme décrit ci-dessus :

```
bool WS2300::LireDesOctets(unsigned short adresseALire, int nbOctetsALire, unsigned char * donneesLues)
{
    // Préparation du tableau d'octets à envoyer : fait lors d'une question précédente.
    // Envoi du message contenu dans octetsAEnvoyer

    // A continuer
}
```

Réception des octets demandés

Une fois les 5 octets envoyés, la station météorologique renvoie les octets demandés plus 1 dernier servant de checksum. Si la station cesse de répondre, la méthode renverra la valeur false.

? Écrire en C++ la lecture de octets demandés dans donneesLues :

```
bool WS2300::LireDesOctets(unsigned short adresseALire, int nbOctetsALire, unsigned char * donneesLues)
{
    // Préparation du tableau d'octets à envoyer : fait lors d'une question précédente.
    // Envoi du message contenu dans octetsAEnvoyer : fait lors d'une question précédente.
    // Lecture des données

    // Lecture du checksum

    return true;
}
```

Interprétation des données reçues

On se place toujours dans le cas où l'on a demandé la direction du vent à la station météorologique.

La station météorologique nous a envoyé 1 octet correspondant aux quartets 0x52C et 0x52D.

Supposons que nous avons reçue la valeur suivante dans donneesLues[0] => 0xD2 :

- Le quartet 'D' est la valeur de l'adresse 0x52D
- Le quartet '2' est la valeur de l'adresse 0x52C

L'information qui nous intéresse se trouve uniquement dans le quartet 0x52C, soit le '2'.

? Écrire en C++ l'extraction du quartet de l'adresse 0x52C, puis afficher sa valeur dans le terminal.

? D'après la Memory MAP, la données lues doit être multipliée par 22,5 pour obtenir un angle en degré. Effectuer le calcul donnant la valeur de la direction du vent et stocker ce résultat dans un entier. Afficher la valeur.

Défi 3 - Extraction des données météorologiques

Objectif

L'objectif de cette partie est d'extraire le maximum d'informations disponibles sur la station météorologique, comme par exemple la température intérieure, la direction du vent, la vitesse du vent, la pression, l'humidité, la date et l'heure.

Définition de la méthode *LireDesOctets()*

La définition de la méthode *LireDesOctets()* a été travaillée dans le TD précédent.

 En utilisant le TD précédent, ajouter à la classe WS2300 la définition de la méthode *LireDesOctets()*.

Définition de la méthode *LireDirectionDuVent()*

D'après le tableau du TD 3, la direction du vent est contenue dans le quartet se trouvant à l'adresse 0x52C. Comme il n'est pas possible de demander 1 seul quartet à la station météorologique mais au moins 1 octet, la station météorologique nous renverra 1 octet décomposé comme suit :

	Quartet de poids fort	Quartet de poids faible
octetLus[0]	0x52D	0x52C

D'après la Memory Map de la station WS2300, pour obtenir la valeur de la direction du vent, il faut multiplier la valeur lue à l'adresse 0x52C par 22,5.

La définition de la méthode *LireDirectionDuVent()* est donnée avec l'algorithme suivant :

```
METHODE LireDirectionDuVent DE LA CLASSE WS2300
ARGUMENT :
VARIABLE RETOURNÉE : direction : float
DEBUT
    b : booleen
    octetsLus[1] : char
    direction ← 999.99
    b ← LireDesOctets(0x52C, 1, octetsLus)
    SI b vaut VRAI ALORS
        direction ← (octetsLus[0] & 0x0F) * 22,5
    FIN SI
FIN
```

 D'après cet algorithme, donner la valeur de 'direction' si la méthode *LireDesOctets()* renvoie *false*.

direction =

 Justifier l'opération binaire '& 0x0F' dans le calcul de la direction du vent.

 Interpréter l'algorithme de la méthode *LireDirectionDuVent()* en C++

```
float WS2300::LireDirectionDuVent()
{
    float direction;
```

```
    return direction;
}
```

- Ajouter la définition de la méthode *LireDirectionDuVent()* dans votre fichier WS2300.cpp
- Appeler cette méthode dans le programme principal et afficher le résultat qu'elle renvoie.

Définition de la méthode *LireTemperatureInterieure()*

D'après le tableau du TD3, la température intérieure se lit à partir de l'adresse 0x346 sur 2 octets.

La réponse de la station météorologique (stockée dans le tableau *octetsLus*) est la suivante :

	Quartet de poids fort	Quartet de poids faible
octetsLus[0]	0x347	0x346
octetsLus[1]	0x349	0x348

D'après la Memory Map, les quartets précédents contiennent les valeurs suivantes :

0x349 : la valeur des dizaines moins 3 de la température intérieure.

0x348 : la valeur des unités de la température intérieure.

0x347 : la valeur des dixièmes de la température intérieure.

0x346 : la valeur des centièmes de la température intérieure.

- Supposons que la valeur reçue dans *octetsLus[0]* soit de 0x70 et celle de *octetsLus[1]* soit de 0x51. Justifier par un calcul que la valeur de la température est de 21,70°C.

Valeur de 0x349 :

Valeur de 0x348 :

Valeur de 0x347 :

Valeur de 0x346 :

température =



Compléter l'algorithme de la méthode *LireTemperatureInterieure()*

```
METHODE TemperatureInterieure DE LA CLASSE WS2300
ARGUMENT :
VARIABLE RETOURNÉE : temperature : float
DEBUT
    b : booleen
    octetsLus[ ] : char
    temperature ← -400.00
    b ← LireDesOctets(      ,      , octetsLus)
    SI b vaut VRAI ALORS
        temperature ←

    FIN SI
FIN
```

- Traduire en C++ l'algorithme précédent et ajouter ce code dans le fichier WS2300.cpp
- Appeler cette méthode dans le programme principal et afficher le résultat qu'elle renvoie.

Les autres données météorologiques

Les méthodes `LirePression()`, `LireHumiditeInterieure()` et `LireVitesseDuVent()`

-  Sur le même principe que les méthodes précédentes, écrire les définitions des méthodes `LirePression()` et `LireHumiditeInterieure()` et `LireVitesseDuVent()`.
-  Tester toutes ces méthodes dans votre programme principal.

Les méthodes `LireDate()` et `LireHeure()`

Nous allons ajouter à notre classe WS2300 les méthodes `LireDate()` et `LireHeure()` qui renverront la date et l'heure de la station météorologique. Les données de date et d'heure seront stockées dans des structures.

La structure de l'heure (WS2300Heure) contiendra 3 attributs :

1. heure,
2. minute,
3. seconde.

La structure de la date (WS2300Date) contiendra 3 attributs :

1. jour,
2. mois,
3. annee.

?

Écrire en C++ la déclaration de ces 2 structures en utilisant le type approprié pour chacun des attributs.

💻 Ajouter les déclarations de ces structures dans le fichier WS2300.h

Les prototypes des méthodes LireDate() et LireHeure() sont les suivantes :

```
WS2300Date LireDate();  
WS2300Heure LireHeure();
```

💻 Écrire en C++ la déclaration et la définition de ces méthodes dans la classe WS2300.

💻 Tester ces méthodes en les appelant dans le programme principal et en affichant les valeurs qu'elles renvoient.

Défi 4 - Application de surveillance météorologique

Objectif

L'objectif de cette partie est de mettre en place une application de surveillance d'une station météorologique. Cette application sera distante, accessible depuis n'importe quel PC du réseau local.

Cette application affichera les informations météorologiques précédemment lues, puis permettra à l'utilisateur de mettre en place des alertes pour être averti lorsqu'un seuil est dépassé.

Le serveur TCP sur la Raspberry

Sur la Raspberry, vos enseignants lanceront un serveur TCP mono client. Ce serveur attend une requête sur 3 caractères, envoie la réponse puis déconnecte automatiquement le client connecté. Le tableau suivant fait la liste des requêtes gérées par le serveur et son format de réponse :

Numéro	Signification	Les requêtes	Les réponses
1	Température INTérieure	TIN	TIN=18.9
2	Direction du VenT	DVT	DVT=45.0
3	Vitesse du VenT	VVT	VVT=20.0
4	Humidité INTérieure	HIN	HIN=49
5	PrESSioN atmosphérique	PSN	PSN=1024
6	DATE	DAT	DAT=24/1/20
7	HeuRE	HRE	HRE=11:25

Première version de l'application : affichage de la température

Dans la première version, l'application graphique affichera la température. Elle contiendra :

- 2 TEdit pour que l'utilisateur saisisse l'adresse IP (*EditAdresseIP*) et le port (*EditPort*) du serveur TCP de la station météo.
- 1 Tbutton (*ButtonTemperature*) pour faire lancer la demande de température au serveur.
- 1 TEdit (*EditTemperature*) pour afficher le message reçu.
- 1 TClientSocket (*ClientSocket1*) pour communiquer avec la station météorologique.

 Gérer l'évènement *OnClick* sur le *ButtonTemperature* pour lancer la connexion au serveur.

 Insérer le code suivant :

```
ClientSocket1->Host = EditAdresseIP->Text;
ClientSocket1->Port = EditPort->Text.ToInt();
ClientSocket1->Active = true;
```

 Expliquer chacune des 3 instructions précédentes :

Si l'instruction précédente fonctionne, une connexion s'établit entre le client et le serveur. L'événement *OnConnect* du *ClientSocket1* est alors déclenché.

 Gérer l'événement *OnConnect* du *ClientSocket1* pour envoyer la requête au serveur.

 Insérer le code suivant :

```
ClientSocket1->Socket->SendText("TIN");
```

 Expliquer le rôle de l'instruction précédente.

Il ne reste plus qu'à recevoir la réponse du serveur. C'est l'événement *OnRead* de *ClientSocket1* qui indiquera quand une réponse a été reçue.

 Gérer l'événement *OnRead* du *ClientSocket1*.

 Insérer dans cet événement l'instruction suivante :

```
EditTemperature->Text = ClientSocket1->Socket->ReceiveText();
```

 Expliquer l'instruction précédente.

 Tester et vérifier le fonctionnement de votre programme.

Deuxième version de l'application : afficher toutes les informations météorologiques disponibles

Dans cette 2ème version, on souhaite une application qui puisse afficher toutes les informations météorologiques disponibles. Il suffira de cliquer sur le bouton correspondant à la donnée souhaitée pour qu'elle soit mise à jour.

 Ajouter autant de *TButton* et de *TEdit* qu'il y a de valeur à lire sur la station météorologique. Leur donner des noms qui ont un sens (pour la pression : un *ButtonPress* et un *EditPress* par exemple).

 Gérer l'événement *OnClick* sur chacun de ces *TButton*. Pour chacun, lancer la connexion au serveur et mettre dans l'attribut privé 'numero' (que vous déclarer dans la classe *TForm1*) la valeur correspondant à la colonne 'Numéro' du tableau précédent. Par exemple, si le bouton concerne la demande de température, mettre dans 'numero' la valeur 1, s'il s'agit de la pression, mettre la valeur 5, etc.

 Dans l'événement *OnConnect* du *ClientSocket1*, envoyer au serveur la requête qui correspond au 'numero'. Par exemple, si 'numero' vaut 1, envoyer "TIN", si 'numero' vaut 5, envoyer "PSN", etc.

 Enfin, dans l'événement *OnRead* du *ClientSocket1*, afficher le message reçu dans le bon *TEdit*. Par exemple, si 'numero' vaut 1, le message reçu s'affichera dans *EditTemperature*, si 'numero' vaut 5, il s'affichera dans *EditPress*, etc.

Troisième version : mise à jour automatique des valeurs avec des Timer

Dans cette troisième version, on fera une mise à jour automatique des données météorologiques en suivant le tableau suivant :

Numéro	Signification	Mise à jour toutes les	Pour les tests, MAJ toutes les
1	Température INTérieure	5 minutes	20 secondes
2	Direction du VenT	1 minute	5 secondes
3	Vitesse du VenT	1 minute	7 secondes
4	Humidité INTérieure	15 minutes	40 secondes
5	PrESSioN atmosphérique	10 minutes	30 secondes
6	DATE	1 heure	40 secondes
7	HeuRE	1 minute	9 secondes

Normalement, on devrait suivre la 3ème colonne, mais les temps d'attente étant trop long, on utilisera les informations de la

4ème colonne.

- Pour réaliser ces répétitions périodiques, utiliser les *TTimer*, paramétrer la durée de répétition et gérer l'événement *OnTimer*.

Quatrième version : ajout d'un fichier log

Dans cette version, on souhaite disposer d'un fichier log qui enregistrera toutes les demandes et les réponses de la station météorologiques. Chaque enregistrement se fera sur un seule ligne et sera daté (jour et heure).

En réutilisant les modules précédents, répondez aux questions suivantes :

?

Donner le nom de la classe permettant d'écrire dans un fichier.

?

Donner la liste de toutes les méthodes que vous avez précédemment utilisées et expliquer le rôle de chacune.

Cinquième version : afficher un message d'alerte en cas de seuil dépassé

L'objectif de cette partie est de permettre à l'utilisateur d'ajouter des messages d'alertes en cas de franchissement de seuil. Par exemple, si la température dépasse les 20°C, un message surgit avec un descriptif de l'alerte.

- Ajouter les éléments graphiques nécessaire pour que l'utilisateur puisse saisir une alerte.
- Si le seuil est dépassé, afficher un message dans une boîte de dialogue spécifique.

Annexe - Memory MAP FOR WS 2300

Address	Data_Sample	Function	
0000 4			0268 0
0001 F			0269 0
0002 F			026A 0
0003 B			TENDENCY/FORECAST
0004 0			026B 0 Forecast: 0=rainy, 1=cloudy, 2=sunny
0005 0			026C 2 Tendency: 0=steady, 1=rising, 2=falling
0006 2	Bitsettings:	bit3=buzzer off	026D 0
0007 0			026E 0
0008 0			026F 0
0009 0			INDOOR TEMPERATURE
000A 0			0346 0 Current Indoor Temperature: BCD offset 30 0.01s [C]
000B 0			0347 6 Current Indoor Temperature: BCD offset 30 0.1s [C]
000C 0			0348 8 Current Indoor Temperature: BCD offset 30 1s [C]
000D 1			0349 5 Current Indoor Temperature: BCD offset 30 10s [C]
000E 6			034A 0
000F 0	Wind unit: 0=m/s, 1=knots, 2=beaufort, 3=km/h, 4=mph		Minimum Indoor Temperature: BCD offset 30 0.01s [C]
0010 2			034B 0 Minimum Indoor Temperature: BCD offset 30 0.1s [C]
0011 0			034C 5 Minimum Indoor Temperature: BCD offset 30 1s [C]
0012 0			034D 0 Minimum Indoor Temperature: BCD offset 30 10s [C]
0013 0			034E 4 Minimum Indoor Temperature: BCD offset 30 10s [C]
0014 4			034F 0
0015 0			0350 0 Maximum Indoor Temperature: BCD offset 30 0.1s [C]
0016 7	Bitsettings:	bit0=backlight	0351 7 Maximum Indoor Temperature: BCD offset 30 0.1s [C]
0017 0			0352 9 Maximum Indoor Temperature: BCD offset 30 1s [C]
0018 0			0353 5 Maximum Indoor Temperature: BCD offset 30 10s [C]
			0354 0 Time min Indoor Temperature:, minutes BCD 1s
			0355 2 Time min Indoor Temperature:, minutes BCD 10s
			0356 2 Time min Indoor Temperature:, hours BCD 1s
			0357 1 Time min Indoor Temperature:, hours BCD 10s
			0358 1 Date min Indoor Temperature:, BCD day 1s
			0359 0 Date min Indoor Temperature:, BCD day 10s
			035A 1 Date min Indoor Temperature:, BCD month 1s
			035B 0 Date min Indoor Temperature:, BCD month 10s
			035C 1 Date min Indoor Temperature:, BCD year 1s
			035D 0 Date min Indoor Temperature:, BCD year 10s
			035E 5 Time max Indoor Temperature:, minutes BCD 1s
			035F 5 Time max Indoor Temperature:, minutes BCD 10s
			0360 7 Time max Indoor Temperature:, hours BCD 1s
			0361 1 Time max Indoor Temperature:, hours BCD 10s
			0362 1 Date max Indoor Temperature:, BCD day 1s
			0363 2 Date max Indoor Temperature:, BCD day 10s
			0364 4 Date max Indoor Temperature:, BCD month 1s
			0365 0 Date max Indoor Temperature:, BCD month 10s
			0366 3 Date max Indoor Temperature:, BCD year 1s
			0367 0 Date max Indoor Temperature:, BCD year 10s
			0369 0 Low Alarm Indoor Temperature: BCD offset 30 0.01s [C]
			036A 1 Low Alarm Indoor Temperature: BCD offset 30 0.1s [C]
			036B 7 Low Alarm Indoor Temperature: BCD offset 30 1s [C]
			036C 4 Low Alarm Indoor Temperature: BCD offset 30 10s [C]
			036D 0
			036E 0 High Alarm Indoor Temperature: BCD offset 30 0.01s [C]
			036F 6 High Alarm Indoor Temperature: BCD offset 30 0.1s [C]
			0370 9 High Alarm Indoor Temperature: BCD offset 30 1s [C]
			0371 5 High Alarm Indoor Temperature: BCD offset 30 10s [C]
			0372 0
			OUTDOOR TEMPERATURE
			0373 0 Current Outdoor Temperature: BCD offset 30 0.01s [C]
			0374 4 Current Outdoor Temperature: BCD offset 30 0.1s [C]
			0375 7 Current Outdoor Temperature: BCD offset 30 1s [C]
			0376 3 Current Outdoor Temperature: BCD offset 30 10s [C]
			0377 0
			0378 0 Minimum Outdoor Temperature: BCD offset 30 0.01s [C]
			0379 5 Minimum Outdoor Temperature: BCD offset 30 0.1s [C]
			037A 2 Minimum Outdoor Temperature: BCD offset 30 1s [C]
			037B 3 Minimum Outdoor Temperature: BCD offset 30 10s [C]
			037C 0
			037D 0 Maximum Outdoor Temperature: BCD offset 30 0.01s [C]
			037E 8 Maximum Outdoor Temperature: BCD offset 30 0.1s [C]
			037F 5 Maximum Outdoor Temperature: BCD offset 30 1s [C]
			0380 6 Maximum Outdoor Temperature: BCD offset 30 10s [C]
			0381 5 Time min Outdoor Temperature:, minutes BCD 1s
			0382 2 Time min Outdoor Temperature:, minutes BCD 10s
			0383 6 Time min Outdoor Temperature:, hours BCD 1s
			0384 0 Time min Outdoor Temperature:, hours BCD 10s
			0385 7 Date min Outdoor Temperature:, BCD day 1s
			0386 1 Date min Outdoor Temperature:, BCD day 10s
			0387 4 Date min Outdoor Temperature:, BCD month 1s
			0388 0 Date min Outdoor Temperature:, BCD month 10s
			0389 3 Date min Outdoor Temperature:, BCD year 1s
			038A 0 Date min Outdoor Temperature:, BCD year 10s
			038B 1 Time max Outdoor Temperature:, minutes BCD 1s
			038C 0 Time max Outdoor Temperature:, minutes BCD 10s
			038D 7 Time max Outdoor Temperature:, hours BCD 1s
			038E 1 Time max Outdoor Temperature:, hours BCD 10s
			038F 7 Date max Outdoor Temperature:, BCD day 1s
			0390 9 Date max Outdoor Temperature:, BCD day 10s
			0391 4 Date max Outdoor Temperature:, BCD month 1s
			0392 0 Date max Outdoor Temperature:, BCD month 10s
			0393 3 Date max Outdoor Temperature:, BCD year 1s
			0394 0 Date max Outdoor Temperature:, BCD year 10s
			0395 0
			0396 0 Low Alarm Outdoor Temperature: BCD offset 30 0.01s [C]
			0397 0 Low Alarm Outdoor Temperature: BCD offset 30 0.1s [C]
			0398 0 Low Alarm Outdoor Temperature: BCD offset 30 1s [C]
			0399 3 Low Alarm Outdoor Temperature: BCD offset 30 10s [C]
			039A 0
			039B 0 High Alarm Outdoor Temperature: BCD offset 30 0.01s [C]
0200 0	TIME		
0201 1	Current time: Seconds BCD 1s		
0202 6	Current time: Seconds BCD 10s		
0203 2	Current time: Minutes BCD 1s		
0204 3	Current time: Minutes BCD 10s		
0205 0	Current time: Hours BCD 1s		
0206 4	Current time: Hours BCD 10s		
0207 4			
			0372 0
			OUTDOOR TEMPERATURE
			0373 0 Current Outdoor Temperature: BCD offset 30 0.01s [C]
			0374 4 Current Outdoor Temperature: BCD offset 30 0.1s [C]
			0375 7 Current Outdoor Temperature: BCD offset 30 1s [C]
			0376 3 Current Outdoor Temperature: BCD offset 30 10s [C]
			0377 0
			0378 0 Minimum Outdoor Temperature: BCD offset 30 0.01s [C]
			0379 5 Minimum Outdoor Temperature: BCD offset 30 0.1s [C]
			037A 2 Minimum Outdoor Temperature: BCD offset 30 1s [C]
			037B 3 Minimum Outdoor Temperature: BCD offset 30 10s [C]
			037C 0
			037D 0 Maximum Outdoor Temperature: BCD offset 30 0.01s [C]
			037E 8 Maximum Outdoor Temperature: BCD offset 30 0.1s [C]
			037F 5 Maximum Outdoor Temperature: BCD offset 30 1s [C]
			0380 6 Maximum Outdoor Temperature: BCD offset 30 10s [C]
			0381 5 Time min Outdoor Temperature:, minutes BCD 1s
			0382 2 Time min Outdoor Temperature:, minutes BCD 10s
			0383 6 Time min Outdoor Temperature:, hours BCD 1s
			0384 0 Time min Outdoor Temperature:, hours BCD 10s
			0385 7 Date min Outdoor Temperature:, BCD day 1s
			0386 1 Date min Outdoor Temperature:, BCD day 10s
			0387 4 Date min Outdoor Temperature:, BCD month 1s
			0388 0 Date min Outdoor Temperature:, BCD month 10s
			0389 3 Date min Outdoor Temperature:, BCD year 1s
			038A 0 Date min Outdoor Temperature:, BCD year 10s
			038B 1 Time max Outdoor Temperature:, minutes BCD 1s
			038C 0 Time max Outdoor Temperature:, minutes BCD 10s
			038D 7 Time max Outdoor Temperature:, hours BCD 1s
			038E 1 Time max Outdoor Temperature:, hours BCD 10s
			038F 7 Date max Outdoor Temperature:, BCD day 1s
			0390 9 Date max Outdoor Temperature:, BCD day 10s
			0391 4 Date max Outdoor Temperature:, BCD month 1s
			0392 0 Date max Outdoor Temperature:, BCD month 10s
			0393 3 Date max Outdoor Temperature:, BCD year 1s
			0394 0 Date max Outdoor Temperature:, BCD year 10s
			0395 0
			0396 0 Low Alarm Outdoor Temperature: BCD offset 30 0.01s [C]
			0397 0 Low Alarm Outdoor Temperature: BCD offset 30 0.1s [C]
			0398 0 Low Alarm Outdoor Temperature: BCD offset 30 1s [C]
			0399 3 Low Alarm Outdoor Temperature: BCD offset 30 10s [C]
			039A 0
			039B 0 High Alarm Outdoor Temperature: BCD offset 30 0.01s [C]
0240 1	DATE UNIT SET		
0241 2	Date the unit was last set to: BCD 1s		
0242 5	Date the unit was last set to: BCD 10s		
0243 0	Month the unit was last set to: BCD 1s		
0244 0	Month the unit was last set to: BCD 10s		
0245 0	Year the unit was last set to: BCD 1s		
0246 0	Year the unit was last set to: BCD 10s		
0247 0			
0248 7			
0249 0			
024A 8			
024B 1			
024C 1			
			0372 0
			OUTDOOR TEMPERATURE
			0373 0 Current Outdoor Temperature: BCD offset 30 0.01s [C]
			0374 4 Current Outdoor Temperature: BCD offset 30 0.1s [C]
			0375 7 Current Outdoor Temperature: BCD offset 30 1s [C]
			0376 3 Current Outdoor Temperature: BCD offset 30 10s [C]
			0377 0
			0378 0 Minimum Outdoor Temperature: BCD offset 30 0.01s [C]
			0379 5 Minimum Outdoor Temperature: BCD offset 30 0.1s [C]
			037A 2 Minimum Outdoor Temperature: BCD offset 30 1s [C]
			037B 3 Minimum Outdoor Temperature: BCD offset 30 10s [C]
			037C 0
			037D 0 Maximum Outdoor Temperature: BCD offset 30 0.01s [C]
			037E 8 Maximum Outdoor Temperature: BCD offset 30 0.1s [C]
			037F 5 Maximum Outdoor Temperature: BCD offset 30 1s [C]
			0380 6 Maximum Outdoor Temperature: BCD offset 30 10s [C]
			0381 5 Time min Outdoor Temperature:, minutes BCD 1s
			0382 2 Time min Outdoor Temperature:, minutes BCD 10s
			0383 6 Time min Outdoor Temperature:, hours BCD 1s
			0384 0 Time min Outdoor Temperature:, hours BCD 10s
			0385 7 Date min Outdoor Temperature:, BCD day 1s
			0386 1 Date min Outdoor Temperature:, BCD day 10s
			0387 4 Date min Outdoor Temperature:, BCD month 1s
			0388 0 Date min Outdoor Temperature:, BCD month 10s
			0389 3 Date min Outdoor Temperature:, BCD year 1s
			038A 0 Date min Outdoor Temperature:, BCD year 10s
			038B 1 Time max Outdoor Temperature:, minutes BCD 1s
			038C 0 Time max Outdoor Temperature:, minutes BCD 10s
			038D 7 Time max Outdoor Temperature:, hours BCD 1s
			038E 1 Time max Outdoor Temperature:, hours BCD 10s
			038F 7 Date max Outdoor Temperature:, BCD day 1s
			0390 9 Date max Outdoor Temperature:, BCD day 10s
			0391 4 Date max Outdoor Temperature:, BCD month 1s
			0392 0 Date max Outdoor Temperature:, BCD month 10s
			0393 3 Date max Outdoor Temperature:, BCD year 1s
			0394 0 Date max Outdoor Temperature:, BCD year 10s
			0395 0
			0396 0 Low Alarm Outdoor Temperature: BCD offset 30 0.01s [C]
			0397 0 Low Alarm Outdoor Temperature: BCD offset 30 0.1s [C]
			0398 0 Low Alarm Outdoor Temperature: BCD offset 30 1s [C]
			0399 3 Low Alarm Outdoor Temperature: BCD offset 30 10s [C]
			039A 0
			039B 0 High Alarm Outdoor Temperature: BCD offset 30 0.01s [C]
0250 0	DATE UNIT SET		
0251 3	Month the unit was last set to: BCD 1s		
0252 0	Month the unit was last set to: BCD 10s		
0253 0			
0254 0			
0264 3			
0265 5			
0266 4	LCD contrast value -1. Changing it has no impact on LCD.		
Read only			
0267 1			

039C 0	High Alarm Outdoor Temperature: BCD offset 30 0.1s [C]	0401 8	Time min Rel Humidity Indoors:, minutes BCD 1s
039D 0	High Alarm Outdoor Temperature: BCD offset 30 1s [C]	0402 1	Time min Rel Humidity Indoors:, minutes BCD 10s
039E 7	High Alarm Outdoor Temperature: BCD offset 30 10s [C]	0403 0	Time min Rel Humidity Indoors:, hours BCD 1s
039F 0		0404 0	Time min Rel Humidity Indoors:, hours BCD 10s
WINDCHILL		0405 1	Date min Rel Humidity Indoors:, BCD day 1s
03A0 0	Current Windchill: BCD offset 30 0.01s [C]	0406 2	Date min Rel Humidity Indoors:, BCD day 10s
03A1 4	Current Windchill: BCD offset 30 0.1s [C]	0407 4	Date min Rel Humidity Indoors:, BCD month 1s
03A2 7	Current Windchill: BCD offset 30 1s [C]	0408 0	Date min Rel Humidity Indoors:, BCD month 10s
03A3 3	Current Windchill: BCD offset 30 10s [C]	0409 3	Date min Rel Humidity Indoors:, BCD year 1s
03A4 8		040A 0	Date min Rel Humidity Indoors:, BCD year 10s
03A5 0	Minimum Windchill: BCD offset 30 0.01s [C]	040B 5	Time max Rel Humidity Indoors:, minutes BCD 1s
03A6 2	Minimum Windchill: BCD offset 30 0.1s [C]	040C 4	Time max Rel Humidity Indoors:, minutes BCD 10s
03A7 8	Minimum Windchill: BCD offset 30 1s [C]	040D 3	Time max Rel Humidity Indoors:, hours BCD 1s
03A8 1	Minimum Windchill: BCD offset 30 10s [C]	040E 2	Time max Rel Humidity Indoors:, hours BCD 10s
03A9 0		040F 4	Date max Rel Humidity Indoors:, BCD day 1s
03AA 0	Maximum Windchill: BCD offset 30 0.01s [C]	0410 1	Date max Rel Humidity Indoors:, BCD day 10s
03AB 8	Maximum Windchill: BCD offset 30 0.1s [C]	0411 4	Date max Rel Humidity Indoors:, BCD month 1s
03AC 9	Maximum Windchill: BCD offset 30 1s [C]	0412 0	Date max Rel Humidity Indoors:, BCD month 10s
03AD 4	Maximum Windchill: BCD offset 30 10s [C]	0413 3	Date max Rel Humidity Indoors:, BCD year 1s
03AE 2	Time min Windchill:, minutes BCD 1s	0414 0	Date max Rel Humidity Indoors:, BCD year 10s
03AF 0	Time min Windchill:, minutes BCD 10s	0415 5	Low Alarm Rel Humidity Indoors: BCD 1s [%]
03B0 6	Time min Windchill:, hours BCD 1s	0416 3	Low Alarm Rel Humidity Indoors: BCD 10s [%]
03B1 0	Time min Windchill:, hours BCD 10s	0417 5	High Alarm Rel Humidity Indoors: BCD 1s [%]
03B2 9	Date min Windchill:, BCD day 1s	0418 6	High Alarm Rel Humidity Indoors: BCD 10s [%]
03B3 1	Date min Windchill:, BCD day 10s	HUMIDITY OUTDOORS	
03B4 4	Date min Windchill:, BCD month 1s	0419 1	Rel Humidity Outdoors: BCD 1s [%]
03B5 0	Date min Windchill:, BCD month 10s	041A 7	Rel Humidity Outdoors: BCD 10s [%]
03B6 3	Date min Windchill:, BCD year 1s	041B 9	Minimum Rel Humidity Outdoors: BCD 1s [%]
03B7 0	Date min Windchill:, BCD year 10s	041C 1	Minimum Rel Humidity Outdoors: BCD 10s [%]
03B8 6	Time max Windchill:, minutes BCD 1s	041D 8	Maximum Rel Humidity Outdoors: BCD 1s [%]
03B9 4	Time max Windchill:, minutes BCD 10s	041E 7	Maximum Rel Humidity Outdoors: BCD 10s [%]
03BA 8	Time max Windchill:, hours BCD 1s	041F 9	Time min Rel Humidity Outdoors:, minutes BCD 1s
03BB 0	Time max Windchill:, hours BCD 10s	0420 1	Time min Rel Humidity Outdoors:, minutes BCD 10s
03BC 3	Date max Windchill:, BCD day 1s	0421 7	Time min Rel Humidity Outdoors:, hours BCD 1s
03BD 2	Date max Windchill:, BCD day 10s	0422 1	Time min Rel Humidity Outdoors:, hours BCD 10s
03BE 4	Date max Windchill:, BCD month 1s	0423 1	Date min Rel Humidity Outdoors:, BCD day 1s
03BF 0	Date max Windchill:, BCD month 10s	0424 2	Date min Rel Humidity Outdoors:, BCD day 10s
03C0 3	Date max Windchill:, BCD year 1s	0425 4	Date min Rel Humidity Outdoors:, BCD month 1s
03C1 0	Date max Windchill:, BCD year 10s	0426 0	Date min Rel Humidity Outdoors:, BCD month 10s
03C2 0		0427 3	Date min Rel Humidity Outdoors:, BCD year 1s
03C3 0	Low Alarm Windchill: BCD offset 30 0.01s [C]	0428 0	Date min Rel Humidity Outdoors:, BCD year 10s
03C4 0	Low Alarm Windchill: BCD offset 30 0.1s [C]	0429 6	Time max Rel Humidity Outdoors:, minutes BCD 1s
03C5 5	Low Alarm Windchill: BCD offset 30 1s [C]	042A 2	Time max Rel Humidity Outdoors:, minutes BCD 10s
03C6 3	Low Alarm Windchill: BCD offset 30 10s [C]	042B 7	Time max Rel Humidity Outdoors:, hours BCD 1s
03C7 0		042C 0	Time max Rel Humidity Outdoors:, hours BCD 10s
03C8 0	High Alarm Windchill: BCD offset 30 0.01s [C]	042D 9	Date max Rel Humidity Outdoors:, BCD day 1s
03C9 0	High Alarm Windchill: BCD offset 30 0.1s [C]	042E 1	Date max Rel Humidity Outdoors:, BCD day 10s
03CA 0	High Alarm Windchill: BCD offset 30 1s [C]	042F 4	Date max Rel Humidity Outdoors:, BCD month 1s
03CB 6	High Alarm Windchill: BCD offset 30 10s [C]	0430 0	Date max Rel Humidity Outdoors:, BCD month 10s
03CC 0		0431 3	Date max Rel Humidity Outdoors:, BCD year 1s
03CD 0		0432 0	Date max Rel Humidity Outdoors:, BCD year 10s
DEWPPOINT		0433 5	Low Alarm Rel Humidity Outdoors: BCD 1s [%]
03CE 9	Current Dewpoint: BCD offset 30 0.01s [C]	0434 4	Low Alarm Rel Humidity Outdoors: BCD 10s [%]
03CF 4	Current Dewpoint: BCD offset 30 0.1s [C]	0435 0	High Alarm Rel Humidity Outdoors: BCD 1s [%]
03D0 2	Current Dewpoint: BCD offset 30 1s [C]	0436 7	High Alarm Rel Humidity Outdoors: BCD 10s [%]
03D1 3	Current Dewpoint: BCD offset 30 10s [C]		
03D2 6		0439 2	
03D3 1	Minimum Dewpoint: BCD offset 30 0.01s [C]	043A C	Next many addresses could be related to rain counting
03D4 1	Minimum Dewpoint: BCD offset 30 0.1s [C]	043B F	it seems that this is the same numbers that are used
03D5 3	Minimum Dewpoint: BCD offset 30 1s [C]	043C 0	in the history data for rain. It must be the area the
03D6 2	Minimum Dewpoint: BCD offset 30 10s [C]	043D C	station keeps track of rain fall by minutes and hours
03D7 0		043E F	in order to be able to give figures for rainfall in
03D8 4	Maximum Dewpoint: BCD offset 30 0.01s [C]	043F 0	the last hour and the last 24 hours.
03D9 4	Maximum Dewpoint: BCD offset 30 0.1s [C]	0440 C	
03DA 8	Maximum Dewpoint: BCD offset 30 1s [C]	0441 F	
03DB 3	Maximum Dewpoint: BCD offset 30 10s [C]	0442 0	
03DC 7	Time min Dewpoint:, minutes BCD 1s	0443 C	
03DD 1	Time min Dewpoint:, minutes BCD 10s	0444 F	
03DE 7	Time min Dewpoint:, hours BCD 1s	0445 0	
03DF 1	Time min Dewpoint:, hours BCD 10s	0446 0	Rain 24 h area
03E0 0	Date min Dewpoint:, BCD day 1s	0447 0	Rain 24 h area
03E1 2	Date min Dewpoint:, BCD day 10s	0448 0	Rain 24 h area
03E2 4	Date min Dewpoint:, BCD month 1s	0449 0	Rain 24 h area
03E3 0	Date min Dewpoint:, BCD month 10s	044A 0	Rain 24 h area
03E4 3	Date min Dewpoint:, BCD year 1s	044B 0	Rain 24 h area
03E5 0	Date min Dewpoint:, BCD year 10s	044C 0	Rain 24 h area
03E6 2	Time max Dewpoint:, minutes BCD 1s	044D 0	Rain 24 h area
03E7 0	Time max Dewpoint:, minutes BCD 10s	044E 0	Rain 24 h area
03E8 0	Time max Dewpoint:, hours BCD 1s	044F 0	Rain 24 h area
03E9 0	Time max Dewpoint:, hours BCD 10s	0450 0	Rain 24 h area
03EA 1	Date max Dewpoint:, BCD day 1s	0451 0	Rain 24 h area
03EB 0	Date max Dewpoint:, BCD day 10s	0452 0	Rain 24 h area
03EC 1	Date max Dewpoint:, BCD month 1s	0453 0	Rain 24 h area
03ED 0	Date max Dewpoint:, BCD month 10s	0454 0	Rain 24 h area
03EE 1	Date max Dewpoint:, BCD year 1s	0455 0	Rain 24 h area
03EF 0	Date max Dewpoint:, BCD year 10s	0456 0	Rain 24 h area
03F0 0		0457 0	Rain 24 h area
03F1 0	Low Alarm Dewpoint: BCD offset 30 0.01s [C]	0458 0	Rain 24 h area
03F2 0	Low Alarm Dewpoint: BCD offset 30 0.1s [C]	0459 0	Rain 24 h area
03F3 0	Low Alarm Dewpoint: BCD offset 30 1s [C]	045A 0	Rain 24 h area
03F4 3	Low Alarm Dewpoint: BCD offset 30 10s [C]	045B 0	Rain 24 h area
03F6 0	High Alarm Dewpoint: BCD offset 30 0.01s [C]	045C 0	Rain 24 h area
03F7 0	High Alarm Dewpoint: BCD offset 30 0.1s [C]	045D 0	Rain 24 h area
03F8 0	High Alarm Dewpoint: BCD offset 30 1s [C]	045E 0	Rain 24 h area
03F9 5	High Alarm Dewpoint: BCD offset 30 10s [C]	045F 0	Rain 24 h area
03FA 2		0460 0	Rain 24 h area
		0461 0	Rain 24 h area
		0462 0	Rain 24 h area
HUMIDITY INDOORS		0463 0	Rain 24 h area
03FB 4	Rel Humidity Indoors: BCD 1s [%]	0464 0	Rain 24 h area
03FC 2	Rel Humidity Indoors: BCD 10s [%]	0465 0	Rain 24 h area
03FD 9	Minimum Rel Humidity Indoors: BCD 1s [%]	0466 0	Rain 24 h area
03FE 1	Minimum Rel Humidity Indoors: BCD 10s [%]	0467 0	Rain 24 h area
03FF 7	Maximum Rel Humidity Indoors: BCD 1s [%]	0468 0	Rain 24 h area
0400 4	Maximum Rel Humidity Indoors: BCD 10s [%]		

0469 0 Rain 24 h area
 046A 0 Rain 24 h area
 046B 0 Rain 24 h area
 046C 0 Rain 24 h area
 046D 0 Rain 24 h area
 046E 0 Rain 24 h area
 046F 0 Rain 24 h area
 0470 0 Rain 24 h area
 0471 0 Rain 24 h area
 0472 0 Rain 24 h area
 0473 0 Rain 24 h area
 0474 0 Rain 24 h area
 0475 0 Rain 24 h area
 0476 C Rain 24 h area coding unknown
 0477 F
 0478 0
 0479 0 Rain 1h area coding unknown
 047A 0 Rain 1h area
 047B 0 Rain 1h area
 047C 0 Rain 1h area
 047D 0 Rain 1h area
 047E 0 Rain 1h area
 047F 0 Rain 1h area
 0480 0 Rain 1h area
 0481 0 Rain 1h area
 0482 0 Rain 1h area
 0483 0 Rain 1h area
 0484 0 Rain 1h area
 0485 0 Rain 1h area
 0486 0 Rain 1h area
 0487 0 Rain 1h area
 0488 0 Rain 1h area
 0489 0 Rain 1h area
 048A 0 Rain 1h area
 048B 0 Rain 1h area
 048C 0 Rain 1h area
 048D 0 Rain 1h area
 048E 0 Rain 1h area
 048F 0 Rain 1h area
 0490 0 Rain 1h area
 0491 0 Rain 1h area
 0492 0 Rain 1h area
 0493 0 Rain 1h area
 0494 0 Rain 1h area
 0495 0 Rain 1h area
 0496 0 Rain 1h area

 RAIN 24 HOUR
 0497 0 Rain 24 hour: BCD 0.01s [mm]
 0498 0 Rain 24 hour: BCD 0.1s [mm]
 0499 0 Rain 24 hour: BCD 1s [mm]
 049A 0 Rain 24 hour: BCD 10s [mm]
 049B 0 Rain 24 hour: BCD 100s [mm]
 049C 0 Rain 24 hour: BCD 1000s [mm]
 049D 8 Rain max 24 hour: BCD 0.01s [mm]
 049E 9 Rain max 24 hour: BCD 0.1s [mm]
 049F 8 Rain max 24 hour: BCD 1s [mm]
 04A0 2 Rain max 24 hour: BCD 10s [mm]
 04A1 1 Rain max 24 hour: BCD 100s [mm]
 04A2 0 Rain max 24 hour: BCD 1000s [mm]
 04A3 9 Time max rain 24h, minutes BCD 1s
 04A4 5 Time max rain 24h, minutes BCD 10s
 04A5 4 Time max rain 24h, hours BCD 1s
 04A6 1 Time max rain 24h, hours BCD 10s
 04A7 1 Date max rain 24h, BCD day 1s
 04A8 2 Date max rain 24h, BCD day 10s
 04A9 4 Date max rain 24h, BCD month 1s
 04AA 0 Date max rain 24h, BCD month 10s
 04AB 3 Date max rain 24h, BCD year 1s
 04AC 0 Date max rain 24h, BCD year 10s
 04AD 0
 04AE 0 Alarm Rain 24h, BCD 0.01 [mm] (probably not really used)
 04AF 0 Alarm Rain 24h, BCD 0.1 [mm]
 04B0 7 Alarm Rain 24h, BCD 1 [mm]
 04B1 0 Alarm Rain 24h, BCD 10 [mm]
 04B2 0 Alarm Rain 24h, BCD 100 [mm]
 04B3 0 Alarm Rain 24h, BCD 1000 [mm]

 RAIN 1 HOUR
 04B4 0 Rain 1 hour: BCD 0.01s [mm]
 04B5 0 Rain 1 hour: BCD 0.1s [mm]
 04B6 0 Rain 1 hour: BCD 1s [mm]
 04B7 0 Rain 1 hour: BCD 10s [mm]
 04B8 0 Rain 1 hour: BCD 100s [mm]
 04B9 0 Rain 1 hour: BCD 1000s [mm]
 04BA 6 Rain max 1 hour: BCD 0.01s [mm]
 04BB 4 Rain max 1 hour: BCD 0.1s [mm]
 04BC 8 Rain max 1 hour: BCD 1s [mm]
 04BD 2 Rain max 1 hour: BCD 10s [mm]
 04BE 1 Rain max 1 hour: BCD 100s [mm]
 04BF 0 Rain max 1 hour: BCD 1000s [mm]
 04C0 8 Time max rain 1h, minutes BCD 1s
 04C1 0 Time max rain 1h, minutes BCD 10s
 04C2 6 Time max rain 1h, hours BCD 1s
 04C3 1 Time max rain 1h, hours BCD 10s
 04C4 0 Date max rain 1h, BCD day 1s
 04C5 2 Date max rain 1h, BCD day 10s
 04C6 4 Date max rain 1h, BCD month 1s
 04C7 0 Date max rain 1h, BCD month 10s
 04C8 3 Date max rain 1h, BCD year 1s
 04C9 0 Date max rain 1h, BCD year 10s
 04CA 0
 04CB 0 Alarm Rain 1h, BCD 0.01 [mm] (probably not really used)
 04CC 7 Alarm Rain 1h, BCD 0.1 [mm]
 04CD 0 Alarm Rain 1h, BCD 1 [mm]
 04CE 0 Alarm Rain 1h, BCD 10 [mm]

 RAIN TOTAL
 04D2 2 Rain total: BCD 0.01s [mm]
 04D3 5 Rain total: BCD 0.1s [mm]
 04D4 4 Rain total: BCD 1s [mm]
 04D5 4 Rain total: BCD 10s [mm]
 04D6 1 Rain total: BCD 100s [mm]
 04D7 0 Rain total: BCD 1000s [mm]
 04D8 0 Time for reset of rain total, minutes BCD 1s
 04D9 4 Time for reset of rain total, minutes BCD 10s
 04DA 3 Time for reset of rain total, hours BCD 1s
 04DB 2 Time for reset of rain total, hours BCD 10s
 04DC 4 Date for reset of rain total, BCD day 1s
 04DD 1 Date for reset of rain total, BCD day 10s
 04DE 4 Date for reset of rain total, BCD month 1s
 04DF 0 Date for reset of rain total, BCD month 10s
 04E0 3 Date for reset of rain total, BCD year 1s
 04E1 0 Date for reset of rain total, BCD year 10s
 04E2 F
 04E3 F
 04E4 F
 04E5 F
 04E6 0
 04E7 0
 04E8 0
 04E9 0
 04EA 0
 04EB 0
 04EC 0
 04ED 0

 WIND MIN/MAX
 04EE 0 Minimum wind: binary nibble 0, nnnn/360 [m/s]
 04EF 0 Minimum wind: binary nibble 1, -
 04F0 0 Minimum wind: binary nibble 2, -
 04F1 0 Minimum wind: binary nibble 3, -
 04F2 0
 04F3 0
 04F4 C Maximum wind: binary nibble 0, nnnn/360 [m/s]
 04F5 D Maximum wind: binary nibble 1, -
 04F6 3 Maximum wind: binary nibble 2, -
 04F7 2 Maximum wind: binary nibble 3, -
 04F8 6 Time min wind, minutes BCD 1s
 04F9 2 Time min wind, minutes BCD 10s
 04FA 3 Time min wind, hours BCD 1s
 04FB 0 Time min wind, hours BCD 10s
 04FC 4 Date min wind, BCD day 1s
 04FD 2 Date min wind, BCD day 10s
 04FE 4 Date min wind, BCD month 1s
 04FF 0 Date min wind, BCD month 10s
 0500 3 Date min wind, BCD year 1s
 0501 0 Date min wind, BCD year 10s
 0502 5 Time max wind, minutes BCD 1s
 0503 2 Time max wind, minutes BCD 10s
 0504 3 Time max wind, hours BCD 1s
 0505 0 Time max wind, hours BCD 10s
 0506 4 Date max wind, BCD day 1s
 0507 2 Date max wind, BCD day 10s
 0508 4 Date max wind, BCD month 1s
 0509 0 Date max wind, BCD month 10s
 050A 3 Date max wind, BCD year 1s
 050B 0 Date max wind, BCD year 10s
 050C 0
 050D 0

 WIND ALARM (Read Only - Set values at 533-535 and 538-53A)
 050E 0 Low wind alarm setting (Read Only): binary nibble 0 [km/h / 100]
 050F 0 Low wind alarm setting (Read Only): binary nibble 1 [km/h / 100]
 0510 0 Low wind alarm setting (Read Only): binary nibble 2 [km/h / 100]
 0511 0 Low wind alarm setting (Read Only): binary nibble 3 [km/h / 100]
 0512 F
 0513 F
 0514 7 Low wind alarm setting (Read Only): binary nibble 0 [km/h / 100]
 0515 6 Low wind alarm setting (Read Only): binary nibble 1 [km/h / 100]
 0516 1 Low wind alarm setting (Read Only): binary nibble 2 [km/h / 100]
 0517 0 Low wind alarm setting (Read Only): binary nibble 3 [km/h / 100]
 0518 0
 0519 0 Alarmed windspeed (or current if no alarm): BCD 0.1s [m/s]
 051A 0 Alarmed windspeed (or current if no alarm): BCD 1s [m/s]
 051B 0 Alarmed windspeed (or current if no alarm): BCD 10s [m/s]
 051C 0

 WIND SPEED AND DIRECTION
 0527 0 Wind overflow flag: 0 = normal
 0528 0 Wind minimum code: 0=min, 1=---, 2=0FL
 0529 0 Windspeed: binary nibble 0 [m/s * 10]
 052A 0 Windspeed: binary nibble 1 [m/s * 10]
 052B 0 Windspeed: binary nibble 2 [m/s * 10]
 052C 8 Wind Direction = nibble * 22.5 degrees
 052D 8 Wind Direction 1 measurement ago
 052E 9 Wind Direction 2 measurement ago
 052F 8 Wind Direction 3 measurement ago
 0530 7 Wind Direction 4 measurement ago
 0531 7 Wind Direction 5 measurement ago
 0532 0

WIND ALARM SETTING (Changing this causes 50E-511 or 514-517 to be updated)

0533 0	Low wind alarm setting: BCD 0.1s [m/s]	062B 0	Time max air pressure, hours BCD 10s
0534 0	Low wind alarm setting: BCD 1s [m/s]	062C 8	Date max air pressure, BCD day 1s
0535 0	Low wind alarm setting: BCD 10s [m/s]	062D 1	Date max air pressure, BCD day 10s
0536 0		062E 4	Date max air pressure, BCD month 1s
0537 0		062F 0	Date max air pressure, BCD month 10s
0538 0	High wind alarm setting: BCD 0.1s [m/s]	0630 3	Date max air pressure, BCD year 1s
0539 1	High wind alarm setting: BCD 1s [m/s]	0631 0	Date max air pressure, BCD year 10s
053A 0	High wind alarm setting: BCD 10s [m/s]	0632 4	
053B 0		0633 1	
053C F		0634 6	
		0635 9	
		0636 0	
		0637 9	
		0638 3	
		0639 8	
054C D		063A 2	
054D 0	Connection Type: 0=Cable, 3=lost, F=Wireless	063B 0	
054E 0		063C 6	Low Alarm air pressure, BCD 0.1 [hPa]
054F C	Countdown time to next datBinary nibble 0, [0.5 sec]	063D 3	Low Alarm air pressure, BCD 1 [hPa]
0550 0	Countdown time to next datBinary nibble 1, [0.5 sec]	063E 6	Low Alarm air pressure, BCD 10 [hPa]
0551 0		063F 9	Low Alarm air pressure, BCD 100 [hPa]
0552 0		0640 0	Low Alarm air pressure, BCD 1000 [hPa]

AIR PRESSURE

05D8 2	Absolute air pressure: BCD 0.1s [hPa]	0641 6	
05D9 6	Absolute air pressure: BCD 1s [hPa]	0642 4	
05DA 1	Absolute air pressure: BCD 10s [hPa]	0643 8	
05DB 0	Absolute air pressure: BCD 100s [hPa]	0644 2	
05DC 1	Absolute air pressure: BCD 1000s [hPa]	0645 0	
05DD 0		0646 8	
05DE 0		0647 8	
05DF 0		0648 3	
05E0 3		0649 0	
05E1 0		064A 1	
05E2 4	Relative air pressure: BCD 0.1s [hPa]	064B 7	
05E3 8	Relative air pressure: BCD 1s [hPa]	064C 6	
05E4 1	Relative air pressure: BCD 10s [hPa]	064D 0	
05E5 0	Relative air pressure: BCD 100s [hPa]	064E 3	
05E6 1	Relative air pressure: BCD 1000s [hPa]	064F 0	
05E7 7		0650 0	High Alarm air pressure, BCD 0.1 [hPa]
05E8 0		0651 1	High Alarm air pressure, BCD 1 [hPa]
05E9 0		0652 4	High Alarm air pressure, BCD 10 [hPa]
05EA 3		0653 0	High Alarm air pressure, BCD 100 [hPa]
05EB 0		0654 1	High Alarm air pressure, BCD 1000 [hPa]
05EC 2	Air pressure correction: BCD 0.1s [hPa] offset 1000	0655 4	
05ED 2	Air pressure correction: BCD 1s [hPa] offset 1000	0656 7	
05EE 0	Air pressure correction: BCD 10s [hPa] offset 1000		HISTORY SETTINGS
05EF 0	Air pressure correction: BCD 100s [hPa] offset 1000	06B2 1	History saving interval: Binary nibble 0 [minutes]
05F0 1	Air pressure correction: BCD 1000s [hPa] offset 1000	06B3 0	History saving interval: Binary nibble 1 [minutes]
05F1 7		06B4 0	History saving interval: Binary nibble 2 [minutes]
05F2 0		06B5 1	Countdown to next saving: Binary nibble 0 [minutes]
05F3 0		06B6 0	Countdown to next saving: Binary nibble 1 [minutes]
05F4 0		06B7 0	Countdown to next saving: Binary nibble 2 [minutes]
05F5 1		06B8 6	Time last record, minutes BCD 1s
05F6 0	Absolute air pressure minimum: BCD 0.1s [hPa]	06B9 2	Time last record, minutes BCD 10s
05F7 3	Absolute air pressure minimum: BCD 1s [hPa]	06BA 3	Time last record, hours BCD 1s
05F8 1	Absolute air pressure minimum: BCD 10s [hPa]	06BB 0	Time last record, hours BCD 10s
05F9 0	Absolute air pressure minimum: BCD 100s [hPa]	06BC 4	Date last record, BCD day 1s
05FA 1	Absolute air pressure minimum: BCD 1000s [hPa]	06BD 2	Date last record, BCD day 10s
05FB 1		06BE 4	Date last record, BCD month 1s
05FC 9		06BF 0	Date last record, BCD month 10s
05FD 9		06C0 3	Date last record, BCD year 1s
05FE 2		06C1 0	Date last record, BCD year 10s
05FF 0		06C2 6	Pointer to last written Record: Binary nibble 0 [Range 00-AE]
0600 2	Relative air pressure minimum: BCD 0.1s [hPa]	06C3 9	Pointer to last written Record: Binary nibble 1
0601 5	Relative air pressure minimum: BCD 1s [hPa]	06C4 F	Number of Records: Binary nibble 0 [Range 00-AF]
0602 1	Relative air pressure minimum: BCD 10s [hPa]	06C5 A	Number of Records: Binary nibble 1
0603 0	Relative air pressure minimum: BCD 100s [hPa]		HISTORY
0604 1	Relative air pressure minimum: BCD 1000s [hPa]	4,3,2,1,0: Indoor and outdoor temperature	
0605 7		Tindoor = (value % 1000)/10 - 30 [C]	
0606 9		Toutdoor = (value - (value % 1000))/10000 - 30 [C]	
0607 9		Where % is the modulus operator.	
0608 2		9,8,7,6,5: Absolute Air Pressure and Indoor Humidity.	
0609 0		Pressure= 1000 + (value % 10000)/10. If pressure is greater than or equal to 1500 then you subtract 1000.	
060A 6	Absolute air pressure maximum: BCD 0.1s [hPa]	Indoor humidity = (value - (value % 10000))/10000	
060B 4	Absolute air pressure maximum: BCD 1s [hPa]		
060C 3	Absolute air pressure maximum: BCD 10s [hPa]	11,10: Outdoor Humidity in plain human readable BCD	
060D 0	Absolute air pressure maximum: BCD 100s [hPa]	14,13,12: Rain. (RAINCOUNTn)	
060E 1	Absolute air pressure maximum: BCD 1000s [hPa]	The value is binary and steps 0.518 mm/step.	
060F 5		The absolute value does not seem to be related to anything else than an internal "household" value inside the station.	
0610 5		Every period the current 12-bit rain count value is stored as history data.	
0611 0		You use it by keeping a reference total rain value RAINref, the corresponding reference count RAINCOUNTref.	
0612 3		RAINTotal = RAINref + (RAINCOUNTn - RAINCOUNTref)*0.518 [mm]	
0613 0			
0614 8	Relative air pressure maximum: BCD 0.1s [hPa]	17-16-15: Windspeed = value in binary / 10 [m/s]	
0615 6	Relative air pressure maximum: BCD 1s [hPa]	18: Wind direction = value * 22.5 degrees. North is 0 and degrees are clockwise on the circle.	
0616 3	Relative air pressure maximum: BCD 10s [hPa]		
0617 0	Relative air pressure maximum: BCD 100s [hPa]		
0618 1	Relative air pressure maximum: BCD 1000s [hPa]		
0619 1		Record 0	
061A 6		06C6 F History data area should start here	
061B 0		06C7 C	
061C 3		06C8 1	
061D 0			
061E 2	Time min air pressure, minutes BCD 1s		
061F 0	Time min air pressure, minutes BCD 10s		
0620 5	Time min air pressure, hours BCD 1s		
0621 0	Time min air pressure, hours BCD 10s		
0622 3	Date min air pressure, BCD day 1s		
0623 2	Date min air pressure, BCD day 10s		
0624 4	Date min air pressure, BCD month 1s		
0625 0	Date min air pressure, BCD month 10s		
0626 3	Date min air pressure, BCD year 1s		
0627 0	Date min air pressure, BCD year 10s		
0628 6	Time max air pressure, minutes BCD 1s		
0629 4	Time max air pressure, minutes BCD 10s		
062A 2	Time max air pressure, hours BCD 1s		



Module Système 6

Partie d'échecs en réseau retransmise
sur Internet et sur écran géant à LED

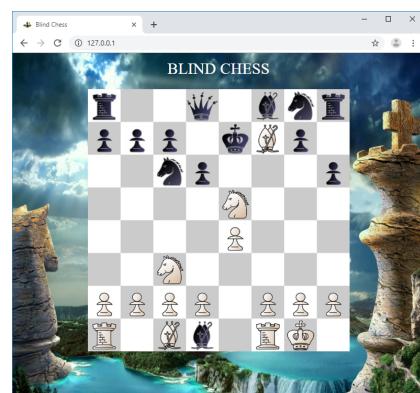


Table des matières

Objectifs, matériaux et ressources.....	2
TD1 – Cahier des charges et analyse UML.....	4
TD2 – Analyse des diagrammes de classes.....	6
TP – Défi 1 – Déplacement des pièces.....	8
TP – Défi 2 – Gestion du trait -le tour de jeu-.....	12
TP – Défi 3 – Gestion des déplacements particuliers.....	14
TP – Défi 4 – Jeu en réseau : connexion et envoi de la couleur.....	17
TP – Défi 5 – Jeu en réseau : envoi et réception des déplacements.....	19
TP – Défi 6 – Génération de l'image de la position.....	22
TP – Défi 7 – Serveur HTTP.....	24
TP – Défi 8 – Interface graphique et gestion du temps.....	25
Annexe 1 : les règles du jeu.....	28
Annexe 2: diagrammes de classes.....	31

Objectifs, matériels et ressources

Objectifs

A partir d'un diagramme de classe détaillé, concevoir une IHM (interface homme-machine) client-serveur, permettant de jouer aux échecs à l'aveugle (ou non), dans une cadence choisie. Les règles du jeu doivent être vérifiées. Le logiciel intégrera un serveur Web en C++ afin que les parties soient visibles depuis un navigateur Web.

- Client TCP en C++
- Serveur TCP en C++
- Serveur HTTP en C++
- Interface graphique
- Thread C++ Builder
- Algorithmie avancée : tableaux à deux dimensions
- UML (cas d'utilisation, déploiement, classe, séquence, activité)

Matériels

- Afficheur géant lumineux à LED



- Journal lumineux à LED

Codes sources

- SNIImage.h
- SNIImage.obj
- SNClientTCP.h
- SNClientTCP.cpp
- SNServeurHTTP.h
- SNServeurHTTP.cpp
- SNServeurTcpMonoclient.h
- SNServeurTcpMonoclient.cpp
- ReglesEtNotation.h
- ReglesEtNotation.cpp
- Deplacement.h

Ressources et documentation

- Pièces.zip
- Web.zip
- Diagrammes UML de classes : impression recto-verso séparée.
- Documentation constructeur de l'afficheur.

- Deplacement.cpp

Logiciels

- C++ Builder
- Logiciel constructeur de l'afficheur.
- Logiciel constructeur du journal lumineux
- PositionBMP2JPG.exe

Annexes

- extrait du site : <https://www.pousseurdebois.fr/cours/regles-du-jeu-d-echecs/>
- Diagrammes de classes

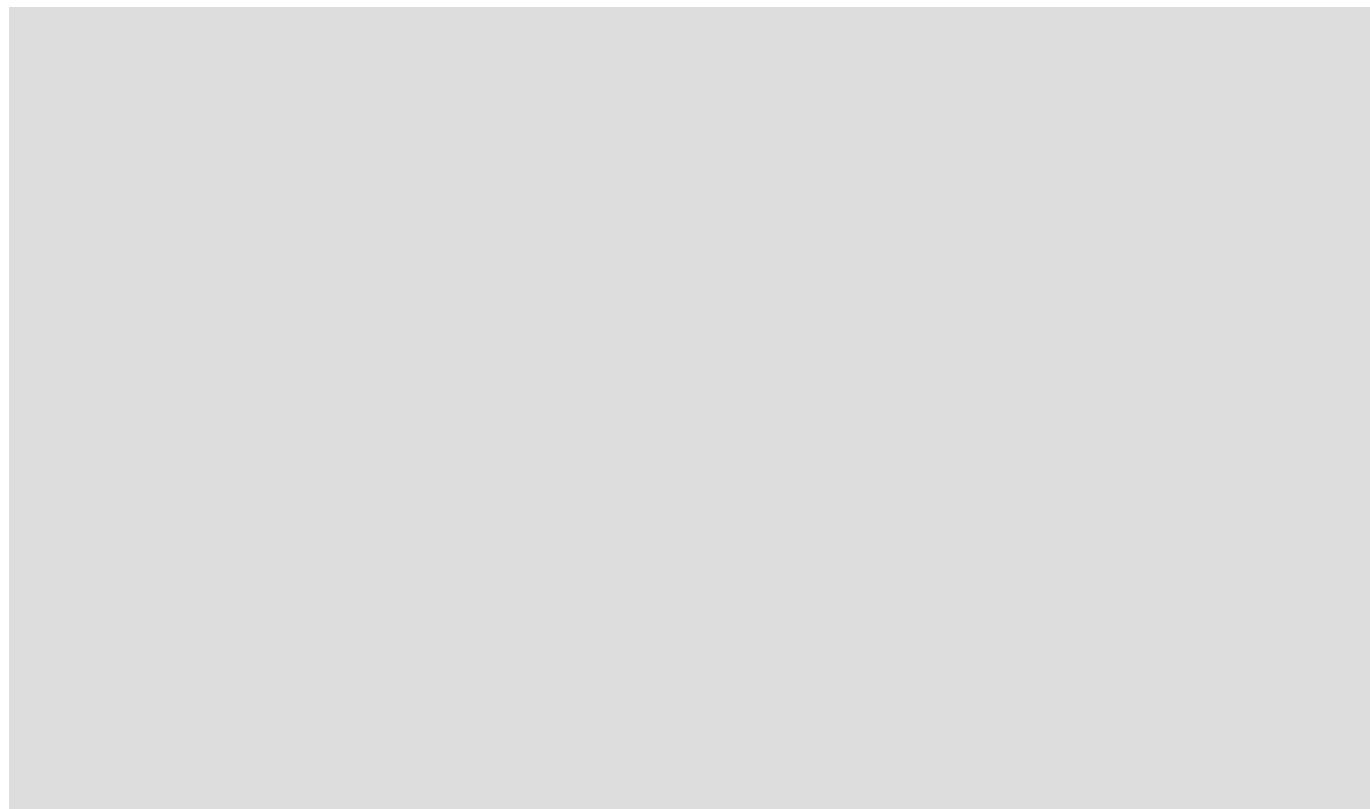
TD1 – Cahier des charges et analyse UML

Le cahier des charges

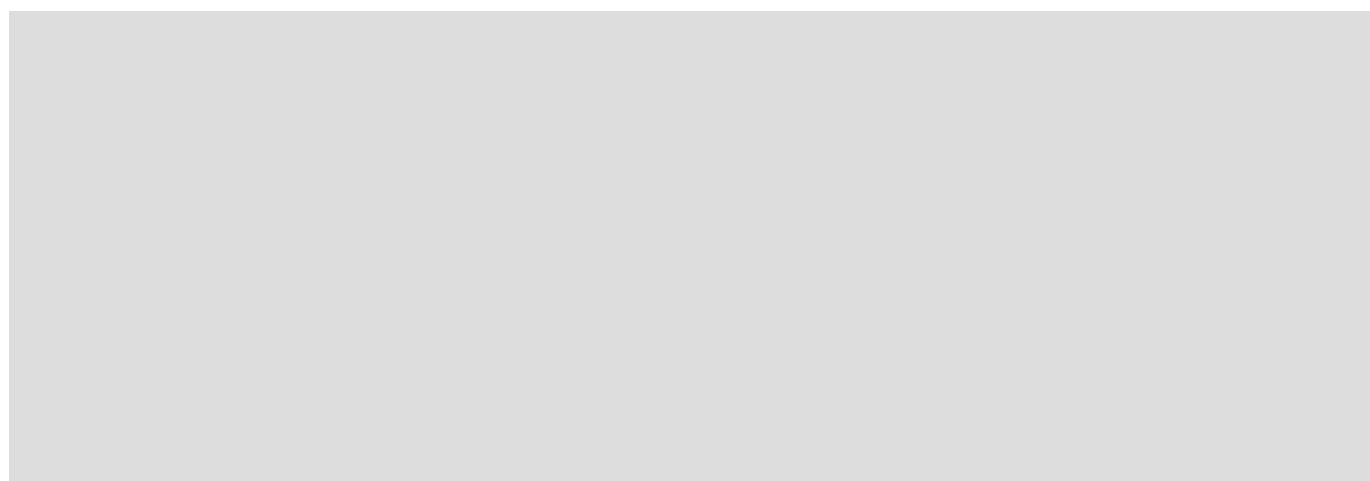
Lors des tournois d'échecs à l'aveugle, les compétiteurs jouent en réseau sur des ordinateurs, en déplaçant des pièces invisibles sur un échiquier vide. Le public peut suivre les matchs grâce à une retransmission sur écran géant à LED avec les pièces visibles, le temps restant à chaque joueur est indiqué (en l'absence d'écran géant, les coups sont indiqués sur un journal lumineux). Les parties sont également suivies sur Internet en temps réel, pour les abonnés disposant d'un login et d'un mot de passe. Les parties seront stockées au format PGN dans une base de données, elles pourront être rejouées.

Les diagrammes de cas d'utilisation, de déploiement et de séquence

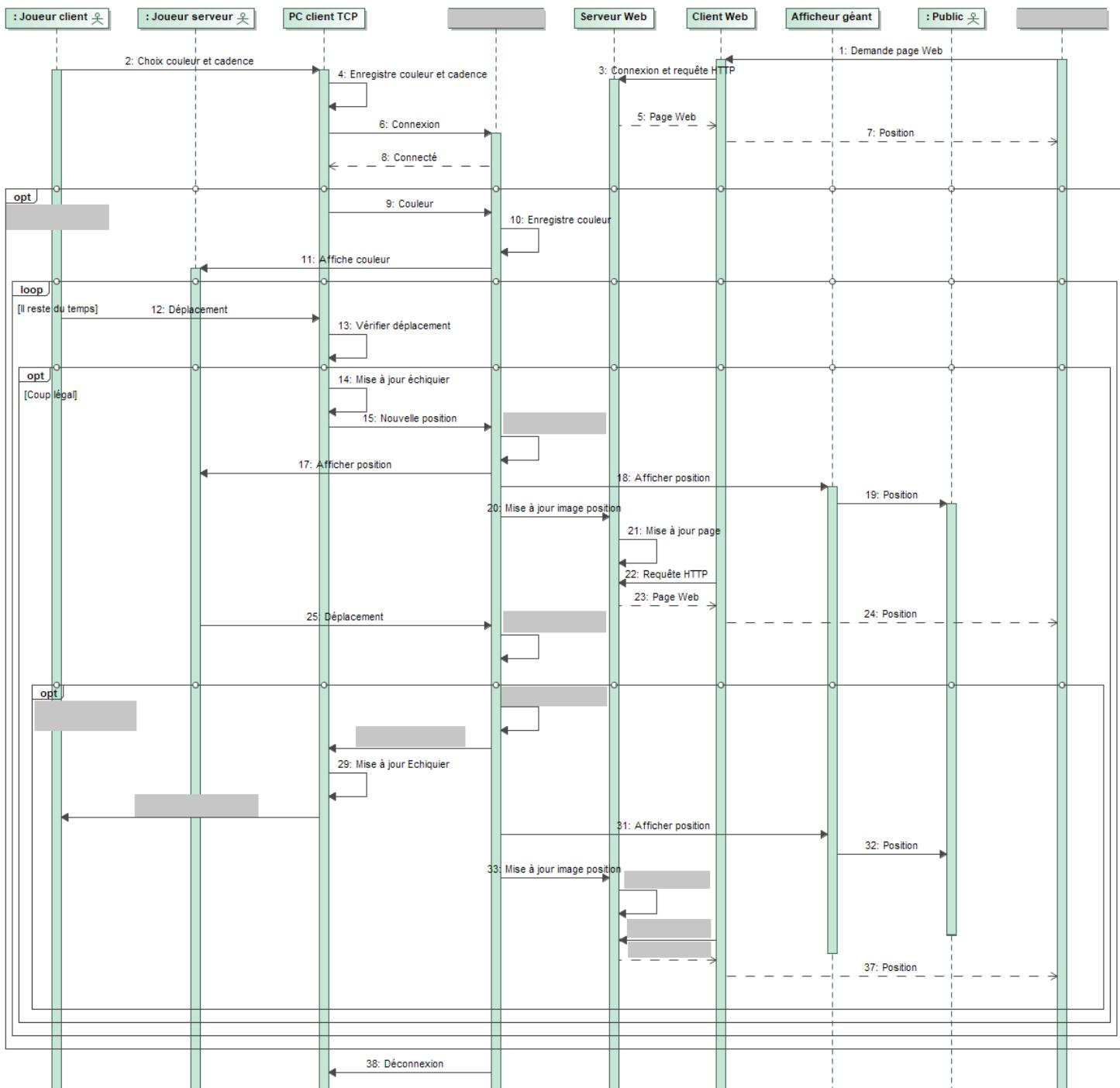
Dessiner le diagramme de cas d'utilisation :



Dessiner le diagramme de déploiement :



Compléter le diagramme UML de séquence du système complet en remplaçant les zones grisées :



TD2 – Analyse des diagrammes de classes

Les diagrammes de classes sont donnés en annexe.

Convention : les noms des variables et des attributs commencent par des minuscules. Une majuscule sépare les mots qui composent une variable. Une méthode ou une fonction commence par une majuscule.

Un coup est composé d'un déplacement blanc suivi d'un déplacement noir, qu'est ce qu'un demi-coup ?

Le trait est le tour de jeu : on dit que les blancs ont le trait signifie que c'est à eux de jouer. Donner et justifier le type de l'attribut trait, à quelle classe appartient cet attribut ?

Quels attributs permettent de gérer le temps restant ?

tabEchiquier est un tableau à 2 dimensions contenant autant de caractères que de cases sur l'échiquier : donner l'instruction permettant de déclarer ce tableau :

A quoi sert la classe partie ?

Que signifient les 2 liaisons entre les classes Echiquier et Joueur, quel nom porte ce type de lien ?

Quelles classes permettent de vérifier les règles ?



Quelle méthode indique si un déplacement est possible ? justifier son type de retour.



Nommer et justifier le lien entre Deplacement et DeplacementDame :



Quelles classes permettent de jouer en réseau ?



Quelle classe permet le suivi de la partie sur Internet ?



Un Thread est un processus léger : quel est son utilité ?



Donner le nom des 3 objets de type thread dans le diagramme de classes.



Dans la classe thread : ThClientServeur, quelle instruction permet d'initialiser le serveur TCP ?



Dans la classe thread : Quelle instruction permet au client de se connecter ?



Quelle classe permet de gérer l'interface graphique ?

TP – Défi 1 – Déplacement des pièces

La première classe à coder est la classe Echiquier.



Donner la déclaration de la classe Echiquier partielle donnée ici, sachant que tabEchiquier est tableau à 2 dimensions : [8][8] à la place de [0..*], les types char"*\$*" doivent être remplacés par char*.

Echiquier
-tabEchiquier : char [0..*]
-demiCoup : int
+Echiquier()
+LireCaseEchiquier(ligne : int, colonne : int) : char
+InitialiserEchiquier() : void
+VisualiserEchiquier() : string
+Les64Caracteres() : string
+ChargerEchiquierComplet(les64Caracteres : char \$" "*) : void
+ModifierCaseEchiquier(cPiece : char, ligne : int, colonne : int) : void
+EstUnePieceNoire(ligne : int, colonne : int) : bool
+EstUnePieceBlanche(ligne : int, colonne : int) : bool
+EstVide(ligne : int, colonne : int) : bool
+ChangerLeTrait() : void
+Deplacer(idep : int, jdep : int, iarr : int, jarr : int, piecePromotion : char) : bool
+Trait() : string
+DemiCoup() : int
+SauvegarderEchiquierBMP(fichier : char \$" "*) : void

Pour analyser la position (vérifier les règles), et pour transmettre la position en TCP/IP, il est nécessaire de la stocker dans un tableau de lettres à 2 dimensions : tabEchiquier est un tableau contenant des lettres minuscules pour les noirs, et majuscules pour les blancs. Les cases vides sont représentées par un espace ''.

Le premier indice est la ligne, le second la colonne : ainsi tabEchiquier[7][3] contient une dame blanche 'D'.

tcfdrfct
pppppppp

PPPPPPPP
TCFDRFCT



Compléter la définition de la méthode InitialiserEchiquier() :

```
void Echiquier::InitialiserEchiquier()
{
    tabEchiquier[0][0] = 't'; tabEchiquier[0][1] = 'c';
    tabEchiquier[0][2] = 'f'; tabEchiquier[0][3] = 'd';
    tabEchiquier[0][4] = 'r'; tabEchiquier[0][5] = 'f';
    tabEchiquier[0][6] = 'c'; tabEchiquier[0][7] = 't';
    tabEchiquier[ ] [ ] = ; tabEchiquier[ ] [ ] = ;
    tabEchiquier[ ] [ ] = ; tabEchiquier[ ] [ ] = ;
    tabEchiquier[ ] [ ] = ; tabEchiquier[ ] [ ] = ;
    tabEchiquier[ ] [ ] = ; tabEchiquier[ ] [ ] = ;
    for(int j=0; j<8 ; j++)
    {   tabEchiquier[1][j] = 'p';
        [REDACTED]
    }
    for(int i=2; i<=5 ; i++)
    {
        [REDACTED]
        {   tabEchiquier[i][j] = [REDACTED];
        }
    }
}
```



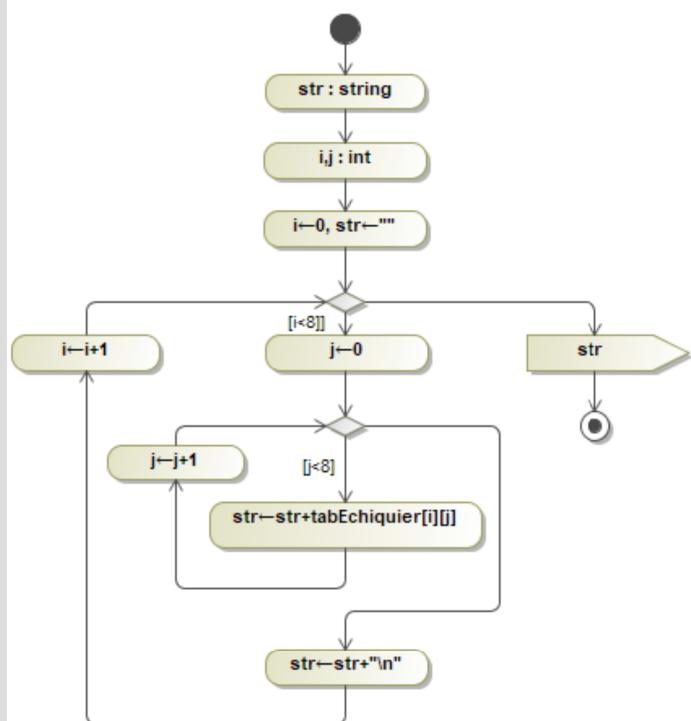
Donner la définition du constructeur chargé d'appeler la méthode d'initialisation :

[REDACTED]



En suivant le diagramme d'activité fourni, donner la définition de la méthode VisualiserEchiquier(), méthode qui construit et retourne une chaîne de caractères de type string contenant la position sur l'échiquier :

[REDACTED]





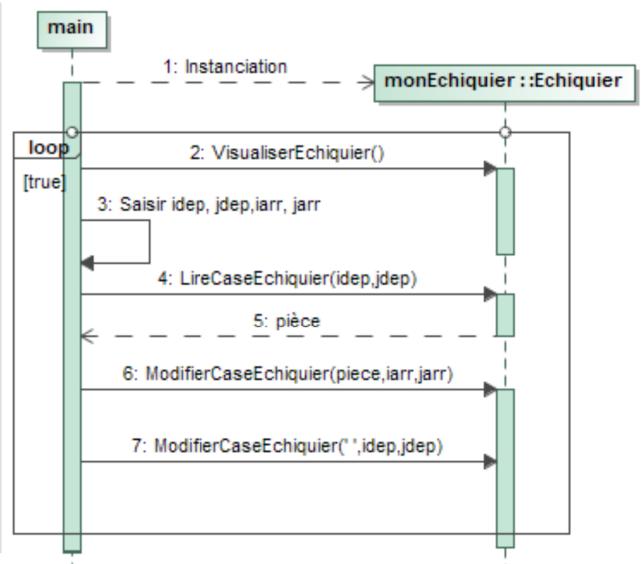
Donner la définition de la méthode LireCaseEchiquier(...) chargé de retourner le caractère présent dans la case [ligne] [colonne] du tableau tabEchiquier :



Donner la définition de la méthode ModifierCaseEchiquier(...) chargé de placer le caractère cPiece dans la case [ligne] [colonne] du tableau tabEchiquier :



Donner le programme principal correspondant au diagramme de séquence suivant :



Coder et tester votre programme en mode console.



Modifier le programme principal : la saisie de -1 pour la ligne de départ entraîne la sortie de la boucle (instruction break).

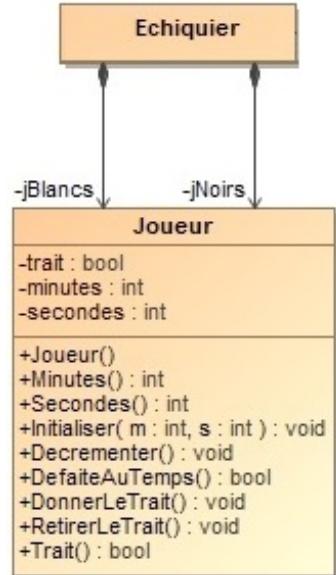


Afin d'alléger le main, placer les méthodes permettant le déplacement dans la méthode Deplacer. Tester votre programme en appelant dans le main cette méthode (mettre un espace '' dans l'argument piecePromotion).

TP – Défi 2 – Gestion du trait -le tour de jeu-

Chaque joueur doit jouer à son tour. Pour gérer le trait, une classe Joueur doit être codée, deux objets jBlancs et jNoirs seront ajoutés à la classe Echiquier.

Donner la déclaration de la classe Joueur :



Donner la définition des méthodes DonnerLeTrait() et RetirerLeTrait(), ces méthodes permettent de mettre l'attribut trait à true ou bien à false. Donner la définition de la méthode Joueur::Trait() permettant de donner l'accès en lecture à l'attribut du même nom :

Quelle ligne faut-il ajouter à InitialiserEchiquier() afin de donner le trait au blancs en début de partie :

Donner la définition de la méthode ChangerLeTrait() : si le trait est aux blancs, il faut le donner aux noirs et réciproquement :



Donner la définition de la méthode Echiquier::Trait() qui reverra "blancs" si les blancs ont le trait, et "noirs" si c'est aux noirs de jouer :

Pour détecter la couleur d'une pièce, où bien l'absence de pièce sur une case, il faut tester si la case contient un espace, une majuscule ou bien une minuscule.



Donner la définition des méthodes EstVide, EstUnePieceNoire et EstUnePieceBlanche. :

Il ne reste plus qu'à vérifier qu'au tour des blancs on déplace une pièce blanche, et une pièce noire au tour des noirs.
Dans la méthode Deplacer il faut soumettre le déplacement à une longue condition :

SI les cases de départ et d'arrivée sont bien dans l'échiquier (lignes et colonnes entre 0 et 7)

ET

(les blancs ont le trait

ET

une pièce blanche est bien sur la case de départ

ET

une pièce blanche ne se trouve pas sur la case d'arrivée)

OU

(les noirs ont le trait

ET

une pièce noire est bien sur la case de départ

ET

une pièce noire ne se trouve pas sur la case d'arrivée)

)

ALORS effectuer le déplacement puis changer le trait.



Donner en C++ la condition nécessaire au déplacement :

La méthode Deplacer renverra true si le déplacement est possible, false dans le cas contraire.



Coder et tester votre programme : un message "coup illégal" indiquera si le déplacement ne respecte pas le tour de jeu.

TP – Défi 3 – Gestion des déplacements particuliers

Les règles du jeu sont données en annexe.

Promotion :

Dans le programme principal, lorsqu'un pion atteint la case de promotion, il faut demander au joueur de choisir la pièce de promotion : D T F C ou bien d t f c.

Donner la condition de promotion en C++ :

Lorsqu'il y a promotion, la pièce choisie est transmise à la méthode Deplacer (piecePromotion). Dans la méthode Deplacer, après avoir effectuer le déplacement, si un pion se retrouve sur la case d'arrivée et que celle ci est située sur la ligne 0 pour les blancs ou sur la ligne 7 pour les noirs, alors ce pion doit être remplacé par piecePromotion.

 Coder et tester votre programme.

Roque :

Dans déplacer, il faut gérer 4 cas différents : le grand roque blancs, le petit roque blancs, le grand roque noirs, le petit roque noirs.

Cas du grand roque blancs :

7 T R
0 1 2 3 4

Les cases entre R et T doivent être vides, R doit être en (7,4) et T en (7,0) : alors si R se déplace en (7,2) la tour se déplace automatiquement en (7,3).

Donner le code permettant de tester les conditions de ce grand roque et de déplacer la T de (7,0) à (7,3) :

 Coder et tester votre programme afin de prendre en compte les 4 différents roques.

Prise en passant :

Un pion capture en diagonale, dans un seul cas il se rend en diagonale sur une case vide : il prend en passant un autre pion qui vient de sauter une case.

Cas du trait aux blancs :

lorsqu'un pion blanc est sur la ligne 3, s'il se déplace en diagonale sur une case vide (2,jdep-1) ou (2,jdep+1) et qu'un pion adverse se trouve respectivement en (3,jdep-1) ou (3,jdep+1), alors ce dernier disparaît.

0	1	0
1	1	1
2	=>	2 P
3 P P		3
0 1 2 3		0 1 2 3

 Donner en C++ la condition de prise en passant d'un pion noir par un pion blanc. :

Dans le cas du trait au noir, les pions vont vers le bas, la ligne de départ doit être la ligne 4, celle d'arrivée la ligne 5.

 Donner en C++ la condition de prise en passant d'un pion blanc par un pion noir :

 Modifier la méthode Deplacer afin de permettre d'effectuer la prise en passant. Tester votre programme.

TP – Défi 4 – Jeu en réseau : connexion et envoi de la couleur

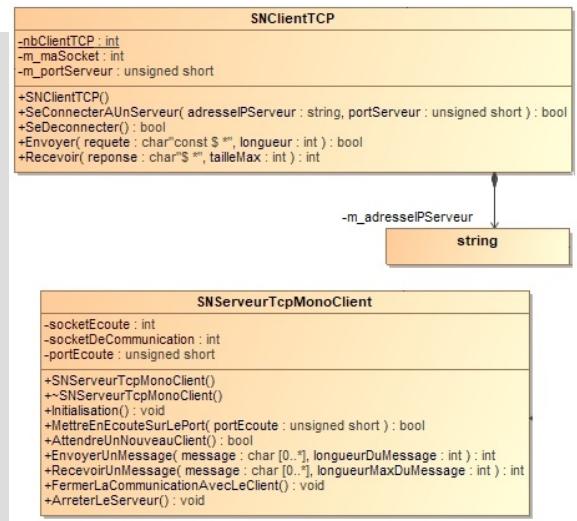
Les classes C++ permettant de coder un serveur TCP et un client TCP sont données. Seul le programme principal sera modifié : une variable entière mode permettra de gérer 3 modes (local, serveur et client). Une variable jaiLesBlancs indiquera la couleur du joueur lors du jeu en réseau. 2 tableaux de caractères seront nécessaires : IP pour stocker l'adresse IP du serveur, et message qui stockera le message reçu ou bien le message à envoyer. Le message contiendra l'ensemble des caractères représentant une position.



Donner la déclaration des 4 variables à ajouter, en précisant la taille des 2 tableaux :



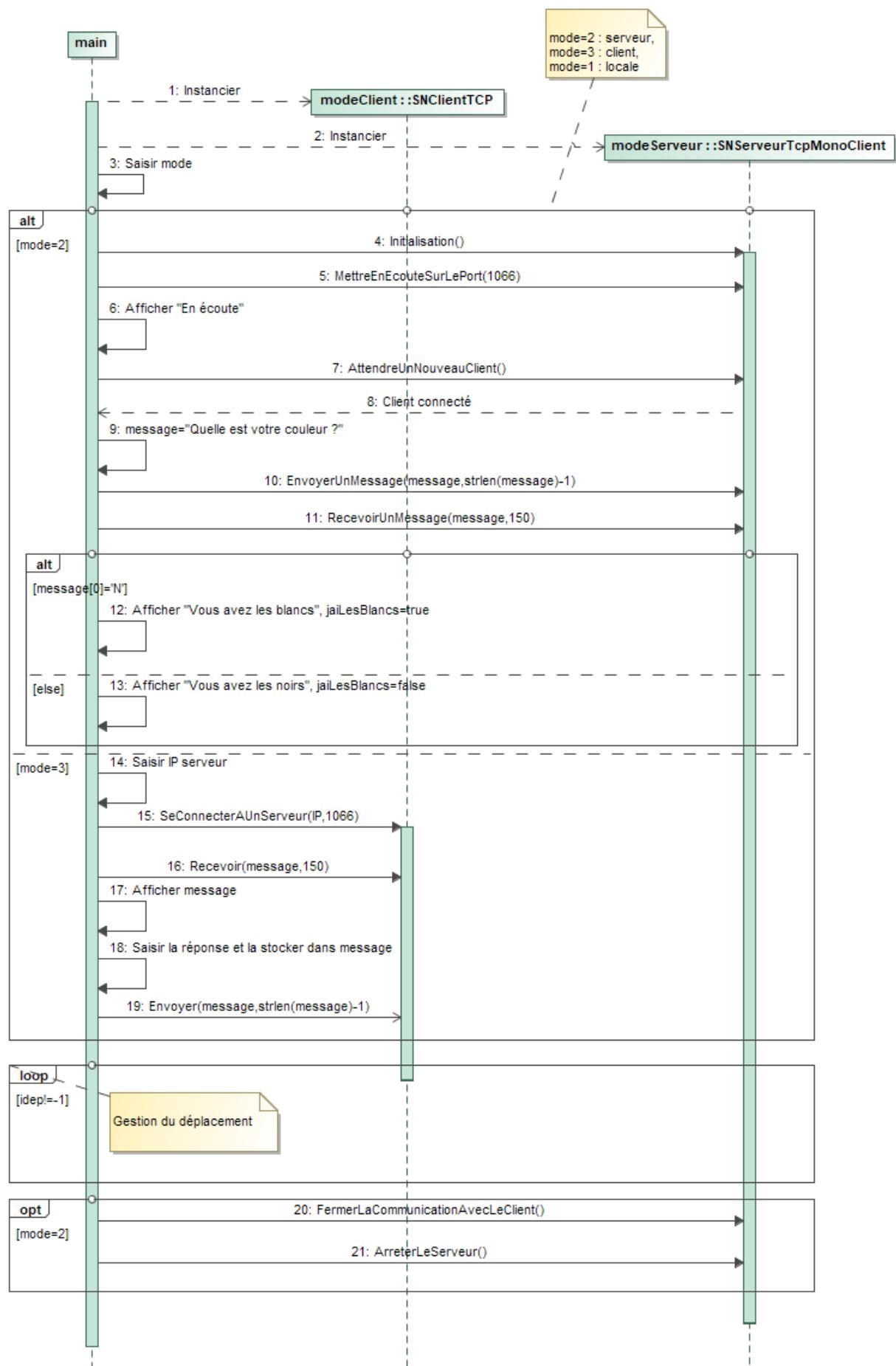
Donner le code partiel du main (sans la gestion du déplacement), correspondant au diagramme de séquence donné page suivante :



Qui choisit sa couleur, le serveur ou bien le client ?



Compléter et tester votre programme principal.



TP – Défi 5 – Jeu en réseau : envoi et réception des déplacements

Dans la classe Echiquier, deux méthodes doivent être codées : Les64Caracteres et ChargerEchiquierComplet.

Les64Caracteres() vise à mettre sur une même ligne les 64 caractères de l'échiquier. Cette méthode ressemble fortement à VisualiserEchiquier() : il faut y ôter les "\n".

Donner la définition de la méthode Les64Caracteres() :

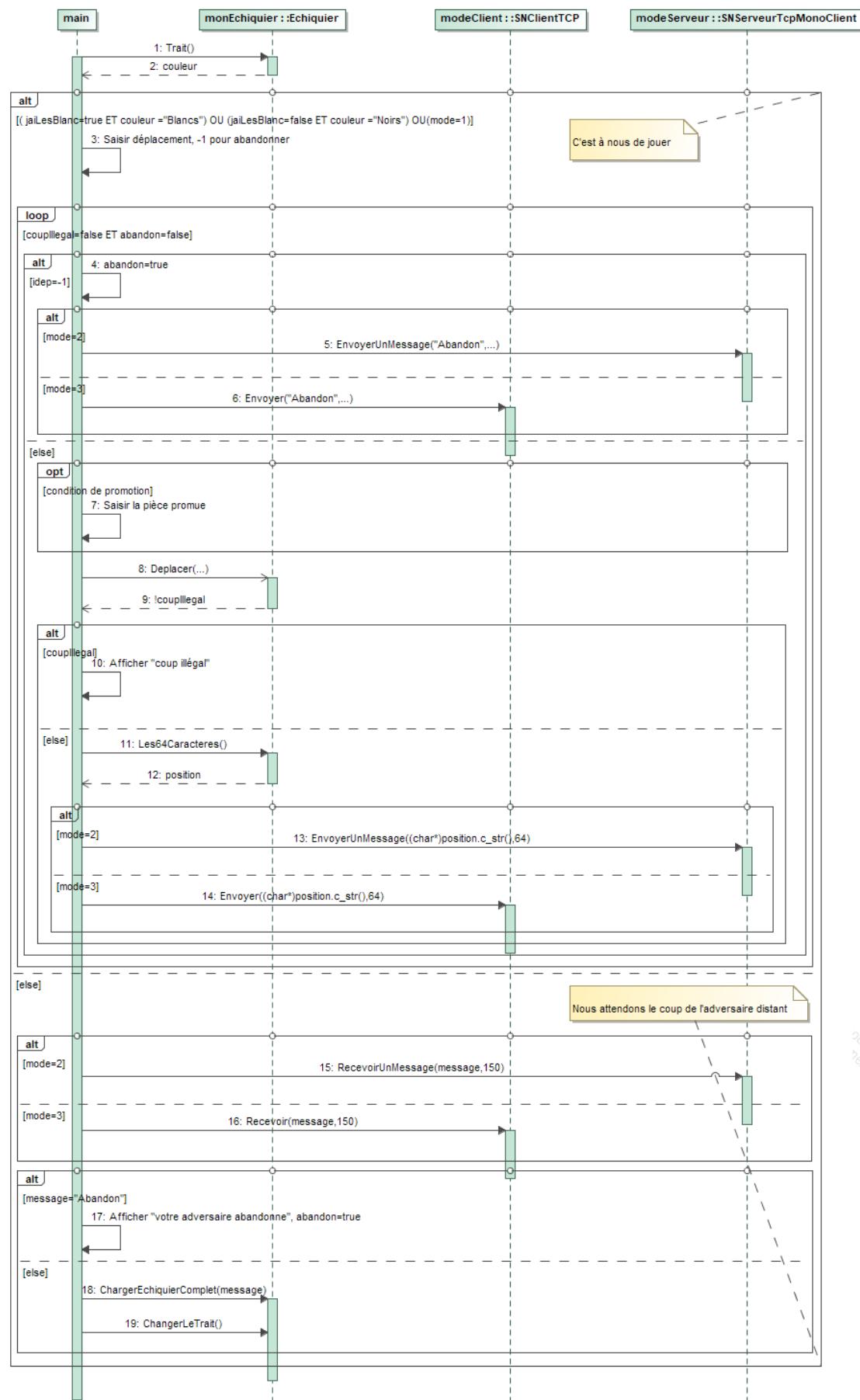
La méthode ChargerEchiquierComplet permet de modifier l'échiquier en y chargeant la nouvelle position envoyée par un adversaire distant : les64Caractères est le tableau de caractères reçu.

Compléter l'algorithme suivant correspondant à la méthode ChargerEchiquierComplet :

```
k←0  
Pour i allant de 0 à 7 par incrément de 1  
    Pour j allant de 0 à 7 par incrément de 1  
        tabEchiquier[i] [j]←  
        k←k+1  
    FinPour  
FinPour
```

Donner la définition de la méthode ChargerEchiquierComplet :

Compléter le programme principal en suivant le diagramme de séquence suivant : ce diagramme représente le contenu de la boucle de déplacement do{ ... }while (!abandon); : tester votre programme.



Facultatif : Vérification des règles et notation des coups

Les classes ReglesEtNotation et Deplacement sont données.

Dans la classe Echiquier :

Il faut ajouter les attributs et les méthodes concernant le roque, la mémorisation de l'échiquier précédent (pour la prise en passant notamment), la notation et le numéro du demi-coup :

int demiCoup; //0 au départ, 1 après le premier coup des blancs, 2 après celui des noirs ...
(à modifier avant notation)

char tabEchiquierPrecedent[8][8]; //contient la position précédente

bool grandRoqueNoirPossible, petitRoqueNoirPossible, grandRoqueBlancPossible,
petitRoqueBlancPossible;

//sont positionnés à false si le roi ou la tour concernée ont bougé

void SauvegarderEchiquierPrecedent(char tEchiquier[8][8]);

//pour la prise en passant ET après notation

//permet de copier les 64 caractères de tEchiquier dans tabEchiquierPrecedent

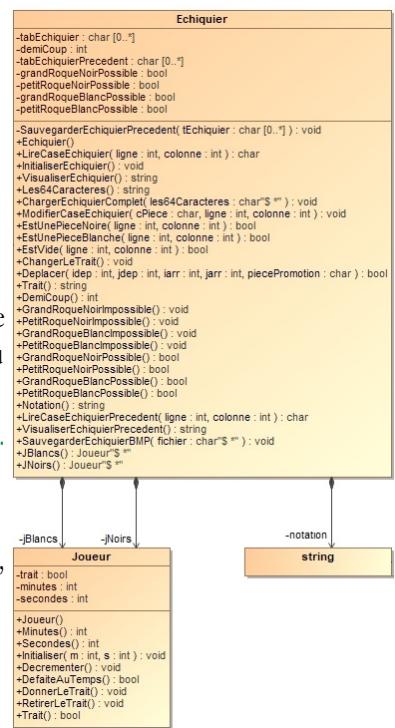
string notation; //contiendra la notation du coup, par exemple 1:e2-e4 signifie que le pion e2 se déplace en e4

int DemiCoup(); //permet l'accès à demiCoup

void GrandRoqueNoirImpossible(); //positionne grandRoqueNoirPossible à false, même principe pour les 3 autres cas

bool PetitRoqueBlancPossible(); //permet l'accès à petitRoqueBlancPossible, même principe pour les 3 autres cas

string Notation(); //permet l'accès à notation



Modifier la déclaration de la classe Echiquier et ajouter la définition des nouvelles méthodes.

Dans la méthode Déplacer :

Créer un objet de la classe ReglesEtNotation.

Un déplacement sera possible si DeplacementPossible de la classe ReglesEtNotation renvoie true. Le pointeur sur Echiquier à passer en argument est l'objet courant : indiquer le mot clé **this**.

Après un déplacement, incrémenter demicoup.

Stocker dans l'attribut notation le résultat de la méthode EnregistrerNotation.

Appeler la méthode SauvegarderEchiquierPrecedent en passant en argument tabEchiquier.

Modifier la méthode Deplacer et tester le fonctionnement du programme : le programme principal affichera la notation correspondante à chaque coups. Les déplacements illégaux ne sont pas permis.

TP – Défi 6 – Génération de l'image de la position

La méthode SauvegarderEchiquierBMP utilise la classe SNImage donnée : .h et .obj (le .obj est le .cpp déjà compilé). Cette méthode va générer une image de la position en utilisant les images des pièces (100x100), ainsi que Echiquier.bmp : une image blanche de 800x800. Elle contient quatre boucles for imbriquées.

En étudiant SNimage.h, que représente le type de variable Pixel, s'agit-il d'un tableau, d'une classe, d'une structure... ?

Que représente l'attribut image de la classe SNimage, que signifie ** lors de sa déclaration ?

Quelle méthode permet de charger une image ?

Qu'est ce qu'un modulo, et quel symbole représente cet opérateur en langage C ?

A quelle classe appartient la méthode SauvegarderEchiquierBMP ?

Traduire en C++ l'algorithme suivant de la méthode SauvegarderEchiquierBMP :

```

Créer un objet SNImage pour chaque type de pièces de l'échiquier : cavB, cavN, fouB...
Charger les images avec les objets SNImage créés (cavB.bmp, cavN.bmp, fouB.bmp...).
Créer un objet position de la classe SNImage et charger Echiquier.bmp.
Créer 2 variables de type Pixel : blanc={255,255,255} et gris={200,200,200}
Pour ligne allant de 0 à 8 par incrément de 1
    Pour colonne allant de 0 à 8 par incrément de 1
        Pour i allant de 0 à 100 par incrément de 1
            Pour j allant de 0 à 100 par incrément de 1
                Créer une variable couleur de type Pixel
                Initialiser couleur à la valeur de cavB.image[0][0]
                Si tabEchiquier[ligne][colonne]='C'
                    couleur ← cavB.image[i][j]
                FinSi
                ... faire ainsi pour les 12 types de pièces ...
                Si couleur=cavB.image[0][0] (la couleur de transparence)
                    Si (ligne+colonne) modulo 2 est différent de 0
                        position.image[ligne*100+i][colonne*100+i] ← gris
                    Sinon
                        position.image[ligne*100+i][colonne*100+i] ← blanc
                    FinSi
                Sinon
                    position.image[ligne*100+i][colonne*100+i] ← couleur
                FinSi
            FinPour
        FinPour
    FinPour
Sauvegarder à l'aide de l'objet position l'image dans "Position.bmp".
```

Votre code de la méthode SauvegarderEchiquierBMP :



Coder cette méthode SauvegarderEchiquierBMP, puis appeler cette méthode après VisualiserEchiquier dans le programme principal. Tester votre programme.

Il reste à convertir le BMP et JPG :



Quelles sont les différences entre une image bitmap (.bmp) et une image jpeg (.jpg) ?



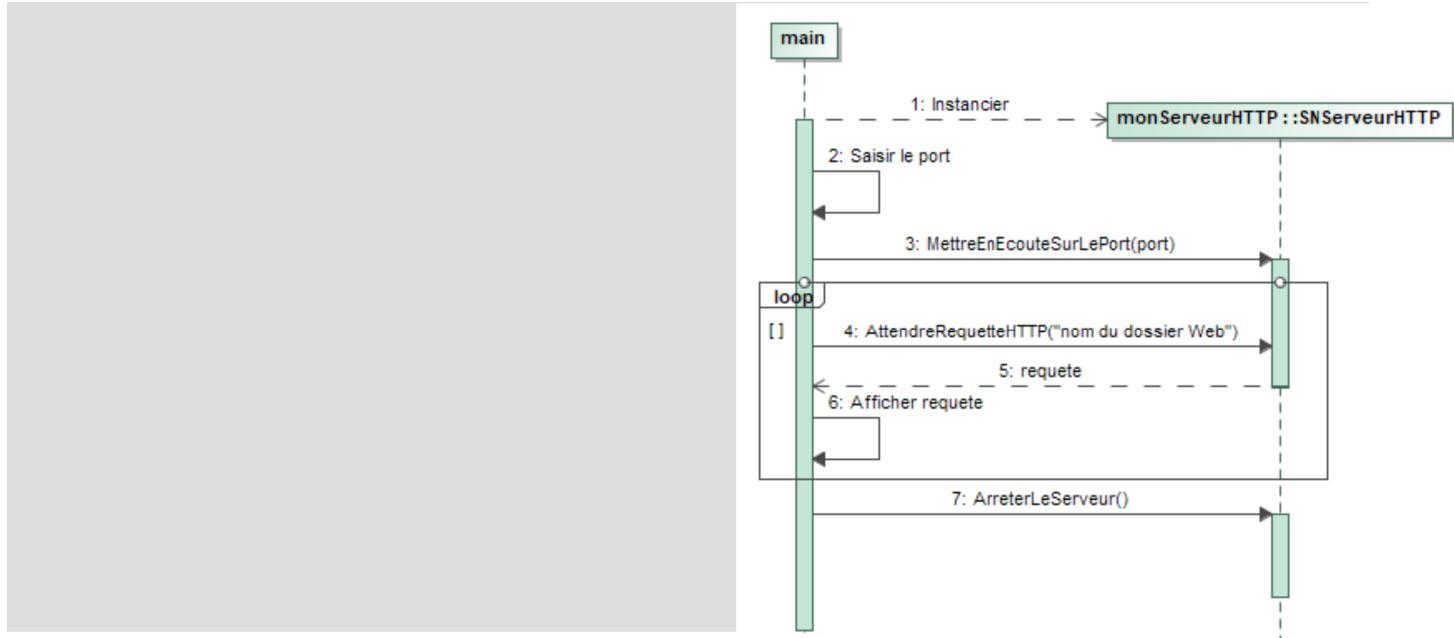
Dans le programme principal, après avoir sauvegarder l'échiquier en BMP, appeler le programme PositionBMP2JPG.exe à l'aide de la fonction WinExec (nécessite Windows.h). Utiliser l'aide de C++ builder ou bien Internet pour connaître le prototype de WinExec. Tester votre programme : une image Position.jpg doit apparaître dans le répertoire de compilation.

TP – Défi 7 – Serveur HTTP

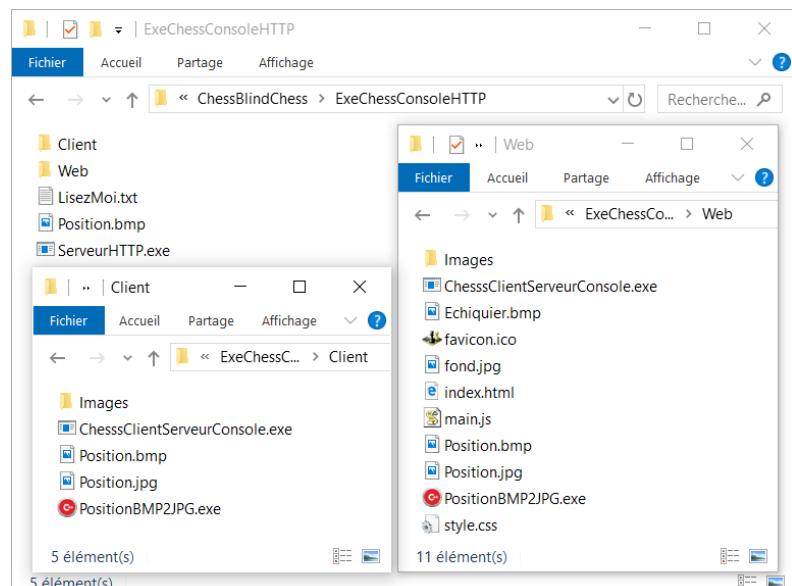
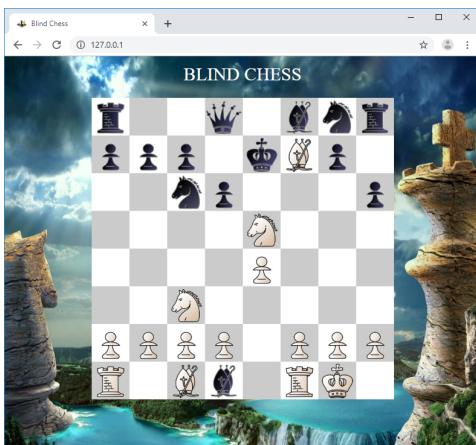
Pour coder et tester le serveur Web en C++, il est nécessaire de créer un nouveau projet en mode console. La classe SNServeurHTTP est donnée. Un exemple de page web minimaliste est proposé dans l'archive Web.zip. Cet exemple est à personnaliser.



Traduire en C++ le diagramme de séquence suivant (le « dossier Web » est un dossier à créer dans le répertoire de compilation) :



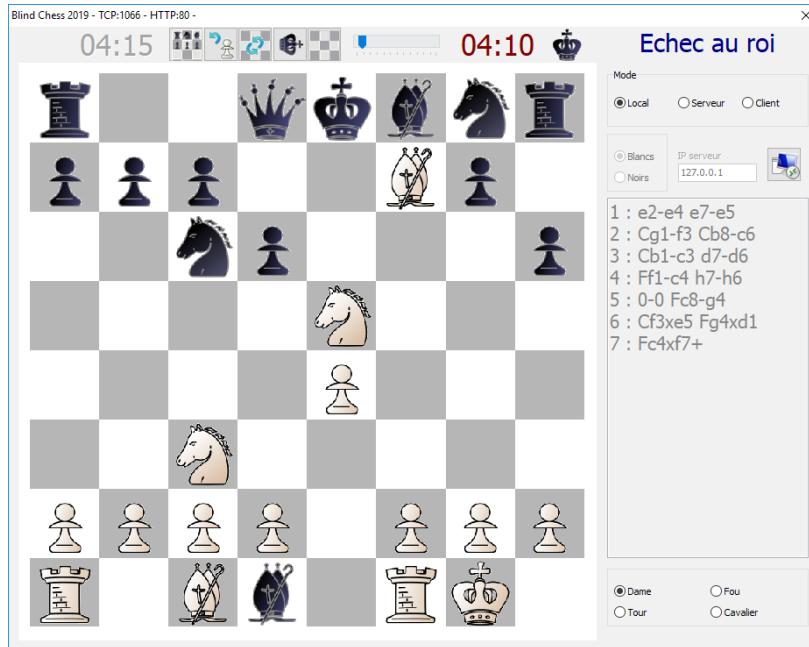
Coder et tester votre programme : il faut utiliser un navigateur Web et se connecter à 127.0.0.1 (serveur Web local).



Tester le jeu complet en plaçant l'exécutable du jeu en mode console ainsi que PositionBMP2JPG.exe dans le répertoire contenant la page Web. Lancer le serveur Web seul, lancer l'exécutable du jeu, connecter un navigateur au serveur Web : la partie peut commencer ! Remarque : pour tester le jeu en client-serveur sur une même machine, il est nécessaire de dupliquer le répertoire contenant l'exécutable du jeu.

TP – Défi 8 – Interface graphique et gestion du temps

Nous allons maintenant créer une nouvelle application graphique sous C++ Builder. Puis y intégrer nos classes créées en mode console.



Création de l'IHM et déplacement des pièces (Drag & Drop) :



Programmer sous C++ Builder l'IHM permettant de déplacer une pièce d'une case vers une autre. La méthode LacherPiece de la classe TForm1 est la méthode centrale du déplacement (DRAG and DROP) : elle doit donc être codée en partie. Placer les 64 images (Timage) dans le bon ordre dans la forme (en commençant en haut à gauche par Image1), et ajouter l'attribut TImage *laCase[8][8] à TForm1. Dans le constructeur, placer les adresses des 64 images dans le tableau laCase :

```
laCase[0][0]=Image1; laCase[0][1]=Image2; laCase[0][2]=Image3; laCase[0][3]=Image4;
laCase[0][4]=Image5; laCase[0][5]=Image6; laCase[0][6]=Image7; laCase[0][7]=Image8;
laCase[1][0]=Image9; laCase[1][1]=Image10; laCase[1][2]=Image11; laCase[1][3]=Image12;
laCase[1][4]=Image13; laCase[1][5]=Image14; laCase[1][6]=Image15; laCase[1][7]=Image16;
laCase[2][0]=Image17; laCase[2][1]=Image18; laCase[2][2]=Image19; laCase[2][3]=Image20;
laCase[2][4]=Image21; laCase[2][5]=Image22; laCase[2][6]=Image23; laCase[2][7]=Image24;
laCase[3][0]=Image25; laCase[3][1]=Image26; laCase[3][2]=Image27; laCase[3][3]=Image28;
laCase[3][4]=Image29; laCase[3][5]=Image30; laCase[3][6]=Image31; laCase[3][7]=Image32;
laCase[4][0]=Image33; laCase[4][1]=Image34; laCase[4][2]=Image35; laCase[4][3]=Image36;
laCase[4][4]=Image37; laCase[4][5]=Image38; laCase[4][6]=Image39; laCase[4][7]=Image40;
laCase[5][0]=Image41; laCase[5][1]=Image42; laCase[5][2]=Image43; laCase[5][3]=Image44;
laCase[5][4]=Image45; laCase[5][5]=Image46; laCase[5][6]=Image47; laCase[5][7]=Image48;
laCase[6][0]=Image49; laCase[6][1]=Image50; laCase[6][2]=Image51; laCase[6][3]=Image52;
laCase[6][4]=Image53; laCase[6][5]=Image54; laCase[6][6]=Image55; laCase[6][7]=Image56;
laCase[7][0]=Image57; laCase[7][1]=Image58; laCase[7][2]=Image59; laCase[7][3]=Image60;
laCase[7][4]=Image61; laCase[7][5]=Image62; laCase[7][6]=Image63; laCase[7][7]=Image64;
```

Deux événements (générés par les 64 TImages) gèrent le DRAG and DROP : ils nous permettront de déplacer une image sur une autre : OnDragOver et OnDragDrop.

Sélectionner les 64 TImages, dans les propriétés préciser DragMode=dmAutomatic.

Dans l'onglet événements : double cliquer dans OnDragOver, puis renommer l'événement en AccepterPiece.

`void __fastcall AccepterPiece(TObject *Sender, TObject *Source, int X, int Y, TDragState State, bool &Accept);`
est alors l'événement (OnDragOver) généré lorsqu'un objet image arrive sur un autre objet image, il faut l'accepter.

Ajouter `Accept=1;` dans le corps de la méthode.

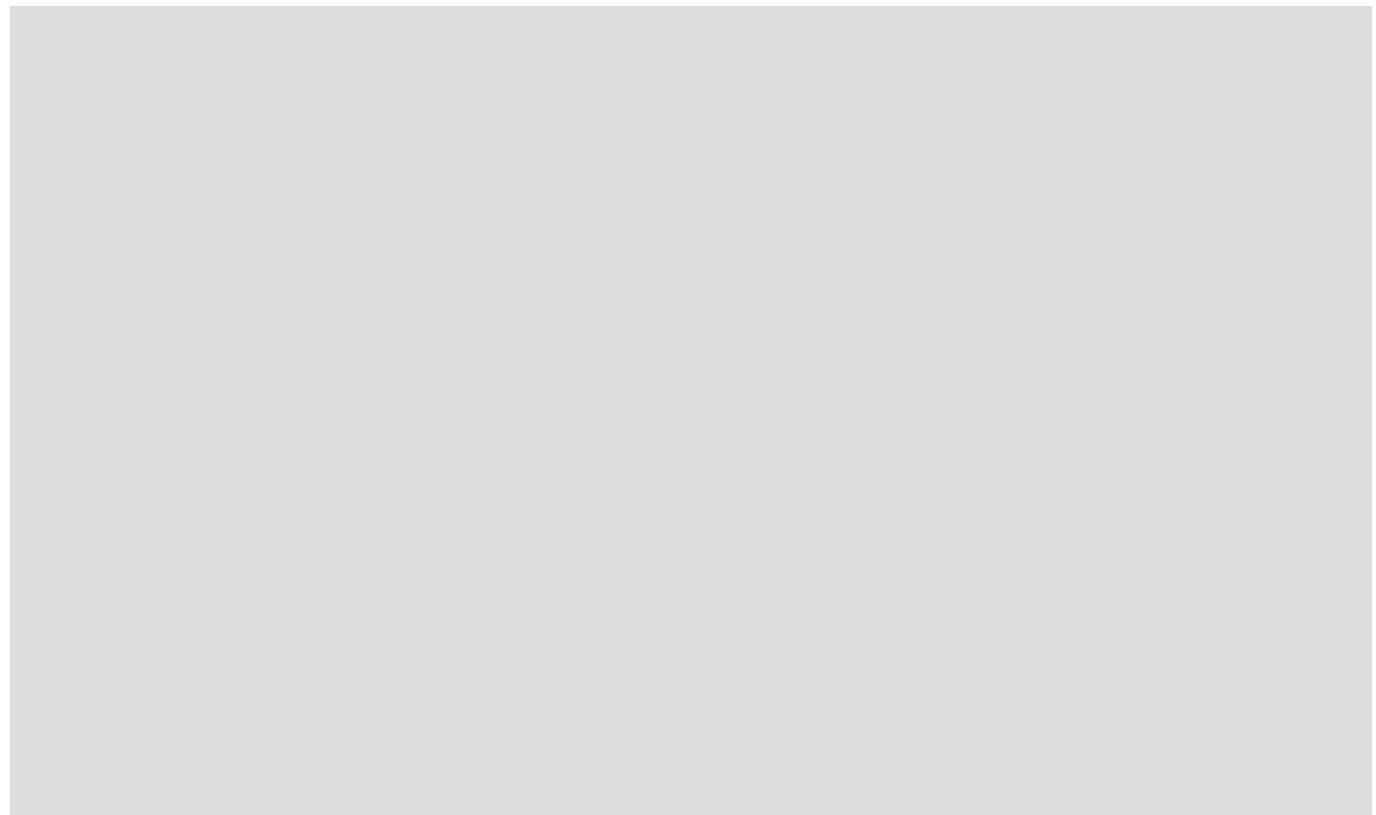
Dans l'onglet événements : double cliquer dans OnDragDrop, puis renommer l'événement en LacherPiece.

`void __fastcall LacherPiece(TObject *Sender, TObject *Source, int X, int Y);`

est alors l'événement (OnDragDrop) généré lorsqu'un objet image quitte son emplacement et arrive sur un autre objet image. En comparant les adresses source (Source) et destination (Sender) avec celles contenues dans le tableau 2D laCase on retrouve les coordonnées de départ et d'arrivée (idep jdep et iarr jarr) : i pour les lignes et j pour les colonnes. Source et Sender doivent être différents. L'algorithme est le suivant :

Si **Source** et **Sender** sont différents Alors
Pour **i** allant de 0 à 7 par incrément de 1
 Pour **j** allant de 0 à 7 par incrément de 1
 Si **Source** = **laCase**[**i**][**j**] Alors
 idep ← **i**
 jdep ← **j**
 FinSi
 ... même principe avec **Sender**, **iarr** et **jarr** ...
 FinPour
FinPour
FinSi

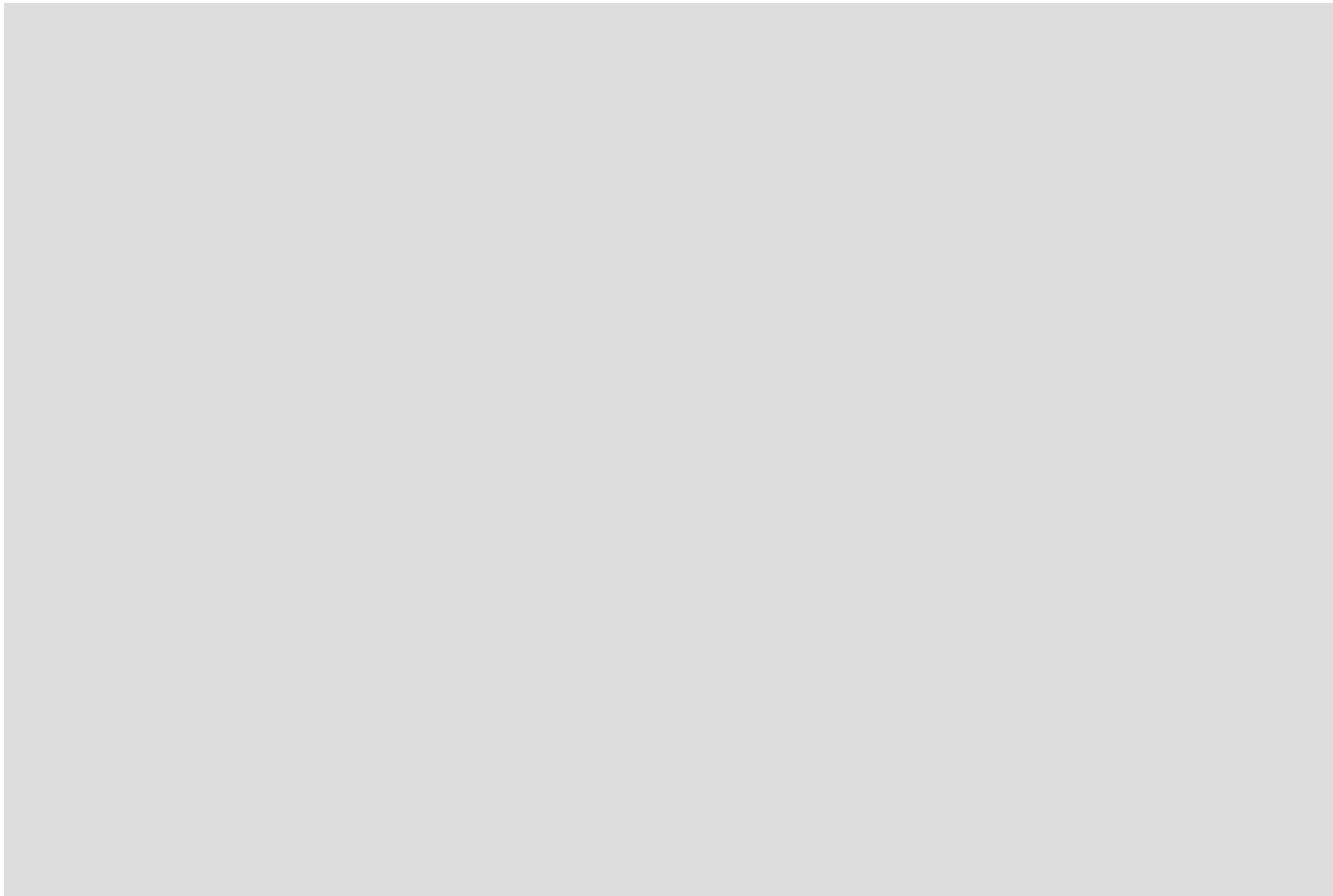
Donner le code de la méthode LacherPiece :



Ajouter un objet monEchiquier de la classe Echiquier à la classe TForm1. Tester la compilation du code.

Chaque mouvement sera d'abord effectué dans l'objet monEchiquier, puis chargé dans l'IHM à l'aide de la méthode MiseAJourEchiquierIHM de la classe TForm1. La méthode MiseAJourEchiquierIHM : Pour chaque case de l'échiquier, en fonction de la pièce lue dans monEchiquier (LireCaseEchiquier(i,j)), la bonne pièce sera chargée dans l'image laCase[i][j] (laCase[i][j] → Picture → LoadFromFile(...)), parmi les images cavB.bmp, cavN.bmp....

Donner le code de la méthode MiseAJourEchiquierIHM :



 Dans LacherPiece, appeler et tester la méthode Deplacer avec l'objet monEchiquier, en cas de promotion un groupe de RadioButton permettra de choisir la pièce de promotion et de passer la caractère correspondant à la méthode Deplacer. Si le déplacement est correct appeler la méthode MiseAJourEchiquierIHM. Tester le programme : l'image se déplace...

 Ajouter un indicateur (Shape ou Label) dans l'IHM permettant de savoir qui a le trait.

 Ajouter et coder un bouton "Nouvelle partie". Ajouter le bouton « Aveugle », les pièces ne sont plus visibles, puis un nouvel appui permet de revoir les pièces.

Gestion du temps :

-  Coder les méthodes et attributs nécessaires dans la classe Joueur. Un objet TTimer ajouté à la Form1 sera chargé de décrémenter le temps des joueurs lorsqu'ils auront le trait, des labels permettront d'afficher le temps restant de chaque joueur.

Jeu en réseau :

-  Créer un Thread : Fichier > Nouveau > Autre > Fichiers C++ Builder > Objet thread. Ce Thread (voir le diagramme de classes) contiendra une partie du programme principal du jeu en mode console : ce thread sera chargé d'attendre le coup de l'adversaire. Dans la fenêtre principale, ajouter la possibilité de saisir l'IP du serveur ainsi que le choix entre : partie locale, serveur ou client.

Création de l'image de la position :

-  Dans le diagramme de classe, threadPosition permet de générer l'image de la position. Créer et coder ce thread en y plaçant la création de l'image BMP. Le code suivant permet de générer un JPG sans utiliser le programme PositionBMP2JPG.exe. Il nécessite d'inclure jpeg.hpp. Ici Image71 contiendra l'image de la position.

```
Form1->LEchiquier()->SauvegarderEchiquierBMP("Web\\Position.bmp");
TJPEGImage *imagejpg = new TJPEGImage;
Form1->Image71->Picture->LoadFromFile("Web\\Position.bmp");
imagejpg->Assign(Form1->Image71->Picture->Bitmap);
imagejpg->SaveToFile("Web\\Position.jpg");
```

Serveur Web :

-  Créer un Thread (voir le diagramme de classes) chargé de gérer la communication entre le navigateur et le serveur Web. Il contiendra la main du serveur Web en mode console.

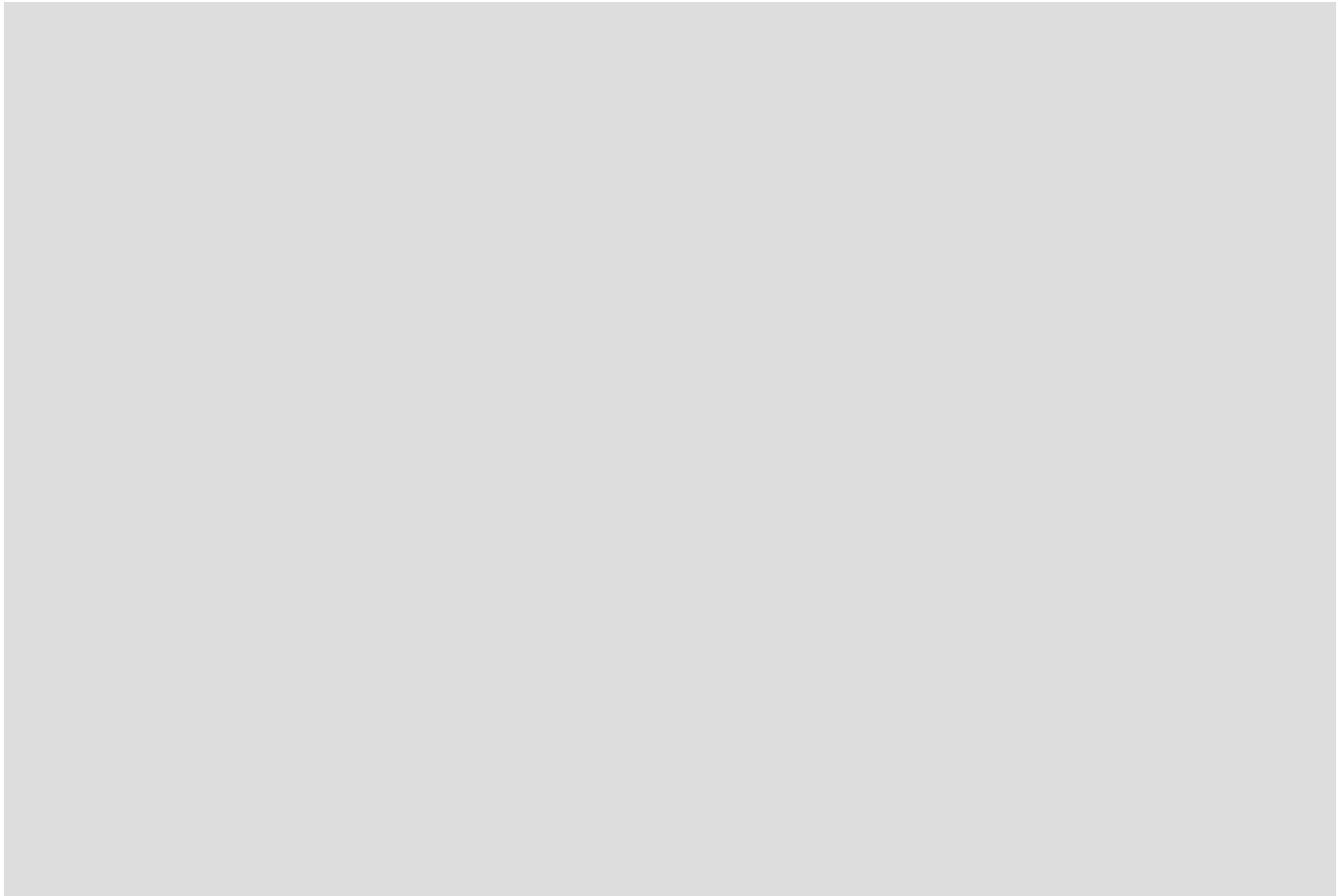
Mémorisation de la partie : la classe Partie

-  Ajouter la possibilité de reprendre son coup en créant une classe Partie (voir le diagramme de classes). Le principe est de stocker à chaque demi-coup l'objet monEchiquier dans un tableau d'Echiquiers : suiteEchiquier. Tester sur PC vos méthodes en ajoutant un objet maPartie de la classe Partie à la classe TForm1. Ajouter un bouton "retour" permettant de revenir au coup précédent.

Rotation de l'échiquier



Donner le diagramme d'activité permettant de tourner de 180° le tableau à 2 dimensions laCase (il faudra ajouter une image temporaire : `TImage *temp=new TImage(this);`).



Coder cette rotation et ajouter un bouton permettant de tourner l'échiquier.

Pour aller plus loin...

Affichage des coups dans l'interface...

Gestion des parties nulles...

Gestion des cadences Fisher...

Entrer une position libre...

Envoyer des commentaires à l'adversaire (Chat)...

Affichage des coups sur un journal lumineux...

Affichage de la position sur écran géant...

Annexe 1 : les règles du jeu

I – L’ÉCHIQUIER



■ But du jeu :

Faire **échec et mat**, (<— définition et tableaux de mats) évidemment, l’adversaire peut abandonner et vous avez la possibilité de gagner au temps.

On dit que le **Roi est en échec**, lorsque la case qu'il occupe est contrôlée par une pièce adverse. (en d'autres termes, lorsqu'une pièce peut le manger). → le Roi doit donc OBLIGATOIREMENT parer cet échec.

Si le Roi ne peut parer l'échec, il perd la partie, puisqu'il est **échec et mat**.

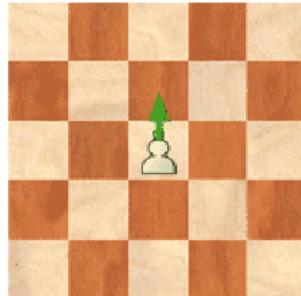
→ Caractéristiques de l'échec et mat :

- Le Roi ne peut plus se déplacer
- Aucune pièce alliée ne peut s'interposer pour parer l'échec
- La pièce qui fait l'échec ne peut être éliminée

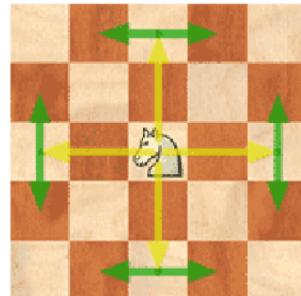
■ Voici la liste des cas qui déterminent le match nul :

- Il y a **Pat** (le Roi n'est pas en échecs, mais aucune pièce ne peut être déplacée)
- Il n'y a pas assez de matériel pour mater les Rois
- Un joueur fait des **échecs perpétuels** (échec à l'infini)
- La même position est répétée trois fois
- **50 coups sont joués sans aucune prise, ni aucun déplacement de pion.**

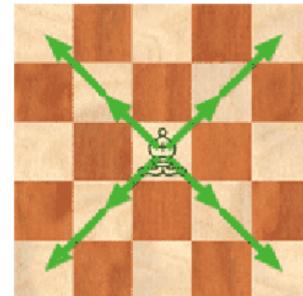
II- LA MARCHE DES PIÈCES



— Le pion (1)



— Le Cavalier (3)



— Le Fou (3)

- Avance droit devant lui
- Ne recule jamais
- Avance d'1 case à la fois SAUF lors de son 1er coup où il peut avancer de 2 cases
- **Capture en diagonale**

- Déplacement en "L"
- Change de couleur de case à chaque déplacement
- Saute par dessus les autres

- Déplacement en diagonale
- d'autant de cases qu'il veut



— La Tour (5)



— Le Roi



— La Dame (9)

- Déplacement horizontal/vertical
- d'autant de cases qu'elle veut
- Peut avancer/reculer

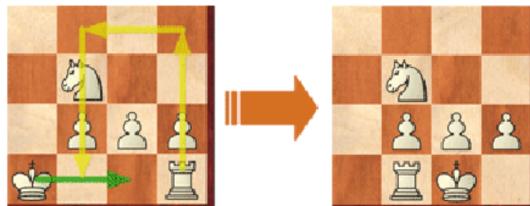
- Déplacement toutes directions
- d'1 seule case à la fois
- Il doit y avoir au moins une case entre 2 Rois adverses

- Pièce la plus puissante
- Cumule déplacement de la Tour (horizontal/vertical) et du Fou (diagonal)
- Peut contrôler 27 cases !

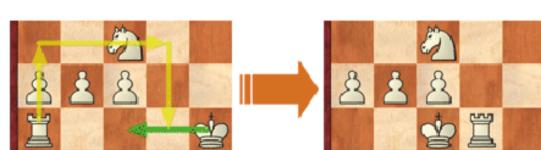
III – RÈGLES PARTICULIÈRES

LE ROQUE

- Le **Roi** se décale de deux cases, en direction d'une de ses Tours. – La **Tour** vient sauter par dessus le Roi pour se placer juste à côté (sur une case adjacente)



Petit Roque O-O



Grand Roque O-O-O

Les conditions du Roque :

- Ni le **Roi**, ni la **Tour** concernés, ne doivent avoir bougé pendant le jeu et aucune pièce ne doit les séparer.
- Le **Roi** ne peut être en échec.
- Aucune pièce ennemie ne doit contrôler les deux cases que le Roi parcourt pour roquer.

LE PION

■ La Promotion :

Quand le pion atteint la dernière rangée de l'échiquier, il se **transforme** en une autre pièce de sa couleur (en général une Dame). Le pion ne peut se transformer en Roi.

■ La Prise en passant :



Un pion peut capturer un pion adverse (de colonne adjacente), si celui-ci saute deux cases, **comme s'il n'avait avancé que d'une case**. On dit que ce pion prend le pion ennemi "en passant".

Pour vous aider à visualiser la scène, imaginez deux chevaliers au galop, face à face, pratiquant la **joute équestre**. Les deux chevaliers se croisent, mais l'un des deux tombe (celui qui a sauté deux cases). Une prise en passant s'effectue **immédiatement** (le tour suivant, il est trop tard).

Annexe 2: diagrammes de classes

