Projet DMX 2018

Thomas Demont BTS SNIR

Version 1.0

La Salle Avignon

Frères des Écoles Chrétiennes

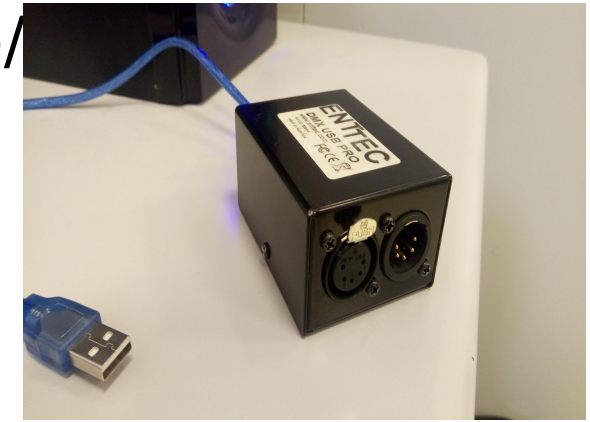
Expression du besoin

Créer une application permettant :

- Contrôle de projecteurs depuis un ordinateur
- Contrôle de projecteurs depuis une console d’éclairage

DMX 512

- Basée sur la norme RS485 (250kb/
- Permet de contrôler 512 canaux
- 32 projecteurs maximum
- Interfaces USB / DMX



Interfaces DMX - USB (Enttec)

Relier un ordinateur par son interface USB à une chaîne DMX

- DMX USB PRO (125 euros)
- OPEN DMX USB (55 euros)

Console Playback Wing (Enttec)

- 4 touches de contrôle, 10 *faders* et 40 boutons
- Communication réseau (non DMX)

Voir Spécification technique p 8-9

Protocole Wing

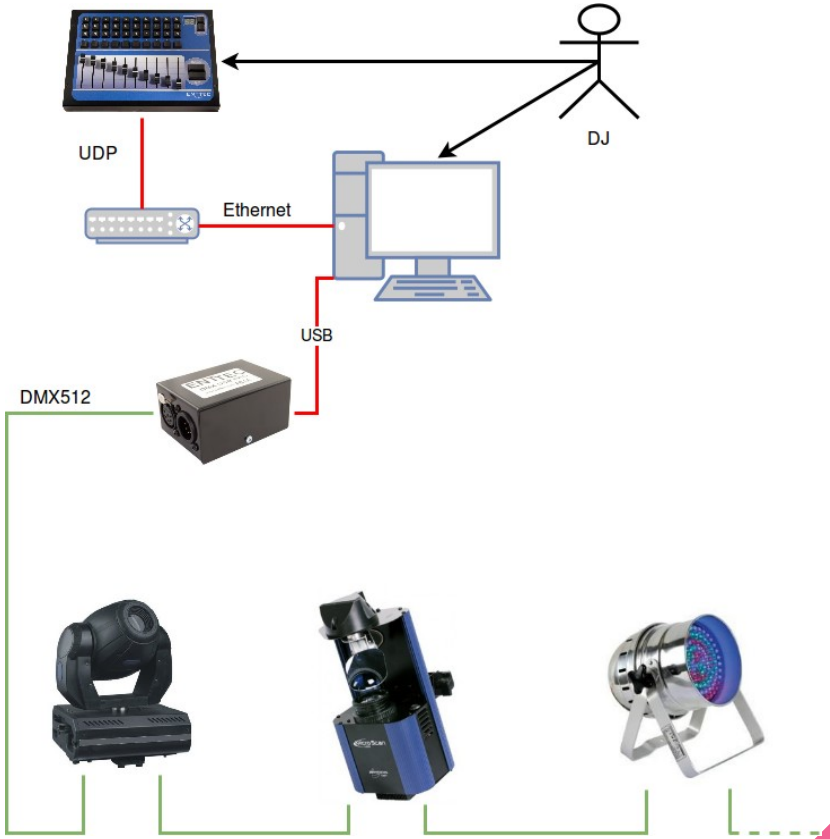
```
0000  ff ff ff ff ff ff 00 50  c2 07 59 5f 08 00 45 00   .......P ..Y_..E.
0010  00 38 0a d2 00 00 40 11  7a 67 c0 a8 34 d4 ff ff   .8....@. zg..4...
0020  ff ff 0d 02 0d 02 00 24  44 62 57 4f 44 44 10 01   .......$ DbWODD..
0030  ff ff ff ff ff ff 00 00  00 2f 00 00 00 00 00 00   ........ ./......
0040  00 00 00 00 00 00                                 ......
```

Message de type **WODD**, sortie classique, correspond à un changement de valeur pour le fader 1

- Communique en broadcast UDP.
- Port 3330.

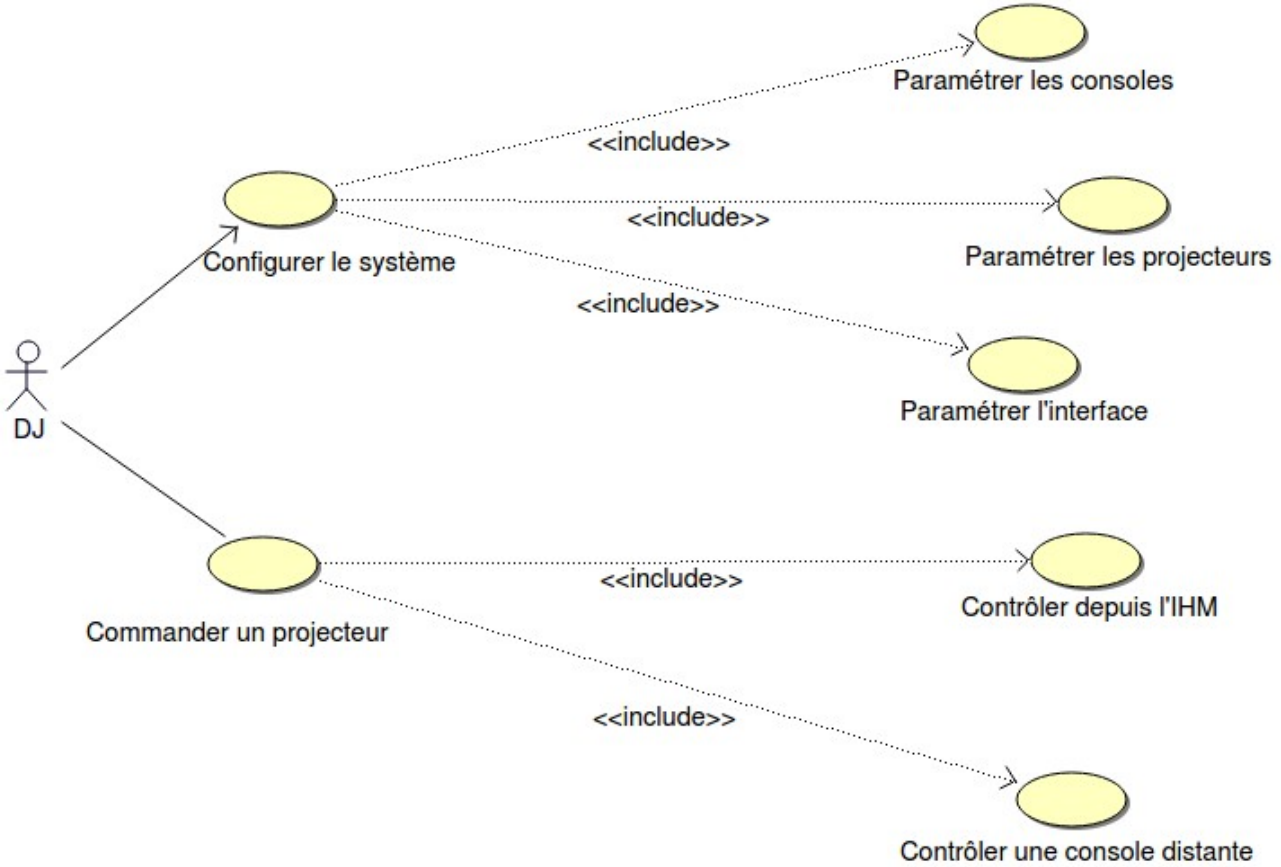
Synoptique

Voir architecture système p 17



Voir Analyse p 19

Analyse



Répartition des tâches

Équipe :

- IR : Tony Reynier, Thomas Demont
- EC : Enzo Guibbani

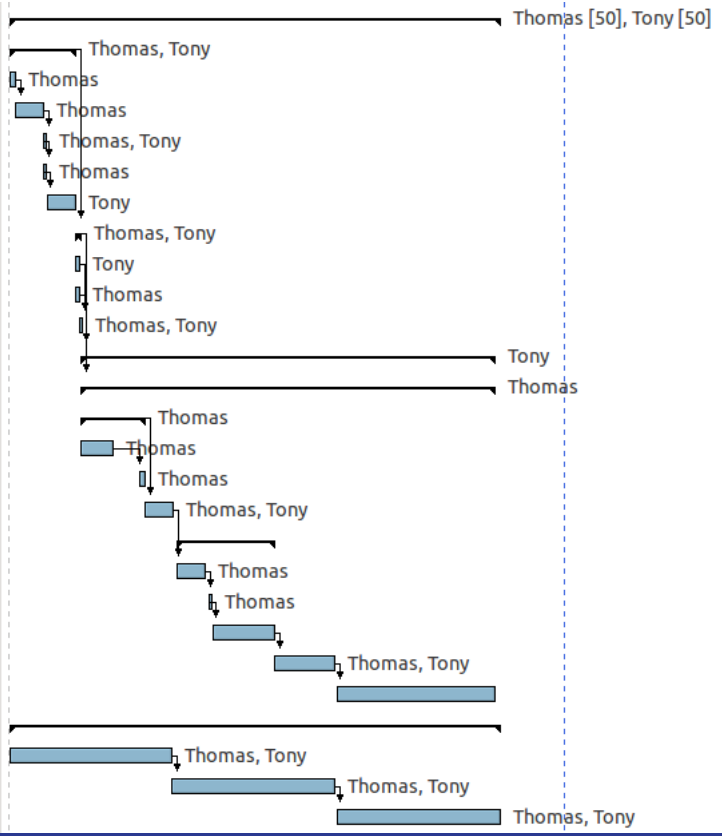
Mes tâches :

- Configurer le système
- Piloter les projecteurs

Planification

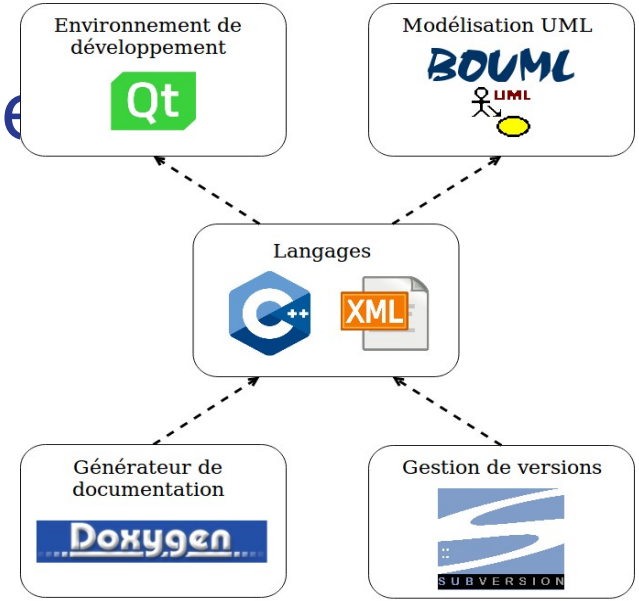
Voir Répartition des tâches p 14 - 15

1	**Projet DMX**	**155j 1h**
1.1	**Analyse**	**3j 6h**
1.1.1	établir tests de validation	1j
1.1.2	établir cas d'utilisation (UML)	1j
1.1.3	établir les objectifs	3h
1.1.4	répartir les tâches (Gantt)	3h 30min
1.1.5	prototyper l'IHM	1j
1.2	**Conception**	**2j 4h**
1.2.1	créer un diagramme de classe	1j
1.2.2	créer un diagramme de séquences	1j
1.2.3	prendre en main le materiel	4h
1.3	**Développement Partie IR (élève 3)**	**43j 4h**
1.4	**Développement Partie IR (élève 4)**	**49j 2h**
1.4.1	**Programmation du logiciel (0.8)**	**3j 1h**
1.4.1.1	établir la connexion entre console wing et pc	2j 1h
1.4.1.2	mettre en oeuvre DMX	1j
1.4.2	tests (0.8)	3j 2h
1.4.3	**Programmation du logiciel (0.9 et 1.0)**	**12j 5h**
1.4.3.1	mettre en oeuvre XML	1j 6h
1.4.3.2	renseigner les fichiers XML (consoles.xml, adaptateurs.xml)	7h
1.4.3.3	implémenter les options de choix d'interface DMX USB depuis le logiciel	10j
1.4.4	tests (0.9)	5j 2h
1.4.5	tests finaux	25j
1.5	**Rédaction de la documentation**	**56j**
1.5.1	rédiger le compte rendu de la revue 2	18j 1h
1.5.2	rédiger le compte rendu de la revue 3	18j 4h
1.5.3	rediger le compte rendu de la revue finale	19j 2h



Ressources de développeme

OS	GNU Linux (Ubuntu 12.04.5 LTS)
EDI	Qt Creator 2.4.1
UML	bouml 7.2
Version	subversion (client svn 1.6.17)
Documentation	Doxygen version 1.7.6.1



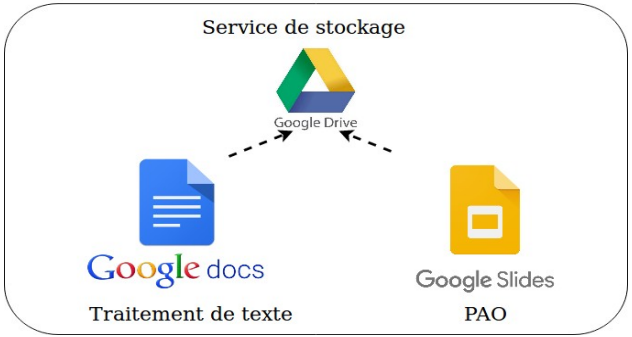
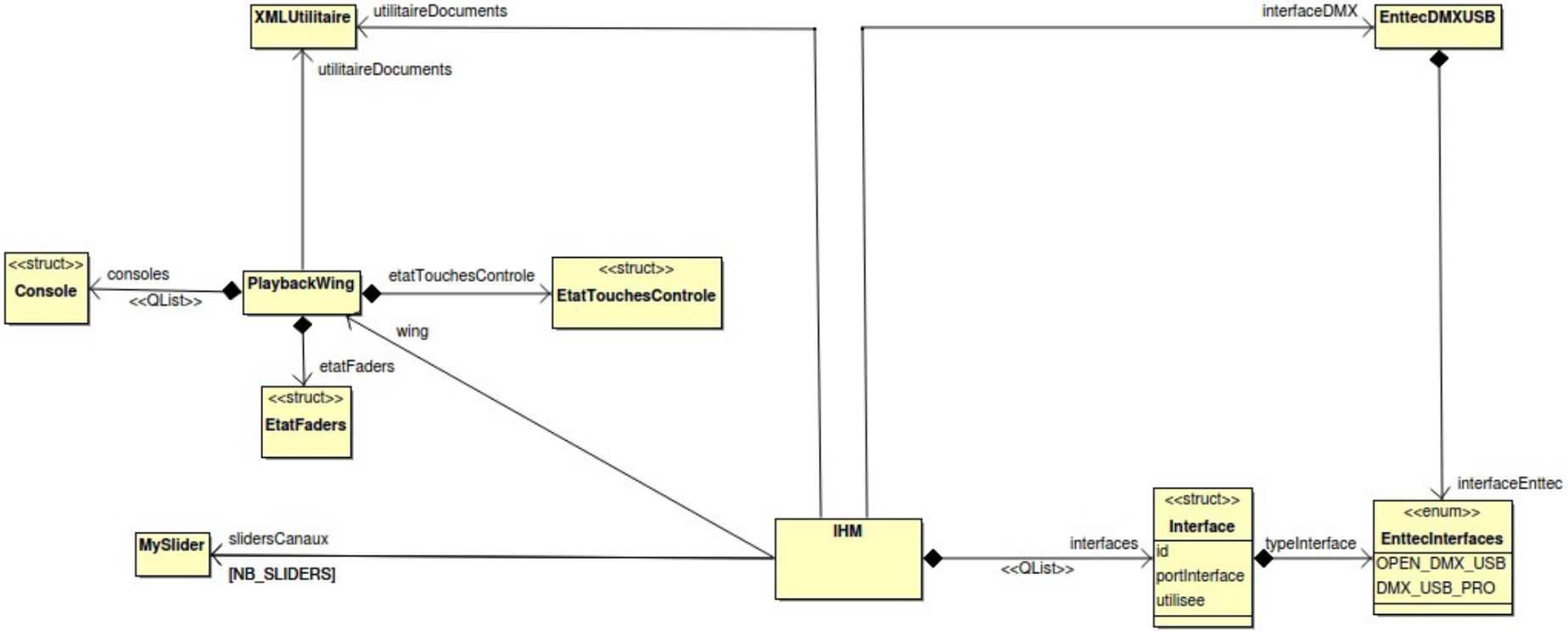


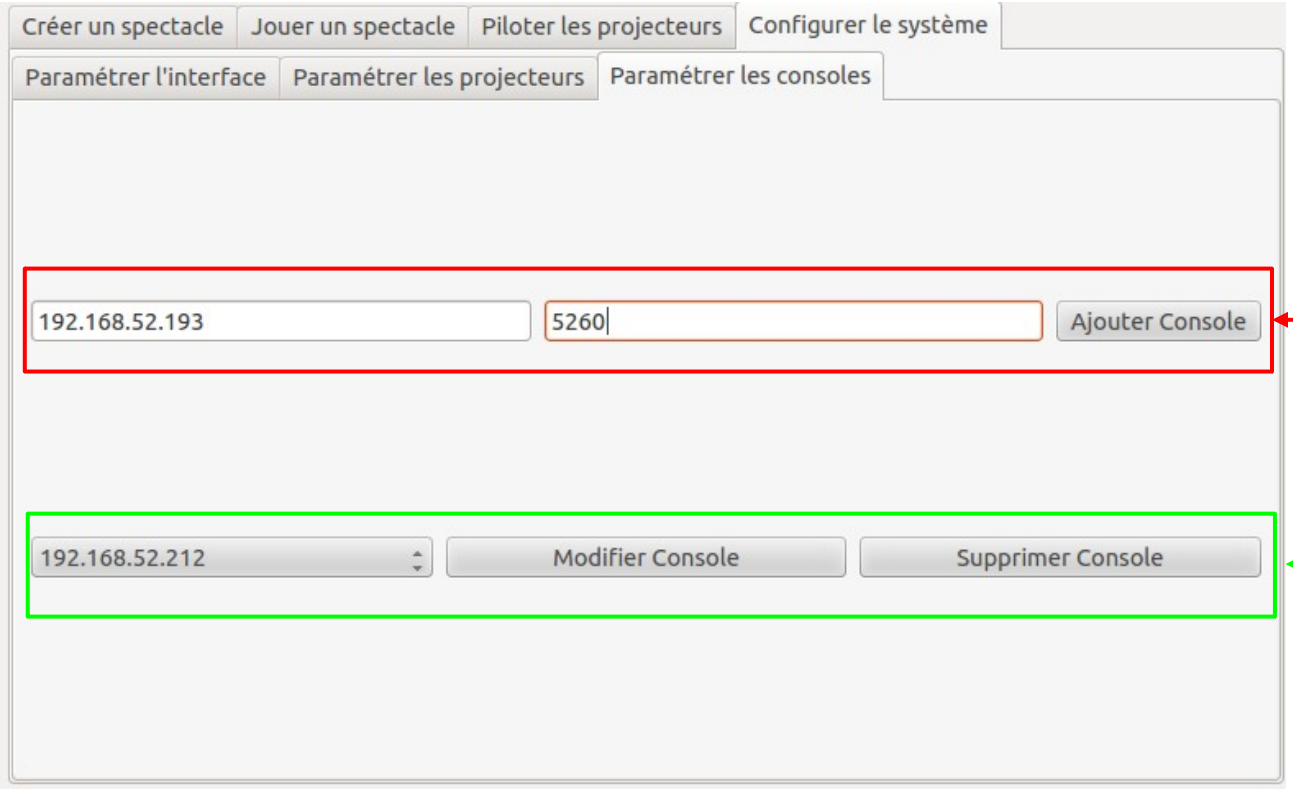
Diagramme de classes

Voir Architecture logicielle p 22



Utilisation du système

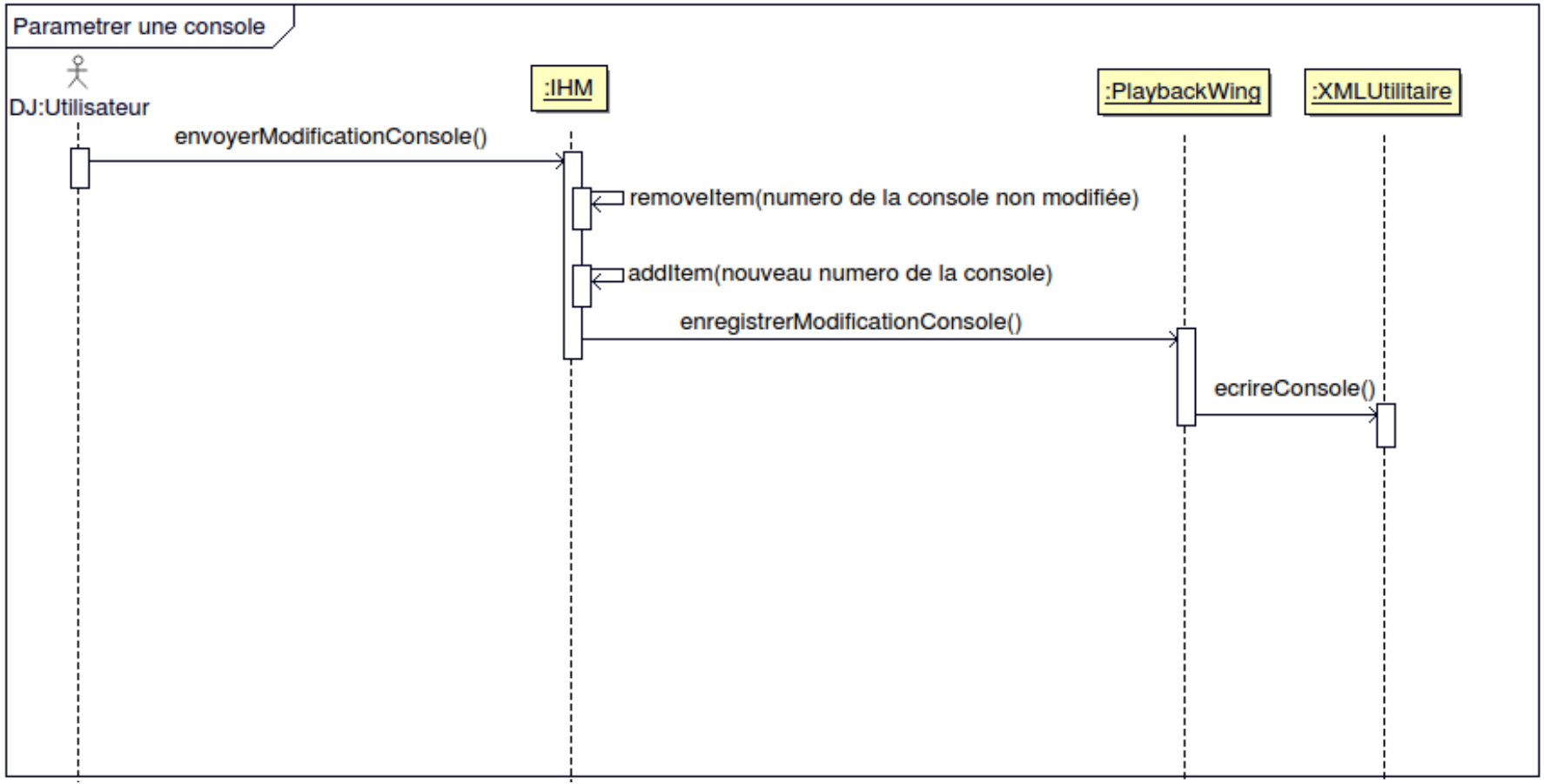
Page Paramétrer les consoles



Ajout de console

Modification / suppression de console

Voir p 30


Parametrer une console
DJ:Utilisateur
:IHM
:PlaybackWing
:XMLUtilitaire
envoyerModificationConsole()
removeItem(numero de la console non modifiée)
addItem(nouveau numero de la console)
enregistrerModificationConsole()
ecrireConsole()

XML

Voir IHM p 27 - 28

adaptateurs.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<interfaces>
  <interface utilisee="1" id="5">
    <port>/dev/ttyUSB0</port>
    <type>1</type>
  </interface>
  <interface utilisee="0" id="2">
    <port>/dev/ttyUSB1</port>
    <type>1</type>
  </interface>
</interfaces>
```

Prologue
Arbre des éléments
Attributs de l'élément interface
Element
Noeuds Enfants de interface

consoles.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<consoles>
  <console>
    <adresseIP>192.168.52.212</adresseIP>
    <port>3330</port>
  </console>
  <console>
    <adresseIP>192.168.52.213</adresseIP>
    <port>5260</port>
  </console>
</consoles>
```

Élément racine
Arbre des éléments
Exemple de donnée

Classe XMLUtilitaire

Module **QtXml** : ensemble de classes Qt pour la gestion de documents XML

Les classes Qt utilisées : **QDomDocument**, **QDomElement** et **QDomNode**

- **QDomDocument** : classe qui représente le document XML entier et fournit l'accès principal aux données du document
- **QDomElement** : classe qui représente un élément dans l'arbre du document
- **QDomNode** : classe de base pour tous les nœuds de l'arborescence du document

Classe XMLUtilitaire : principe

□ Utilisation de la classe **QDomDocument** :

- Définir le contenu entier du document XML → **setContent()**
- Créer des données de document → **createElement()** et **createTextNode()**
- Sauvegarder la représentation XML du document → **save()**
- Accéder à un élément du document → **documentElement()**

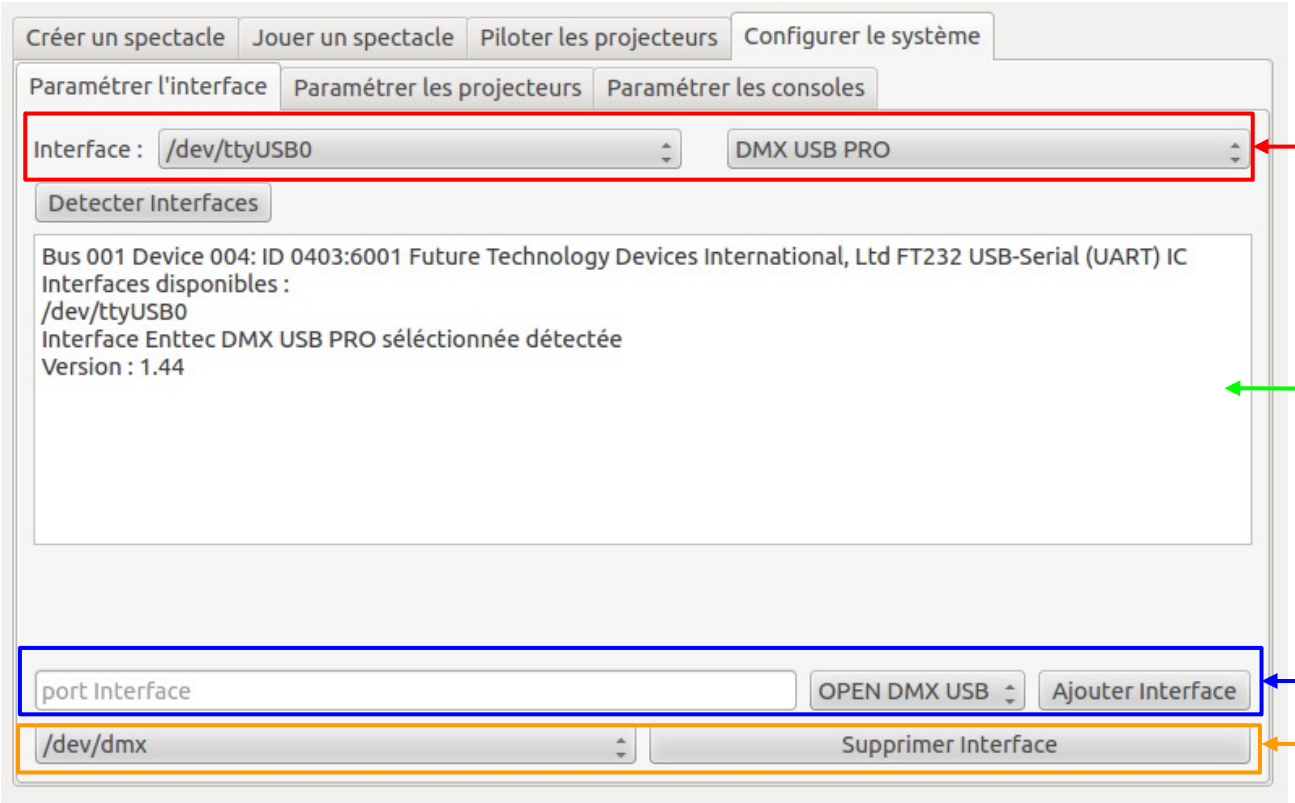
□ Utilisation de la classe **QDomElement** :

- Accéder à un attribut → **attribute()**
- Accéder au texte d'un élément → **text()**

□ Utilisation de la classe **QDomNode** :

- Parcourir les nœuds d'un document → **firstChild()** puis **nextSibling()** pour accéder au suivant
- Accéder aux élément d'un noeud avec **toElement()**

Page Paramétrer l’interface

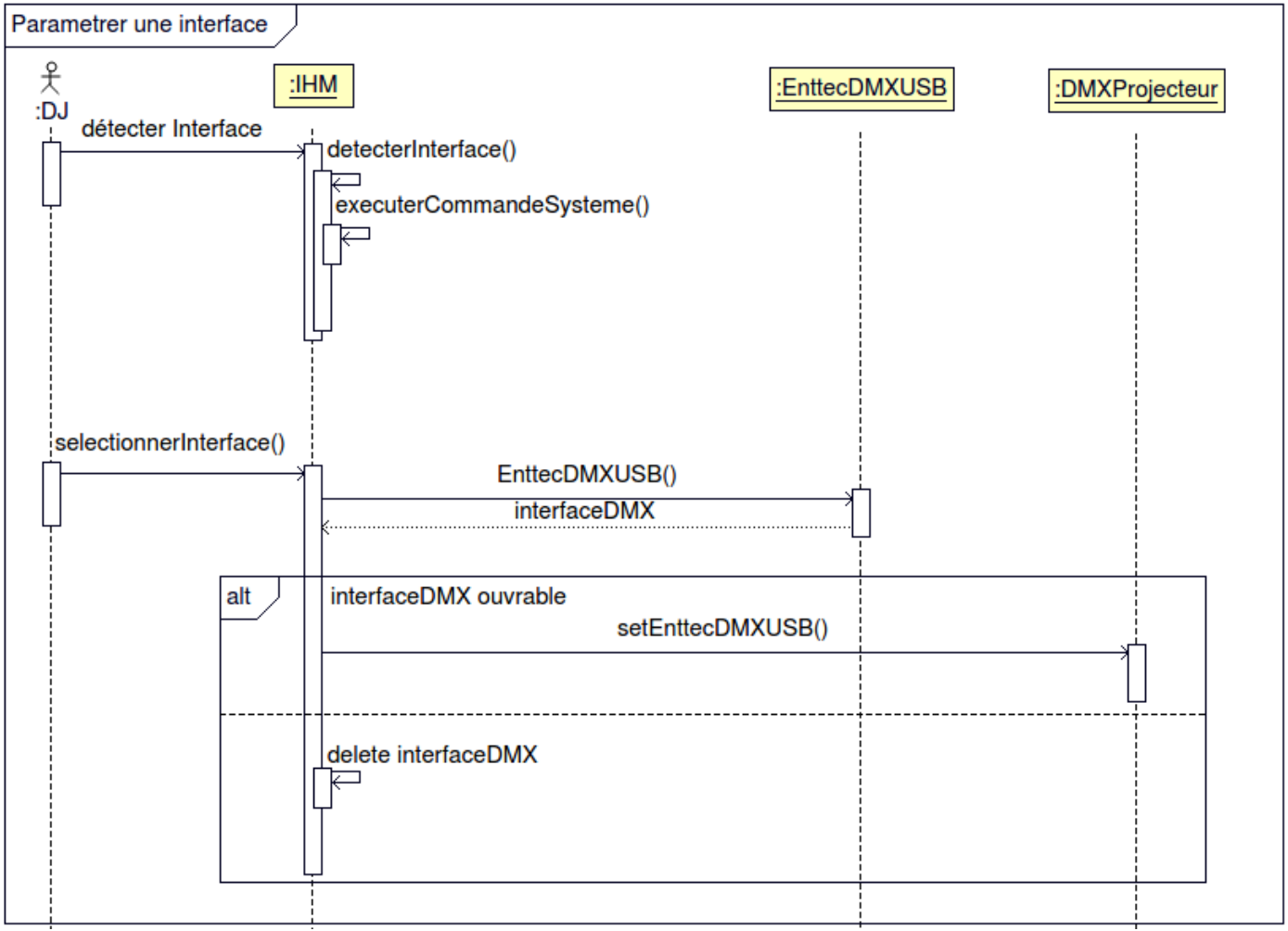


Choix de l’interface active

Résultat de la détection d’interface

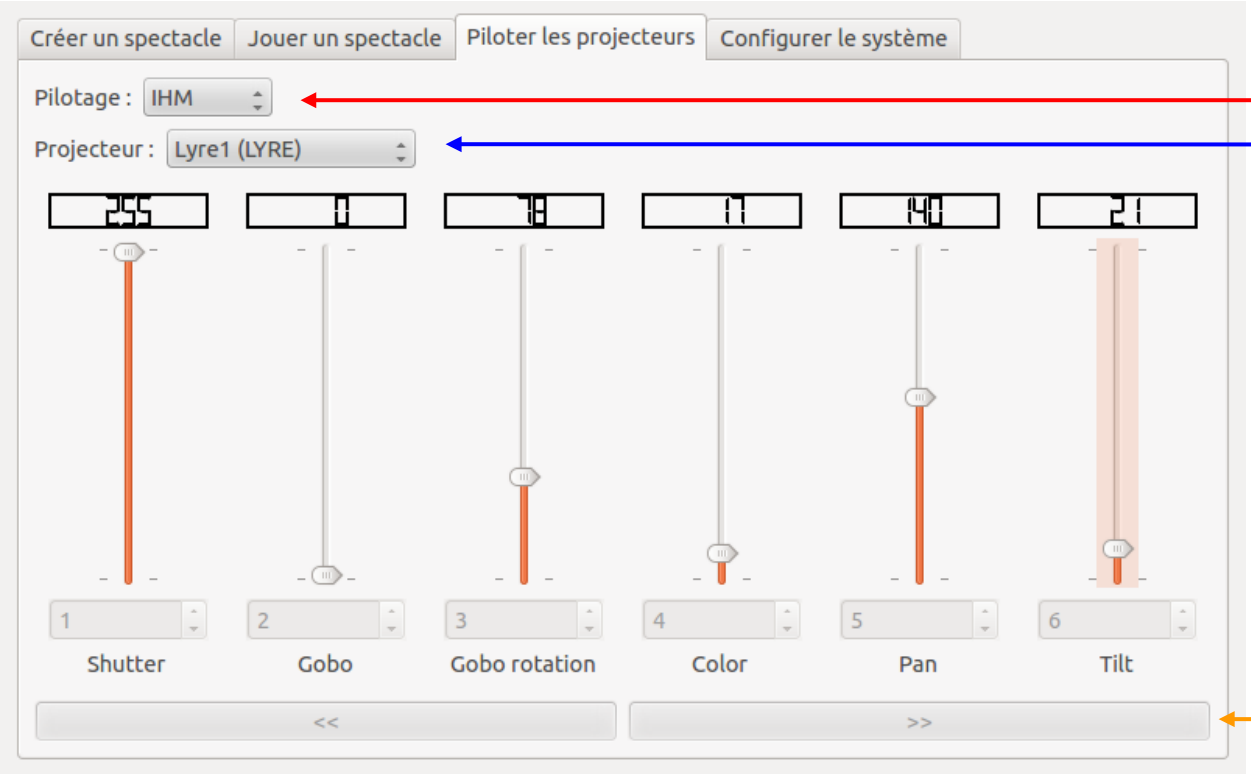
Ajout d’interface

Suppression d’interface

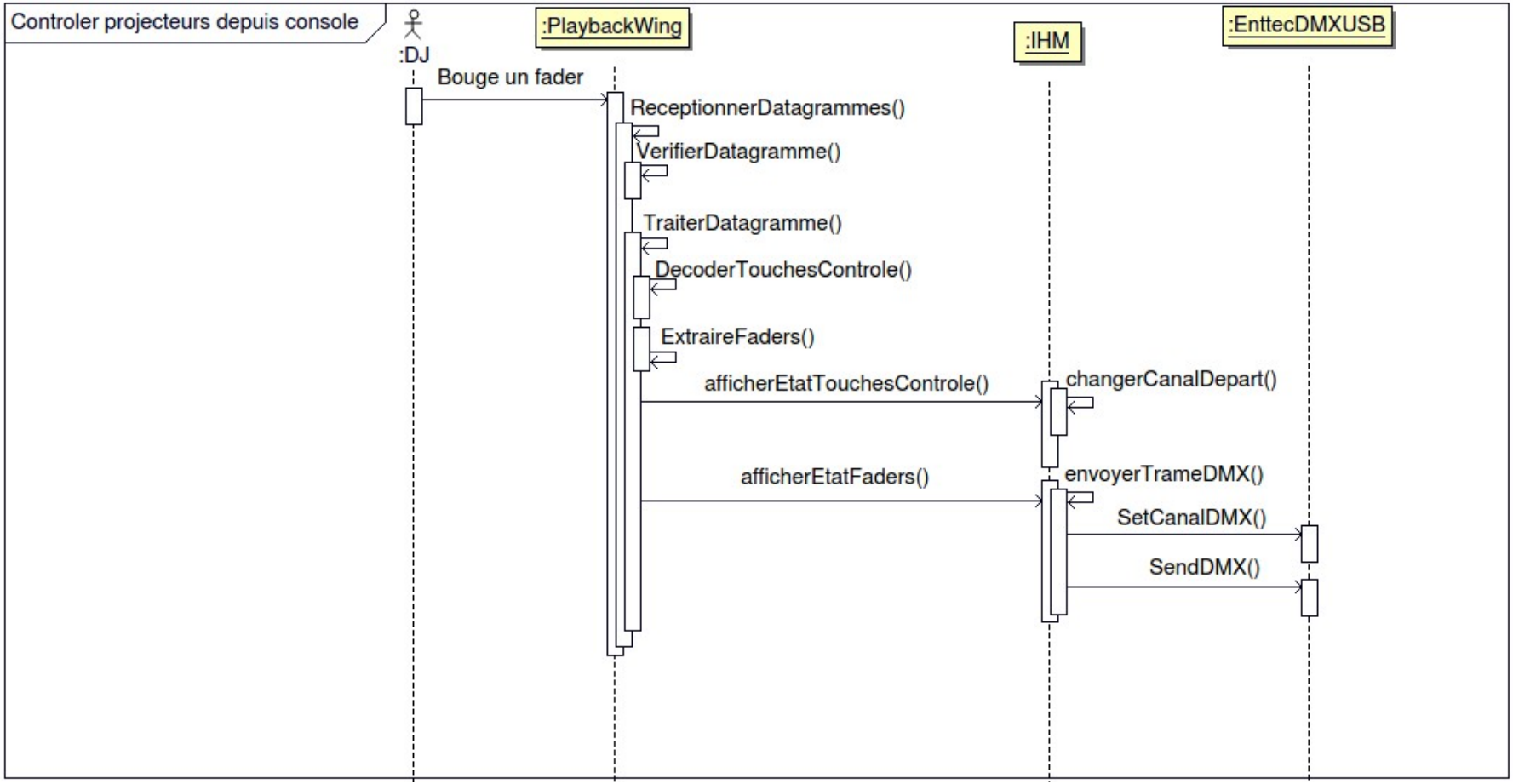

Parametrer une interface
:DJ
:IHM
:EnttecDMXUSB
:DMXProjecteur
détecter Interface
detecterInterface()
executerCommandeSysteme()
selectionnerInterface()
EnttecDMXUSB()
interfaceDMX
alt
interfaceDMX ouvrable
setEnttecDMXUSB()
delete interfaceDMX

Voir p 33

Piloter les projecteurs



Voir p 37


Controler projecteurs depuis console
:DJ
:PlaybackWing
:IHM
:EnttecDMXUSB
Bouge un fader
ReceptionnerDatagrammes()
VerifierDatagramme()
TraiterDatagramme()
DecoderTouchesControle()
ExtraireFaders()
afficherEtatTouchesControle()
changerCanalDepart()
afficherEtatFaders()
envoyerTrameDMX()
SetCanalDMX()
SendDMX()

Dialogue UDP avec Qt : classe **PlaybackWing**

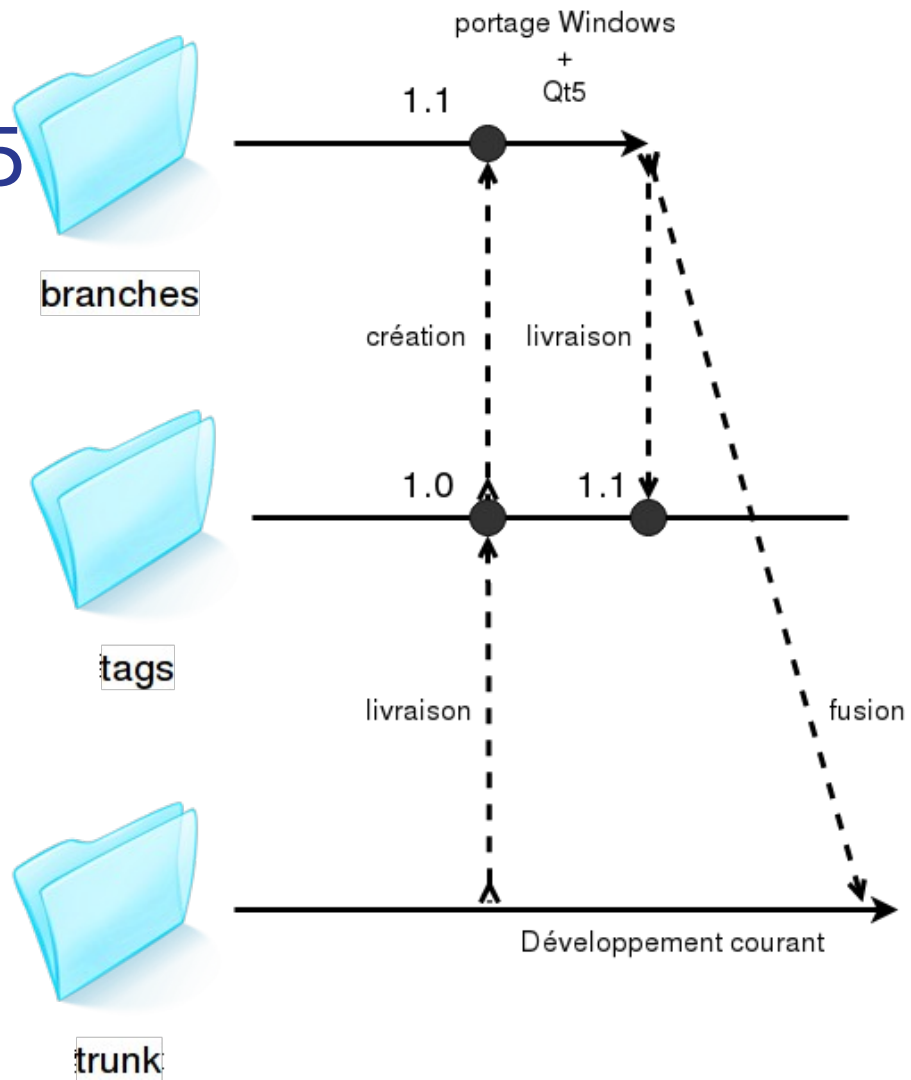
- UDP : Protocole de la couche transport en mode **non connecté**.
- Module **QtNetwork** : ensemble de classes permettant la programmation réseau sous Qt
- Classe **UdpSocket** : fournit un socket UDP
    - Lier localement le socket UDP à une adresse et un port en utilisant **bind()**→ **PlaybackWing()**
    - Connecter le signal **readyRead()** pour gérer l'arrivée des datagrammes → **ReceptionnerDatagrammes()**
    - Appeler **writeDatagram()** et **readDatagram()** pour transférer des données **envoyerCanalDepartWing()** et **ReceptionnerDatagrammes()**

Tests de validation

Désignation	*Résultat (Oui / Non)*
Ajouter et supprimer une console	O
Ajouter et supprimer une interface	O
Commander les éclairages depuis l'application	O
Commander les éclairages depuis une console distante	O

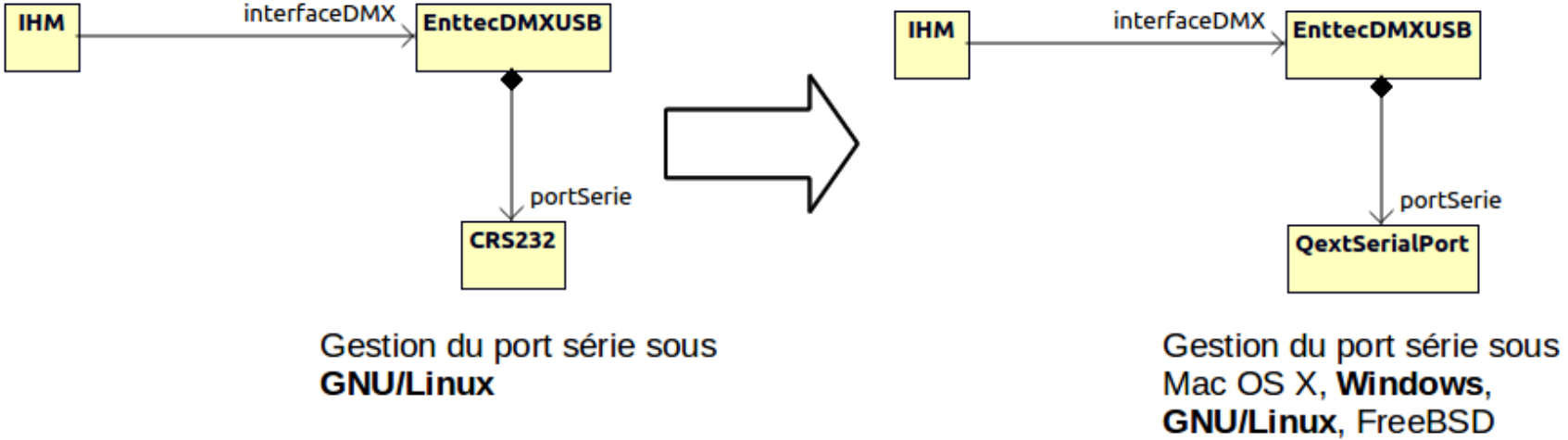
Portage Windows / QT5

Qt → bibliothèques multi-plateforme Linux, Windows ...



Modifications

- Portage Qt4 → Qt5 : changement des modules GUI (Qt 4 → QtGui ⟹ Qt 5 → QtWidgets)
- Portage Linux → Windows : remplacement de la classe **CRS232** (Linux) par **QextSerialPort** (Linux/Windows)



Gestion du port série sous **GNU/Linux**

Gestion du port série sous Mac OS X, **Windows**, **GNU/Linux**, FreeBSD

Conclusion

Élément à améliorer :

- Widgets plus intuitifs
- Portage Mac OS (à valider)

Éléments enrichissants lors de la réalisation du projet

- Communication UDP
- Informatique industrielle (DMX)
- Portage Windows + Qt 4.8 / 5