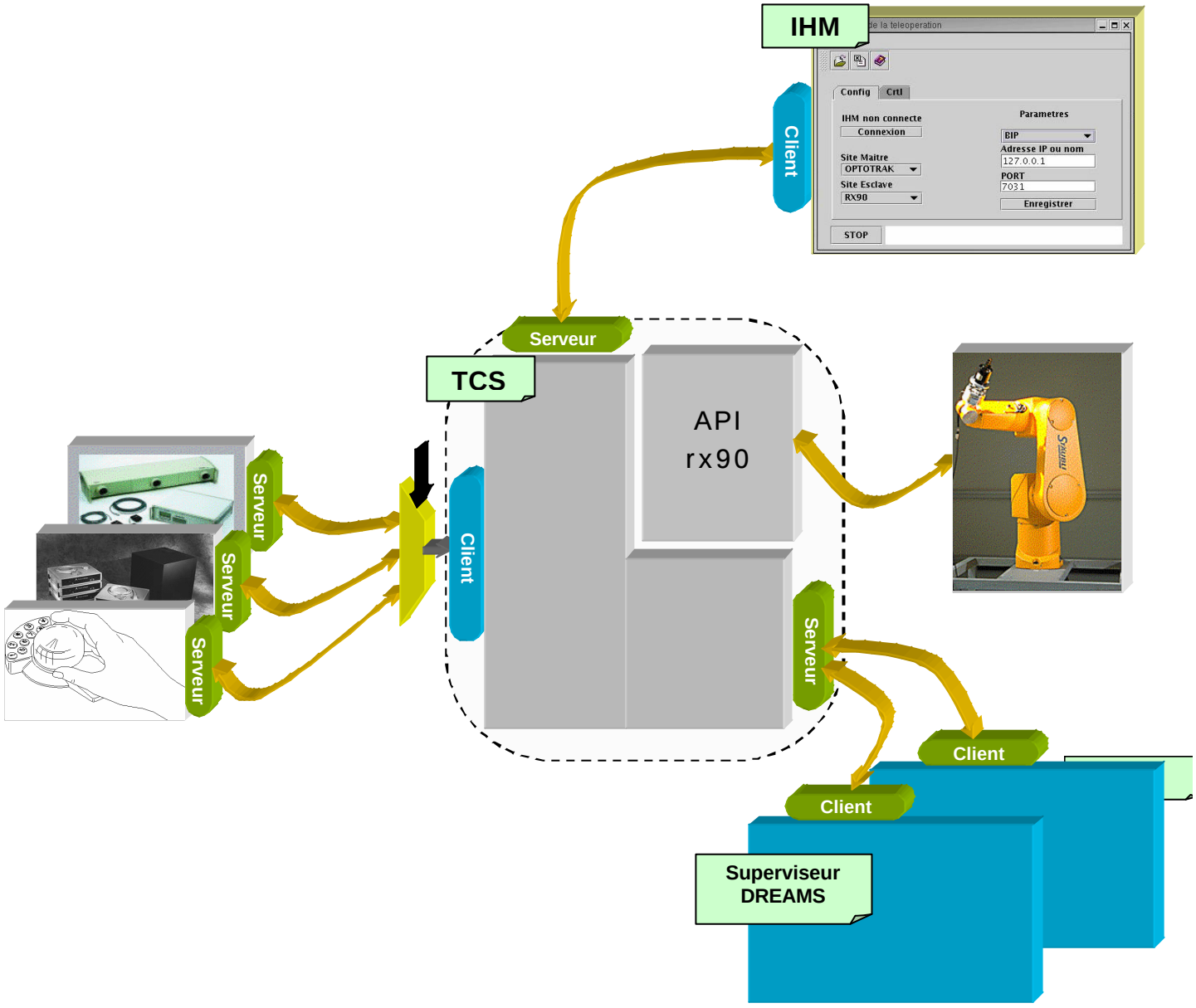
TELEOPERATION

El jalaoui Abdellah

I. Architecture et protocole de communication

I.1. Architecture générale de la spécification



I.2. Protocoles

I.2.1. Sites maîtres et démon

Les sites maîtres attendent que l'on se connecte sur eux.
Souris 3D : Port : *5003* AddIP : *satan*
Flock of Birds : Port: AddIP:
OPTOTRAK : Port: *5001* AddIP: *tabasco*
Après la connexion, les données sont envoyées sous le format suivant : "**temps x y z phi thêta psi \n**"
temps: durée en seconde
x y z : coordonnées en mètres
phi thêta psi : angles de roulis tangage lacet en radian

I.2.2. DREAMS et démon

DREAMS est client et se connecte sur le démon. PORT : 7032
L'addIP sur laquelle il se connecte est configurée dans le fichier **/Trasys/Dreams2/dreamsSys/config/cs/cs.ini** et vaut pour l'instant : **localhost**.
Après connexion, le démon envoi a DREAMS, les données de télémétrie sous cette forme :
"**TM TmPacket Telemetrie " +time ?1, ?2, ?3, ?4, ?5, ?6 + ";\n**" ou **time** est le temps écoulé depuis la première mesure.
Le nom "**Telemetrie"** ainsi que les caractéristiques de **?1, ?2, ?3, ?4, ?5, ?6**, leurs bornes et la disposition des graphiques correspondants sont configuré dans deux fichiers XML :
/Rx90/DreamsFiles/displays-src.xml
/Rx90/DreamsFiles/telemetry-src.xml
(voir Soraya)

I.2.3. IHM et démon

L'IHM est cliente, elle se connecte sur le démon.
Le protocole est le suivant :
1) "10 V1 V2 V3 V4……"
Si la chaîne commence par "10", les valeurs qui suivent servent a mettre a jour une structure globale du démon. (Éléments en gras).

```
typedef struct IHM_Param
{
 int IHM_Telem;
 int DREAMS_Telem;
 int SiteM;
 int RobotRX90;
 int IGRIP;
 int ABS_POS;
 int ABS_ORI;
 int Susp;
 int ArtCart;
 double CoefDem;
 double Retard;
 int RX90_Stat;
 int DREAMS_Stat;
 int IHM_Telem_Stat;
 int IGRIP_Stat;
 int Opto_Stat;
 int FoB_Stat;
 int Souris_Stat;
 int SockListen_Stat;
} IHM_Param;
```

2) "100 N Port AddIP"
Si la chaîne commence par 100, le nombre N qui suit correspond au numéro d'une plateforme selon :

```
#define Opto 0
#define FoB 1
#define Souris 2
#define Ihm 3
#define DREAMS 4
```

Port est le port de la plateforme et AddIP son adresse IP.

3) "1000 "
Si la chaîne commence par 1000, l'IHM demande au démon de lui transmettre les états des éléments de la téléopération.
Le démon lui envoie alors les valeurs des éléments de la structure suivante (en gras) :

```
typedef struct IHM_Param
{
  int IHM_Telem;
  int DREAMS_Telem;
  int SiteM;
  int RobotRX90;
  int IGRIP;
  int ABS_POS;
  int ABS_ORI;
  int Susp;
  int ArtCart;
  double CoefDem;
  double Retard;
  int RX90_Stat;
  int DREAMS_Stat;
  int IHM_Telem_Stat;
  int IGRIP_Stat;
  int Opto_Stat;
  int FoB_Stat;
  int Souris_Stat;
  int SockListen_Stat;
} IHM_Param;
```

Format de l'envoie : "V1 V2 V3 …..\n"

4) "10000 type V1 V2 V3 V4 V5 V6 tps\n"
Si la chaîne commence par **10000**, c'est un ordre de déplacement pour le robot envoyé par l'IHM.
Si type=**0**, alors les Vi sont les coordonnées articulaires : **?1, ?2, ?3, ?4, ?5, ?6**
Si type=**1**, alors les Vi sont des coordonnes de position et d'orientation : **x y z phi thêta psi**
tps est la durée pour faire le mouvement.

II. Implémentation

II.1. L'IHM

II.1.1. Sources

L'IHM est l'interface graphique faite en JAVA
Sources : **/Rx90/IHM/JAVA/**

II.1.2. Compiler et lancer

Se loguer sur akila, Lancer jbuilder (**/opt/JBuilder8/bin/jbuilder**)

II.2. Le contrôleur

II.2.1. Sources

Lancer Orccad : **orc_tools**
Sélectionner sur la fenêtre Orccad: **Spécification**, **Task**…
Ensuite **file**, **open**, **movejRx**
Sur le bloc **jtraj**, prendre le fichier **compute.c**

Sélectionner dans la fenêtre Orccad : **Exécution**, **procédure**
Ensuite **file**, **open**, **manageRX90**, **env**, **apply**
Cliquer sur le bouton **c++** pour générer le code.

II.2.2. Pour compiler :

Se loguer sur Ciney puis:
Cd **Orccad/Exec/Appli/manageRx90**
Ensuite : **make**
Cd **Orccad/User**
Ensuite : **make**

II.2.2. Modifications a apporter

Dans le fichier **compute.c** ligne 176
Remplacer : **if(tf<0.0)** par **if(tf<0.0 || Dmax/tf>=VMAX)**
Pour faire la protection contre les grandes vitesses.

II.3. l'Optotrak

II.3.1. Sources

Sur le compte de RPG
/SRC/OPTOTRAK/OptoSocket.C
/SRC/OPTOTRAK/APIopto.C
/SRC/OPTOTRAK/APIopto.h

II.3.1. Pour compiler

Dans **/SRC/OPTOTRAK**
Faire **make** (voir Laurence)

II.3.1. Pour lancer

Allumer l'Optotrak (boîtier+cameras)
Puis dans **/SRC/OPTOTRAK** faire **./OptoSocket OptoSocket.xml file_name**
OptoSocket.xml est un fichier qui configure le temps d'échantillonnage et le nombre de capteurs utilisé (voir Laurence).
file_name est le nom du fichier ou seront sauvegardé les mesures, il se situera dans **/HUGE/pissard**

II.2.1 Modifs a apporter

Dans **/SRC/OPTOTRAK/OptoRx/APIopto.c**
Modifier la fonction **void OptoRepere(double cpt[6][3], double snaP[3][4])**
Elle reçois dans **cpt** les 3 coordonnées (**x y z**) de 6 capteurs et elle renvoie dans **snaP** une matrice homogène relative a un repère attaché a ces 6 capteurs.
Modifier dans le cas ou les capteurs seront disposés sur un cube ou une sphère. De manière à ce que les capteurs ne soient pas tous occultés et ainsi pouvoir toujours identifier le repère.

II.3. Souris 3D

II.3.1. Sources

/Rx90/Souris/Souris.c

II.3.2. Pour compiler

Dans **/Rx90/Souris/**
Faire **make**.

II.3.3. Pour lancer

Lancer d'abord le driver : **/Xdriver/xdriver**
Puis lancer **/Rx90/Souris/linux/Souris**

II.4. DREAMS

II.4.1. Sources

Fichier de config :
/Rx90/DreamsFiles/displays-src.xml
/Rx90/DreamsFiles/telemetry-src.xml

II.4.2. Pour lancer

./dreams

II.5. Demon

II.5.1 Sources

/Rx90/Appli/Rx90Teleop.c
/Rx90/Appli/Rx90Teleop.h
/Rx90/Appli/mytestRX90.c

II.5.1. Pour compiler

Dans **/Rx90/Appli**
Faire **make**

II.5.3. Pour lancer

/Rx90/Appli/linux/mytestRX90 puis choisir **d** dans le menu. Pour faire marcher la téléopération sans l'interface graphique, choisir **t** dans le menu.

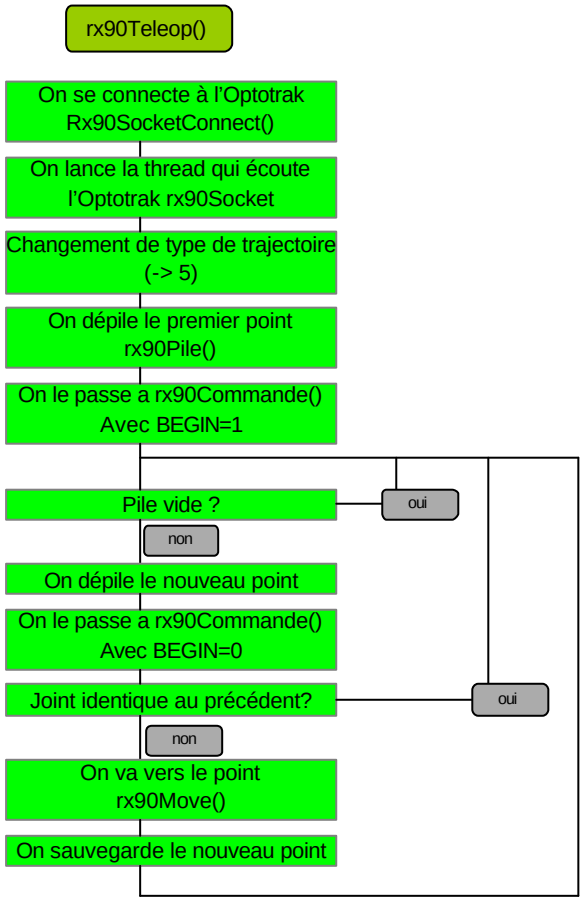
II.5.4. Modifications a apporter

Dans **/Rx90/Rx90Teleop.c**, dans la fonction **rx90Boot()**, modifier la commande **system("xterm…..** car pour l'instant il faut l'exécuter 2 fois pour que le Rx90 puisse booter.

III. Organigrames

III.1. Fonction rx90Teleop()

Cette fonction fait appel aux fonctions : rz90SoxketConnect(), rx90Socket(), rx90Pile(), rx90RTLtoSNA(), rx90Commande(), rx90Move().



III.2. Fonction rx90Demon()

Cette fonction est utilisée pour réaliser la téléopération paramétrée par l'IHM.
Cette fonction fait appel a : rx90SockConnect(), rx90SockIHM(), rx90Robot(), rx90SiteM().
Une structure est déclarée en globale IHM_Data. Elle contient des information partagée par les différentes taches que lance le démon et est mise à jour par la tache qui écoute l'IHM.

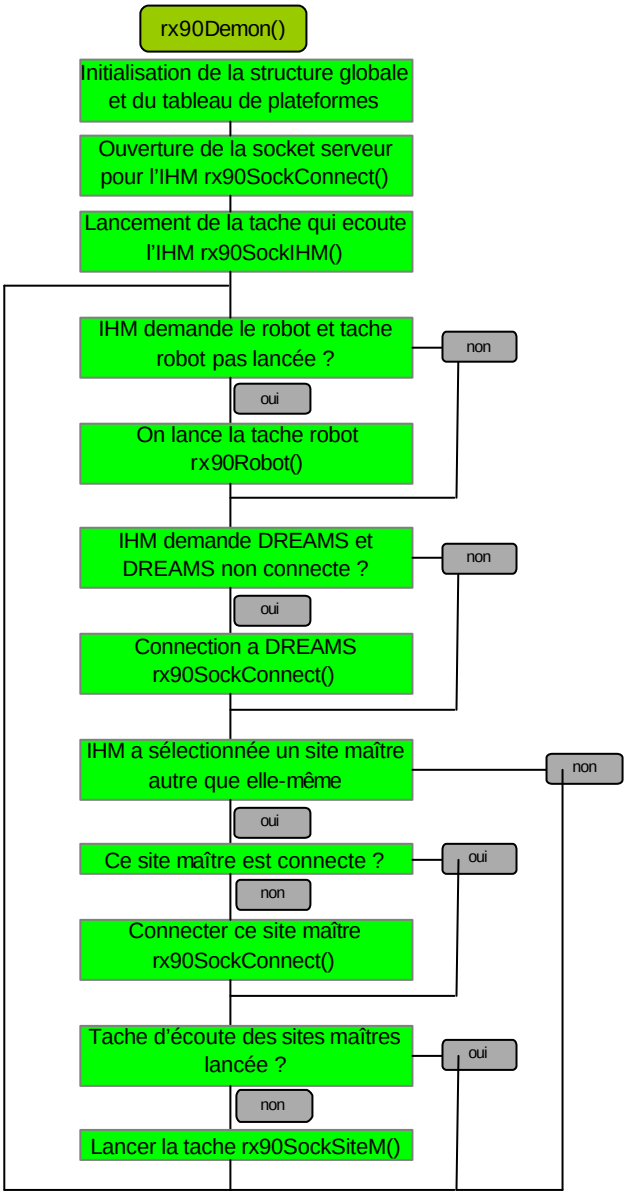
```
typedef struct IHM_Param
{
  int IHM_Telem;
  int DREAMS_Telem;
  int SiteM;
  int RobotRX90;
  int IGRIP;
  int ABS_POS;
  int ABS_ORI;
  int Susp;
  int ArtCart;
  double CoefDem;
  double Retard;
  int RX90_Stat;
  int DREAMS_Stat;
  int IHM_Telem_Stat;
  int IGRIP_Stat;
  int Opto_Stat;
  int FoB_Stat;
  int Souris_Stat;
  int SockListen_Stat;
} IHM_Param;

IHM_Param IHM_Data;
```

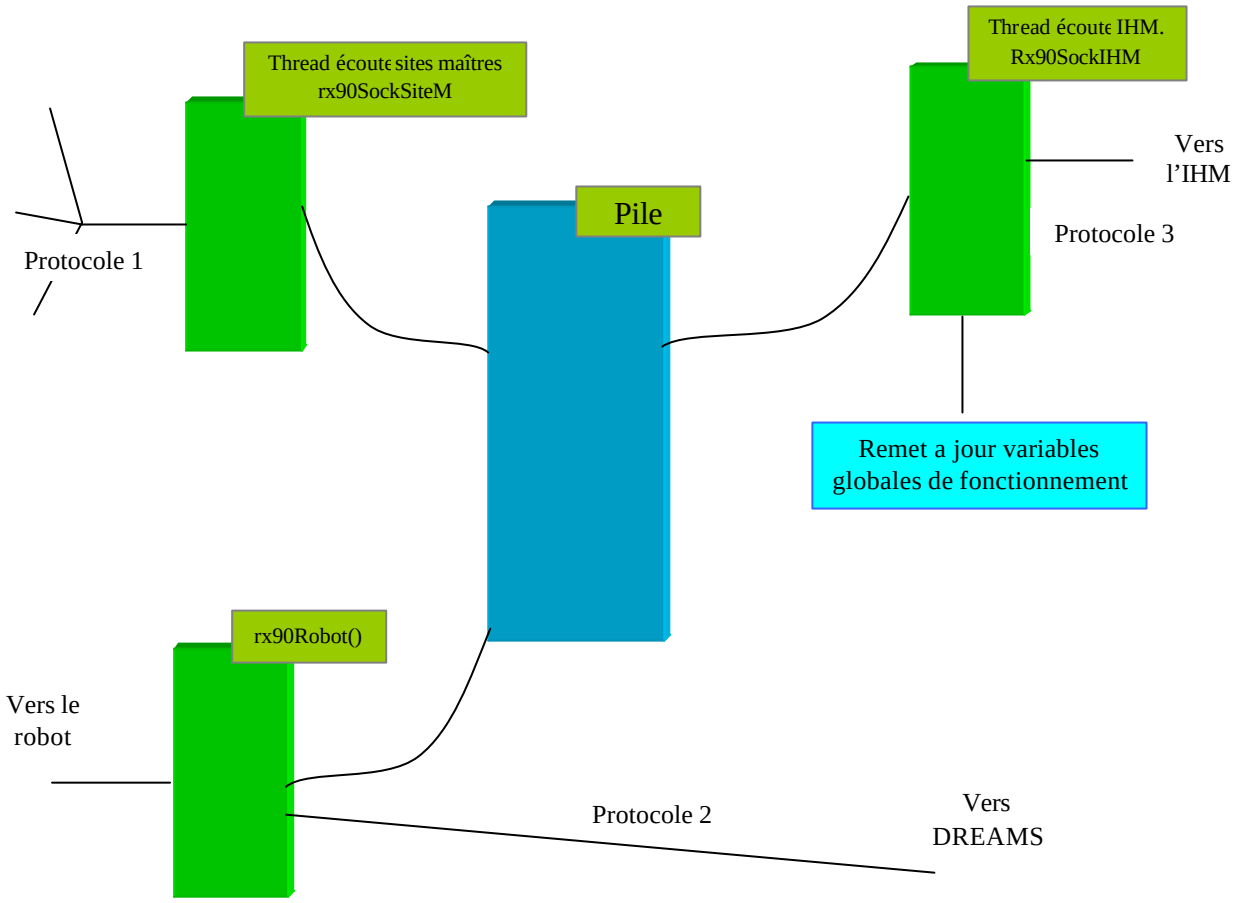
Un tableau de structure est aussi déclaré en globale. Les structures qu'il contient, renseignent sur les propriétés de chaque Plateforme :

```
typedef struct Plateforme
{
  SOCK_SERVER Socket;
  char AddIP[30];
  int Port;
  char message[MAX_SIZE_MESSAGE];
  int connected;
} Plateforme;

Plateforme TabPlateforme[10];
```



III.2.1 Structure du démon :



Protocole 1 :
Les sites maître sont serveur et attendent une connexion du démon.
Après la connexion, les données sont envoyées sous le format suivant : "**temps x y z phi thêta psi \n**"

Protocole 2 :
DREAMS est Client et se connecte sur le demon.
Après connexion le démon lui envoie la configuration articulaire du robot : "**TM TmPacket Telemetrie " +time ?1, ?2, ?3, ?4, ?5, ?6 + ";\n**"

Protocole 3 :
L'IHM est cliente, elle se connecte sur le démon.
Après connexion, si elle envoi une chaîne de caractères commençant par "**10** ",le reste de la chaîne sera les **paramètres de configuration** du démon.
Si la chaîne commence par "**100** ", la suite sera compose de numéro de rang de plateforme, puis de son **port** de connexion et enfin de son adresse **IP**.
Si la chaîne commence par "**1000** ", c'est l'IHM qui demande au démon de lui envoyer les **états** des différents éléments.
Si la chaîne commence par "**10000** ", c'est un **ordre de positionnement** pour le robot envoyé par l'IHM.