Loïc TOUZARD

Gaëtan GOUZI

B3325

3 Avril 2015
Département Informatique
Groupe 3

Compte-Rendu TP Multitâche

Gestion d'un carrefour

Loïc TOUZARD

Gaëtan GOUZI

B3325

3 Avril 2015
Département Informatique
Groupe 3

Compte-Rendu TP Multitâche

Gestion d'un carrefour

Sommaire :

Loïc TOUZARD

Gaëtan GOUZI

B3325

3 Avril 2015
Département Informatique
Groupe 3

I. Présentation

1) Différences entre la version finale et la version originale

a) A l 'origine, nous avions prévu que la tâche Feu envoie un signal à la tâche Voie pour lui indiquer que les feux changent de couleur (un signal « bascule » qui permet de savoir quel feu est vert et lequel et rouge).
Cette solution a été abandonnée pour mettre en place une mémoire partagée qui contient les couleurs courantes relatives à chaque axe (NS et EO). En effet, la gestion de la mémoire partagée (Lecture/Écriture) est plus simple et plus propre que d'utiliser un signal alternatif qui change la valeur des variables une fois sur deux. Il vaut mieux éviter l'utilisation d'un signal si une alternative est possible.

b) La version original ne prévoyait pas de moyen de communication entre la tâche Menu et la tâche Générateur. Or, les voitures peuvent être crées sans le générateur (génération manuelle par l'utilisateur). Ainsi, la boite aux lettres qui contient les messages permettant de générer les voitures est accessible en lecture par la tâche Voie et en écriture par les tâches Générateur et Menu

Loïc TOUZARD

Gaëtan GOUZI

B3325

3 Avril 2015
Département Informatique
Groupe 3

2) Rôles principaux des tâches :

- La tâche Mère s'occupe de la création, de l'initialisation et de la destruction des IPCs, des tâches principales et de l'IHM.
- Les tâches Voie s'occupent de dessiner les voitures générées et de les déplacer. Une tâche Voie par point cardinal.
- La tâche Menu récupère et traite les informations données par clavier. Elle dirige ainsi le générateur, permet une génération manuelle des Voitures et également une modification de la durée des feux.
- La tâche Feu gère les temporisations et couleurs des feux pour appliquer les durées imposées par l'utilisateur sur les voies correspondantes.

3) Les IPCs nécessaires :

Mémoire partagée :

Il existe une seule mémoire partagée qui contient 4 zones, 2 pour les temporisations des axes, et 2 pour les leur couleurs.

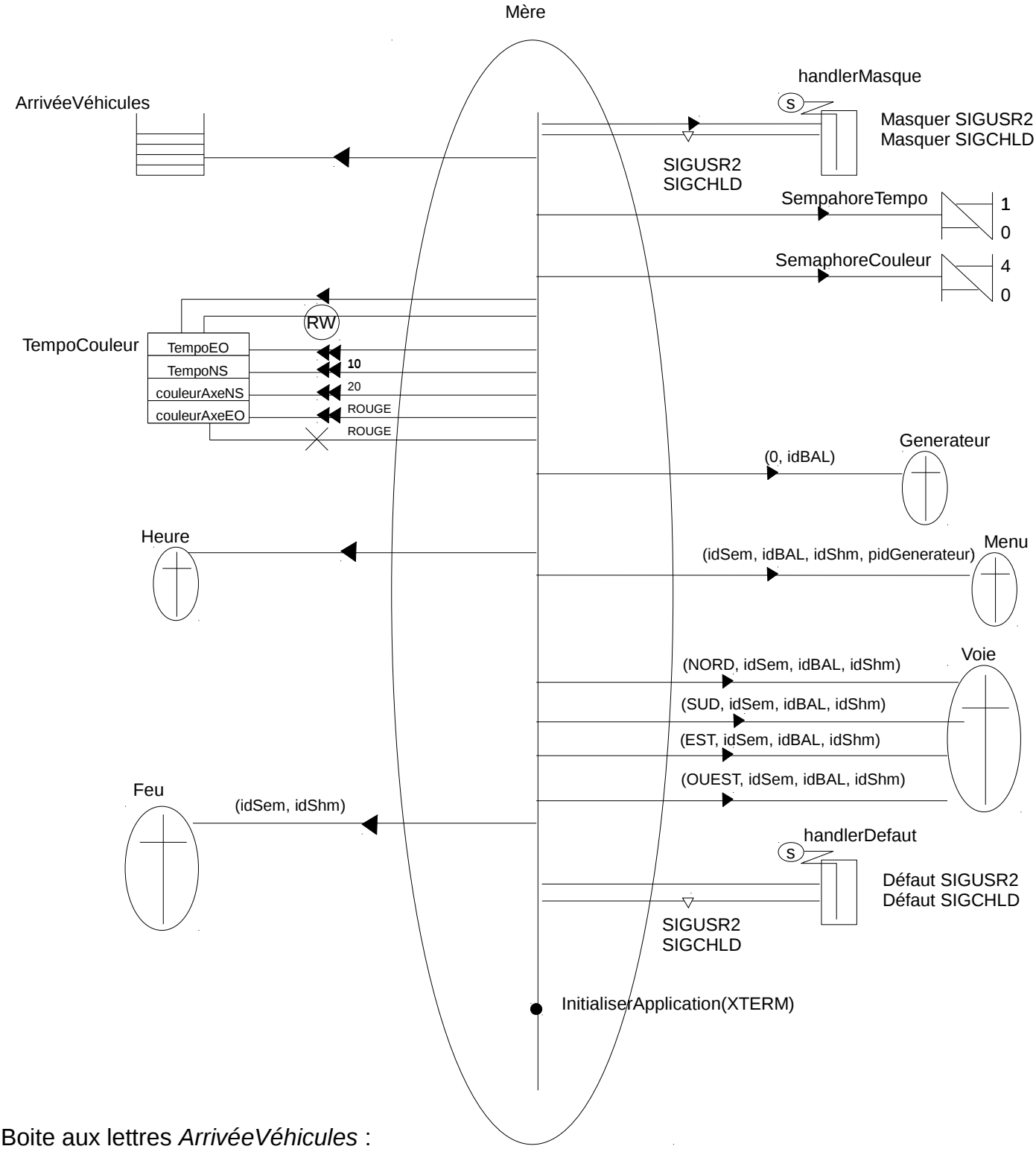
Sémaphores :

Nous avons créé 1 sémaphore général qui contient 2 sémaphores élémentaires protégeant respectivement la zone des temporisations et la zone des couleurs.

File de message :

Nous avons créé une file de message dans laquelle on pose les voitures générées (manuellement ou automatiquement). On retire les voitures lorsqu'elle sont déssinées.

Mère
Initialisation



-Boite aux lettres *ArrivéeVéhicules* :

Stocke des messages *msgVoiture* indiquant au générateur quel type de voiture il faut créer

-Mémoire partagée *TempoCouleur* :

Stocke les durées(zones tempo) et les couleurs(zones couleurs) actuelles des feux pour les deux axes (NS et EO)

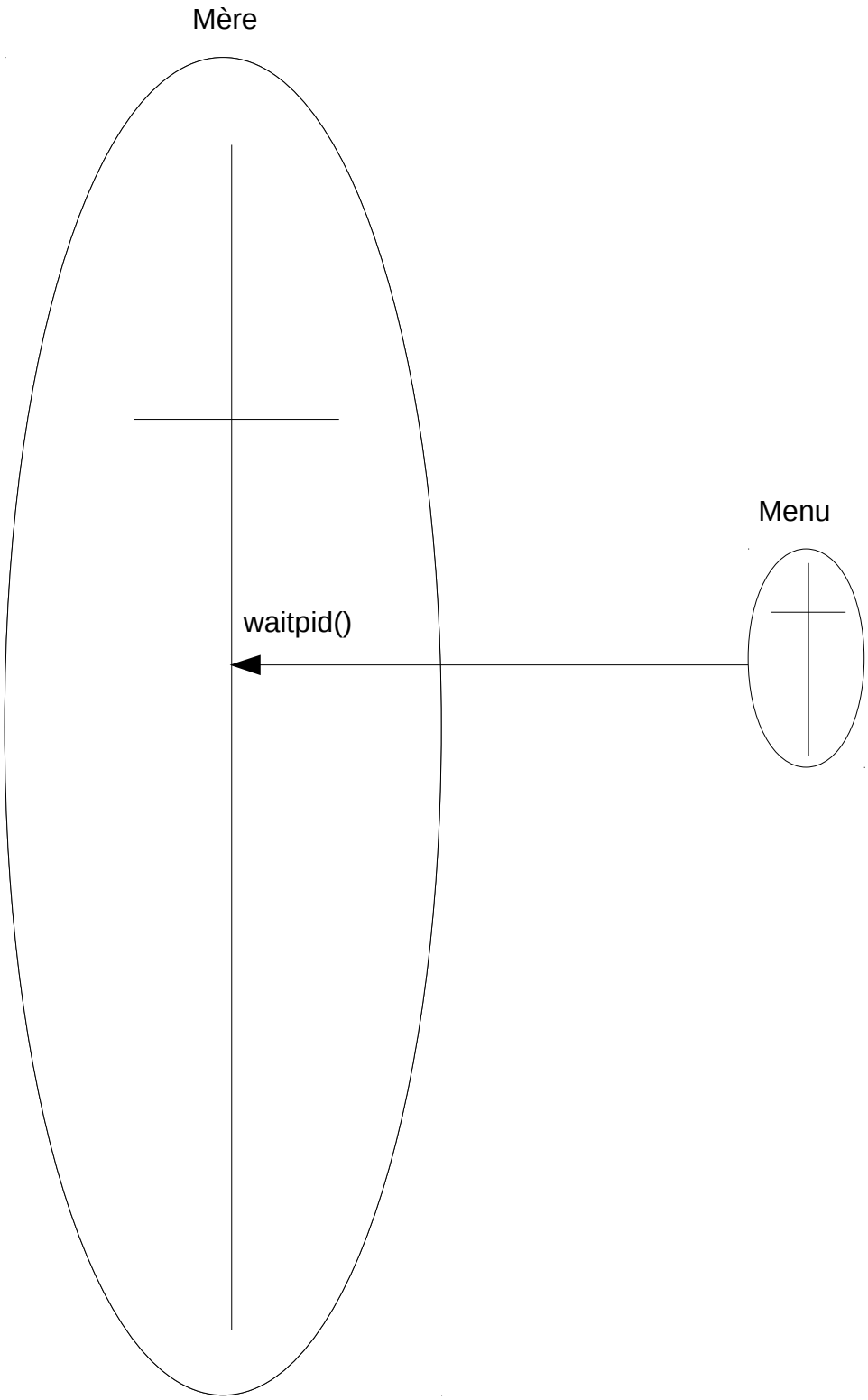
-Sémaphore *SémaphoreTempo* :

Sémaphore associé aux deux zones tempos de la mémoire partagée *TempoCouleur (1 Jeton max)*

-Sémaphore *SémaphoreCouleur* :

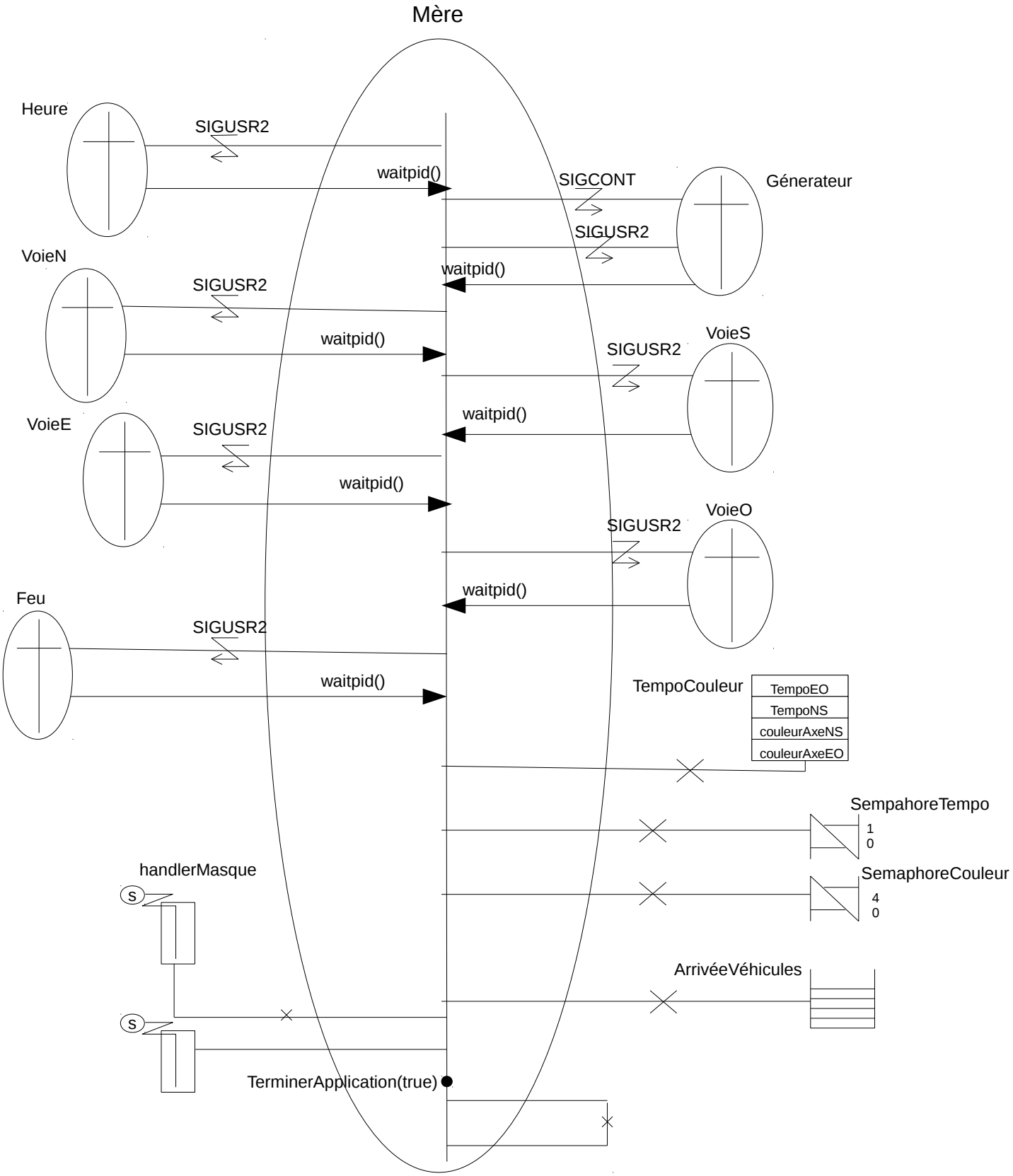
Sémaphore associé aux deux zones couleurs de la mémoire partagée *TempoCouleur (4 Jetons max car 4 instances de Voie qui peuvent potentiellement effectuer une lecture en même temps)*

Mère
Moteur



La phase moteur de la tâche Mère consiste à attendre le signal SIGCHLD de la part de la tâche Menu

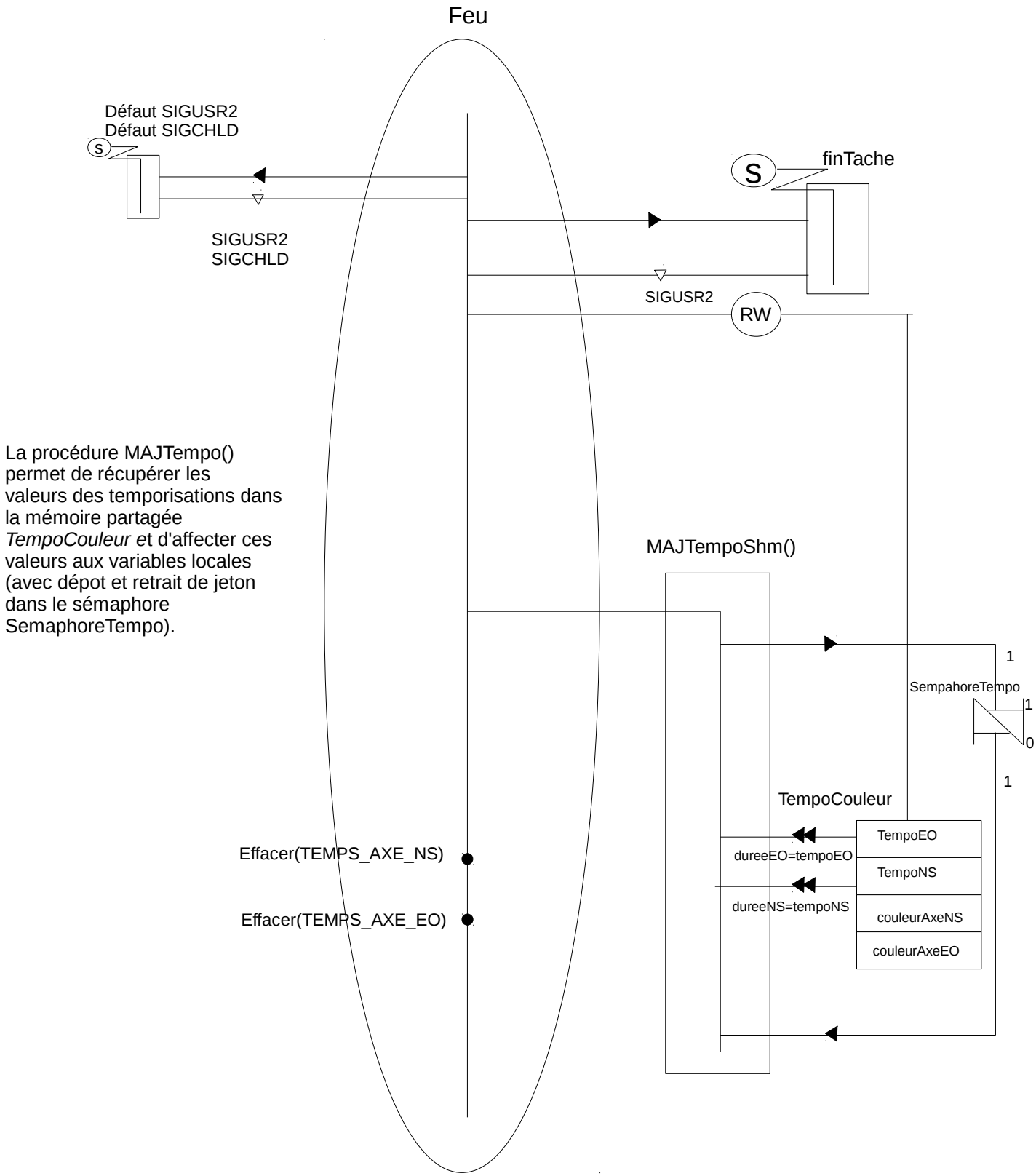
Mère
Destruction



Pour chaque tâche fille : On envoie le signal de fin SIGUSR2
On attend la réception du signal SIGCHLD

Puis on détruit les IPCs avant de terminer l'application et de s'autodétruire

Feu
Initialisation



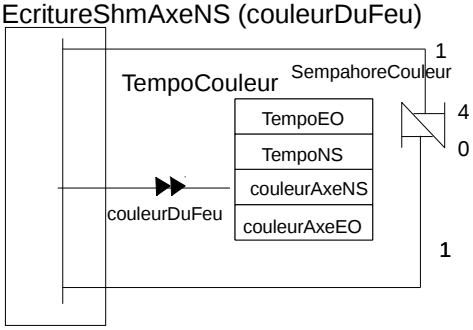
On réactive les signaux qui étaient déactivés dans la mère et on crée un handler finTache sur SIGUSR2
On s'attache à la mémoire partagée puis on récupère les temporisations stockées dans celle-ci
On met à blanc les zones où l'on s’apprête à écrire

Feu

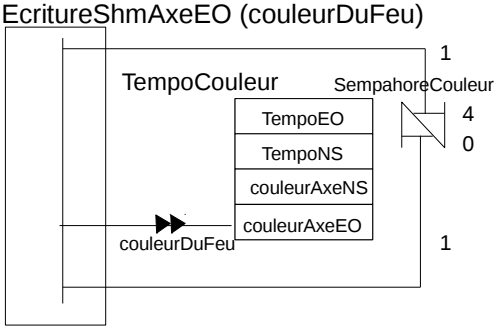
Feu Moteur

- Afficher(DUREE_AXE_NS, dureeNS)
- Afficher(DUREE_AXE_NS, dureeEO)
- EcritureShmAxeNS(idSem, shm, VERT)
- Effacer(COULEUR_AXE_NS)
- Afficher(COULEUR_AXE_NS,"Vert", GRAS, INVERSE)
- Decomptetempo(dureeNS,tempoNS, tempoEO)
- EcritureShmAxeNS(idSem, shm, ORANGE)
- Effacer(COULEUR_AXE_NS)
- Afficher(COULEUR_AXE_NS,"Orange", GRAS, INVERSE)
- Decomptetempo(TEMPO_ORANGE,tempoNS, tempoEO)
- EcritureShmAxeNS(idSem, shm, ROUGE)
- Effacer(COULEUR_AXE_NS)
- Afficher(COULEUR_AXE_NS,"Rouge", GRAS, INVERSE)
- Decomptetempo(TEMPO_ROUGE,tempoNS, tempoEO)
- MAJTempoShm(idSem, shm, dureeNS, dureeEO);
- Afficher(DUREE_AXE_NS, dureeNS)
- Afficher(DUREE_AXE_EO, dureeEO)
- EcritureShmAxeEO(idSem, shm, VERT)
- Effacer(COULEUR_AXE_EO)
- Afficher(COULEUR_AXE_EO,"Vert", GRAS, INVERSE)
- Decomptetempo(dureeEO,tempoNS, tempoEO)
- EcritureShmAxeEO(idSem, shm, ORANGE)
- Effacer(COULEUR_AXE_EO)
- Afficher(COULEUR_AXE_EO,"Orange", GRAS, INVERSE)
- Decomptetempo(TEMPO_ORANGE,tempoNS, tempoEO)
- EcritureShmAxeEO(idSem, shm, ROUGE)
- Effacer(COULEUR_AXE_EO)
- Afficher(COULEUR_AXE_EO,"Rouge", GRAS, INVERSE)
- Decomptetempo(TEMPO_ROUGE,tempoNS, tempoEO)
- MAJTempoShm(idSem, shm, dureeNS, dureeEO)

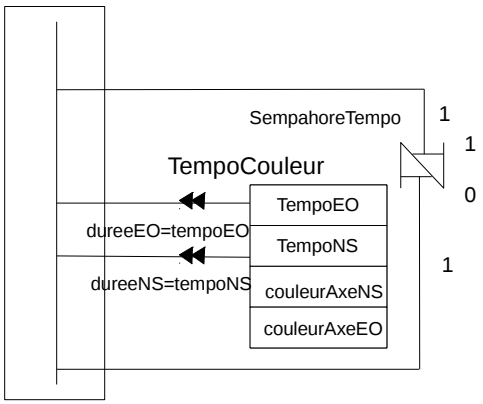
EcritureShmAxeNS (couleurDuFeu)



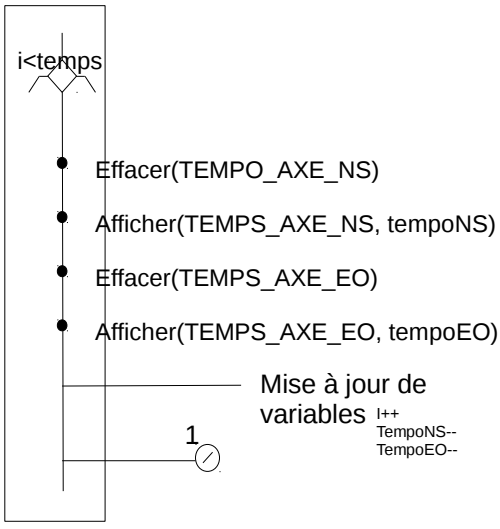
EcritureShmAxeEO (couleurDuFeu)



MAJTempoShm ()



DecompteTempo (temps, &tempoNS, &tempoEO)



Constantes et variables :

TEMPO_ORANGE = 3 Correspond au temps (en secondes) pendant lequel les feux restent oranges

TEMPO_ROUGE = 2 Correspond au temps (en secondes) pendant lequel les feux restent rouges

dureeNS = valeur contenue dans la zone tempoNS dans la mémoire partagée *TempoCouleur*

dureeEO= valeur contenue dans la zone tempoEO dans la mémoire partagée *TempoCouleur*

tempoNS = variable que l'on affiche dans la zone TEMPS_AXE_NS et se décrémentant chaque seconde, indique le temps restant aux feux situés sur l'axe NS avant de changer de couleur

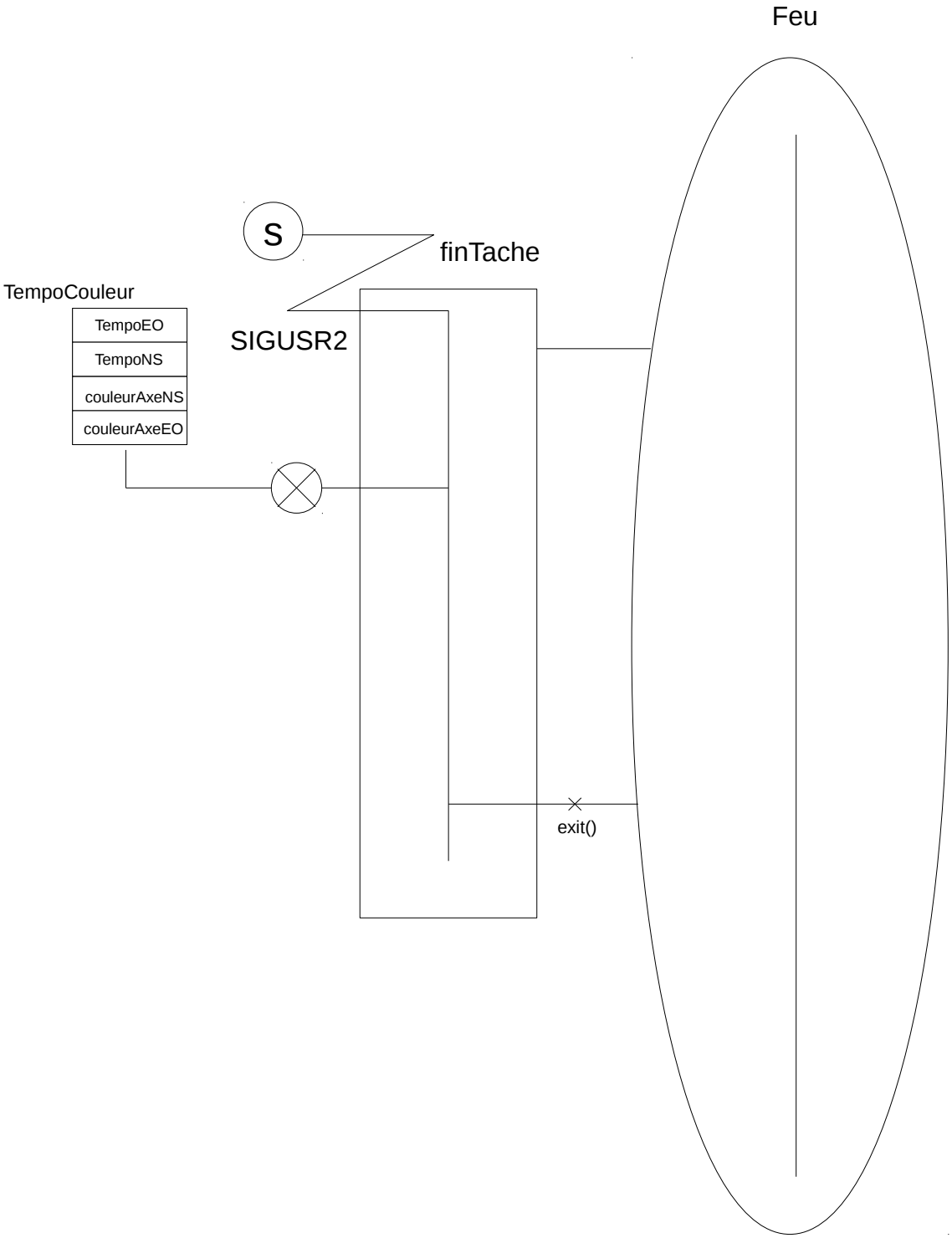
tempoEO = variable que l'on affiche dans la zone TEMPS_AXE_EO et se décrémentant chaque seconde, indique le temps restant aux feux situés sur l'axe EO avant de changer de couleur

Explications sur les procédures utlisées :

Les procédures EcritureShmAxeEO() et EcritureShmAxeNS() permettent d'écrire de manière sécurisée (avec dépot et retrait de jeton dans le sémaphore SemaphoreCouleur) dans la mémoire partagée *TempoCouleur* respectivement dans les zones couleurAxeEO et couleurAxeNS.
Ainsi, à chaque changement de couleur, on fait appel à ces méthodes pour mettre à jour les couleurs des feux dans la mémoire partagée *TempoCouleur*.

La procédure DecompteTempo() permet de « compter » x secondes, d'effectuer les affichages dans les zones TEMPS_AXE_NS et TEMPS_AXE_EO, et de mettre à jour les variables locales tempoNS et tempoEO.

Feu
Destruction



A la réception du signal SIGUSR2,le handler se déclenche :
La tâche Feu se détache de la mémoire partagée et s'autodétruit

Loïc TOUZARD

Gaëtan GOUZI

B3325

3 Avril 2015
Département Informatique
Groupe 3

VI. Conclusion :

Notre programme est entièrement fonctionnel par rapport aux attentes du cahier des charges.

Malgrès cela, la version finale n'est pas parfaite. Plusieurs erreurs système ne sont pas gérées (la création des IPCs par exemple).

Un autre '*problème*' peut aussi survenir, bien que peu probable, il est théoriquement réalisable : Dans la tâche Voie, si l'on réceptionne la mort d'une voiture fille grâce au handlerSIGCHLD, on ne pourra pas réceptionner la mort d'une autre voiture sur cette voie tant que la première n'aura pas fini d'être gérée. En effet la superposition des signaux n'a pas été mise en place. Dans ce cas cela laisserait une voiture zombie jusqu'a la destruction de la tâche voie. Où dans la phase destruction, toutes les voitures non récupérées sont nettoyées. Si cela venait a se produire plusieurs fois dans une longue execution cela monopoliserait de l'espace mémoire inutile. Dans la pratique le temps pour que deux voitures successives meurent est bien plus important que le temps de traitement du handlerSIGCHLD.