Pierre LEMARQUAND
Guillaume MONNET

Rapport final - Cas d'étude

"Drone pour contrôler et suivre l'ours des Pyrénées"

Position du problème :

Afin de réduire le nombre d'animaux d'élevage tués par les ours, une entreprise a décidé d'innover en proposant un drone de surveillance. Celui-ci a comme mission de survoler les zones alentours au troupeau et de traquer la présence d'ours.

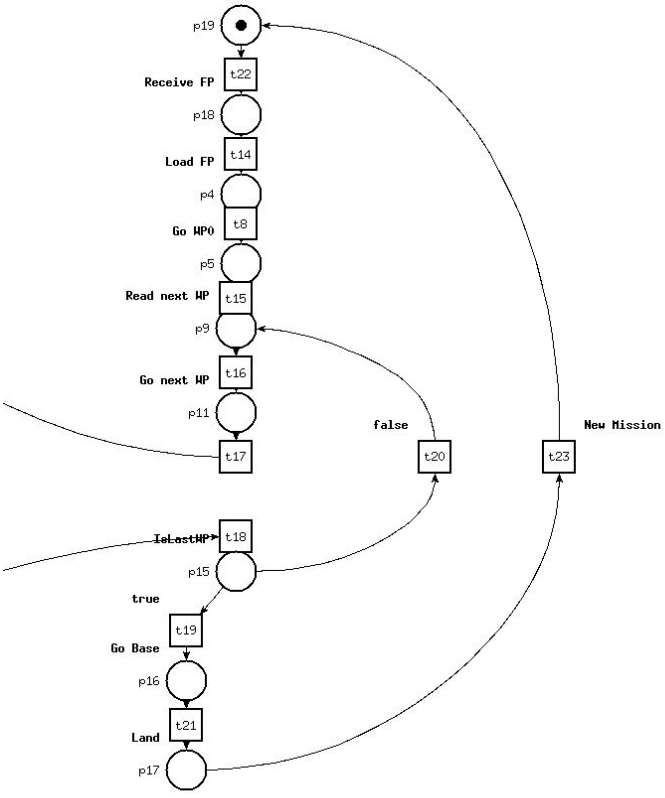
Pour ce faire, le drone est équipé d'une caméra infrarouge et d'un GPS. Un plan de vol comportant une liste ordonnée de points à surveiller est chargé à chaque début de mission. Lors du survol de chaque zone, il analyse les alentours. Si la signature thermique de ce qu'il analyse est celle d'un ours et que la position de celui-ci est inférieure à 1 km du par rapport au troupeau, le drone doit alors envoyer un message au berger et émettre un bruit et une lumière visant à faire fuir l'ours.

L'objectif de ce cas d'étude est de traduire les diagrammes SysML fournis en réseaux de Petri Interprétés de les analyser formellement.

Pierre LEMARQUAND
Guillaume MONNET

I. Mission du drone

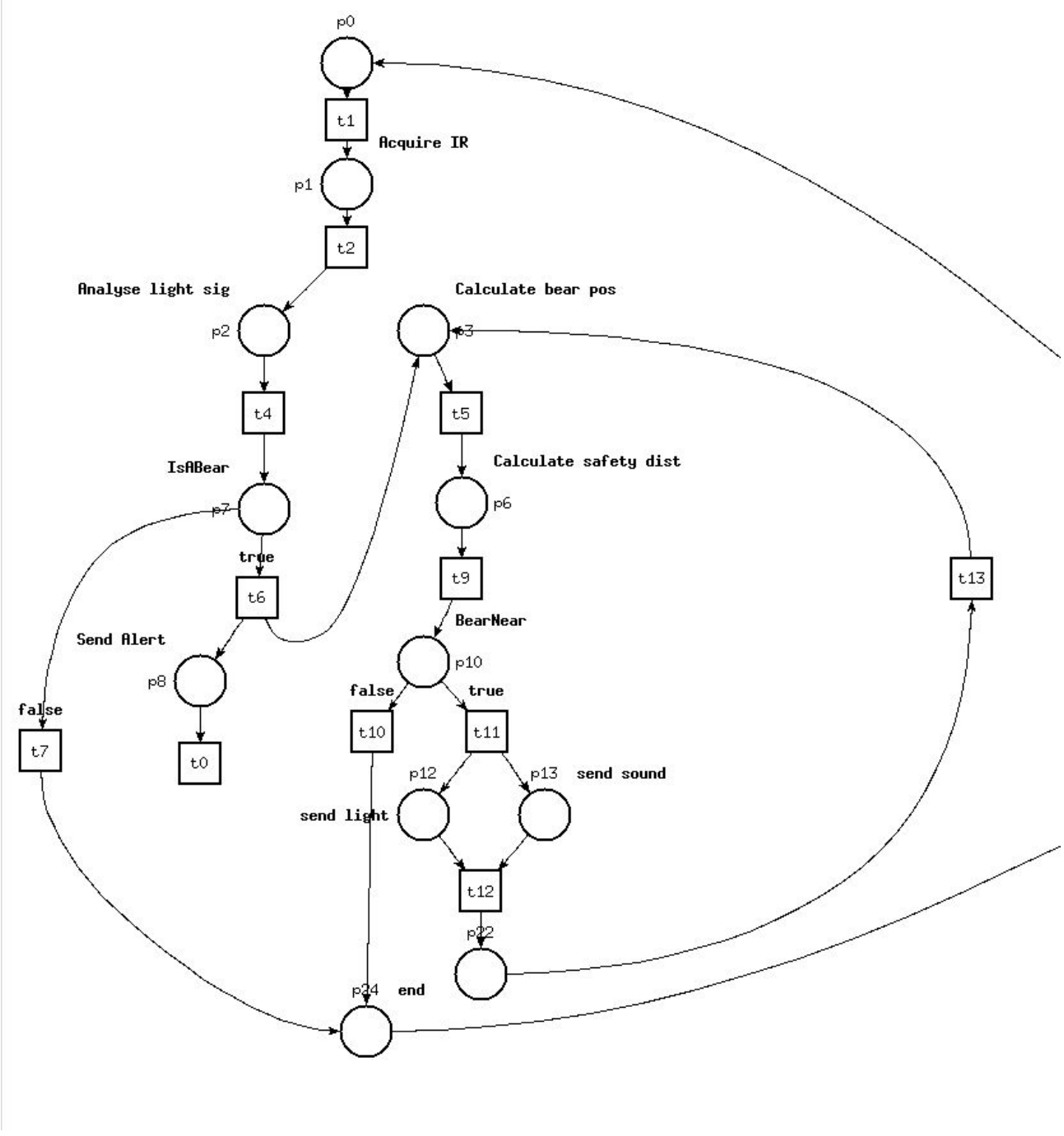
Le fonctionnement du drone lors d'une mission est linéaire et donc très simple à modéliser par un réseau de Pétri. On peut observer la boucle sur les points de passage, ainsi que la boucle de fonctionnement global (qui permet de borner le réseau). À noter, on voit ici 2 traits sortir du réseau, ils correspondent à l'étape où le drone effectue le protocole de surveillance de l'ours.



II. Protocole de surveillance

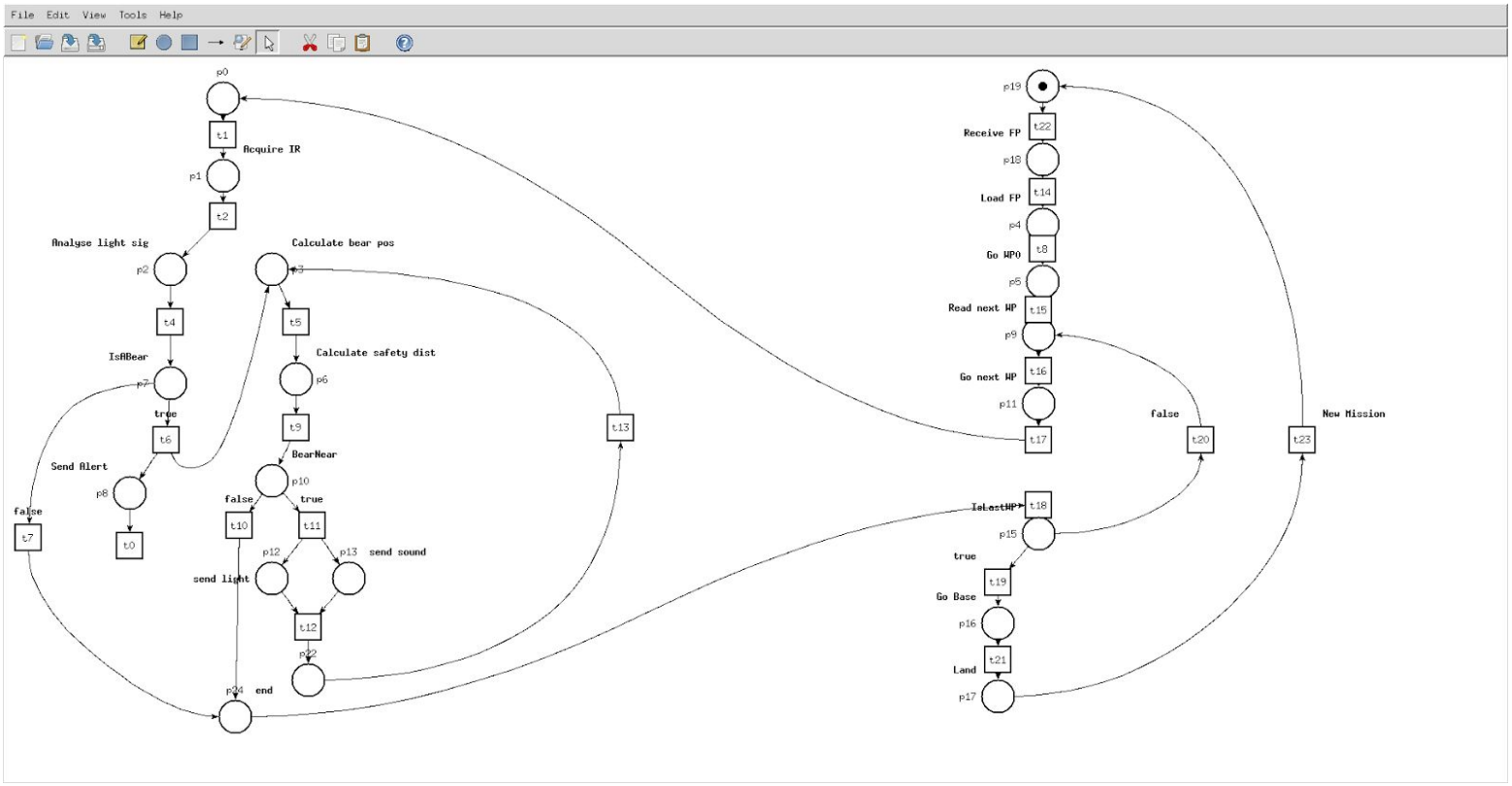
Nous avons décidé de modéliser le protocole de surveillance de l'ours de façon linéaire et logique, qui se décompose en plusieurs grandes étapes de décision :

- Acquisition et décision sur la cible (**IsABear**)
- Si pas d'ours : fin de la surveillance de ce point de passage
- Si on observe bien un ours : envoi de l'alerte au berger puis…
    - Calcul et analyse de la distance par rapport au troupeau (**BearNear**)
- Si l'ours est éloigné : fin de l'observation du point de passage
- Si l'ours est proche : déclenchement des signaux lumineux et sonores pour l'effrayer, puis recalcul de sa position (boucle par **t13**)



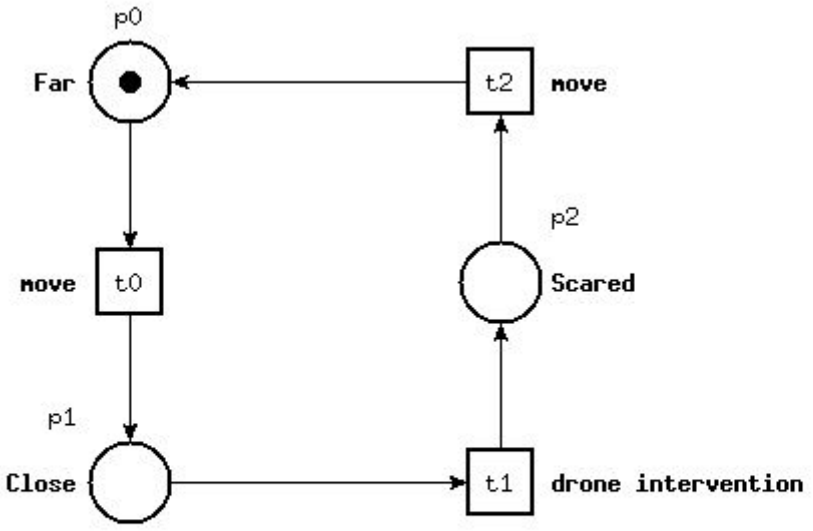
Pierre LEMARQUAND
Guillaume MONNET

III. Modèle de l'UAV complet



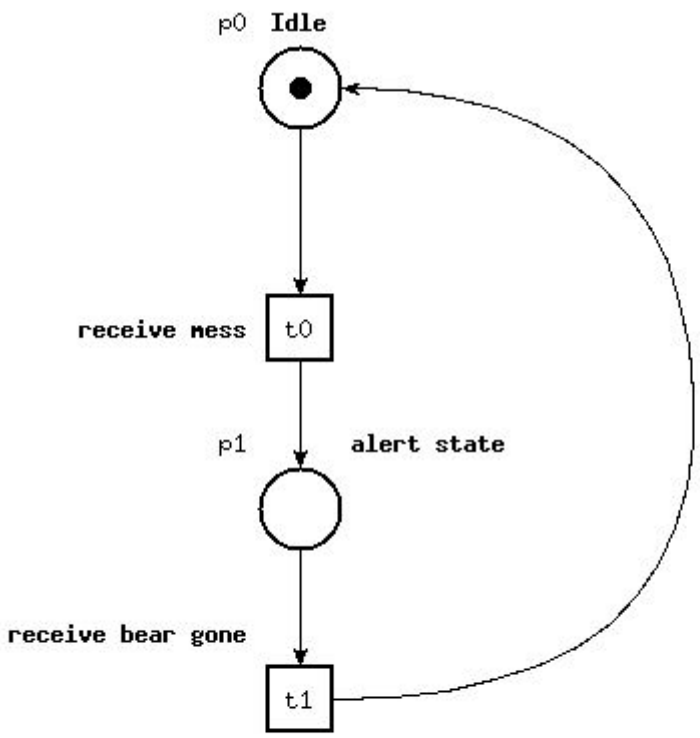
En combinant les 2 réseaux précédents, on obtient le réseau décrivant le drone lors de l'intégralité de ses missions de surveillance des ours. Cependant, à cause de l'état **p8**, mis en place pour assurer les communications avec le réseau modélisant le berger, on n'observe pas de propriétés intéressantes. Cependant, en enlevant **p8** et **t0**, on a bien un réseau *borné*, *vivant* et *inversible*.

IV. Modèle de l'ours



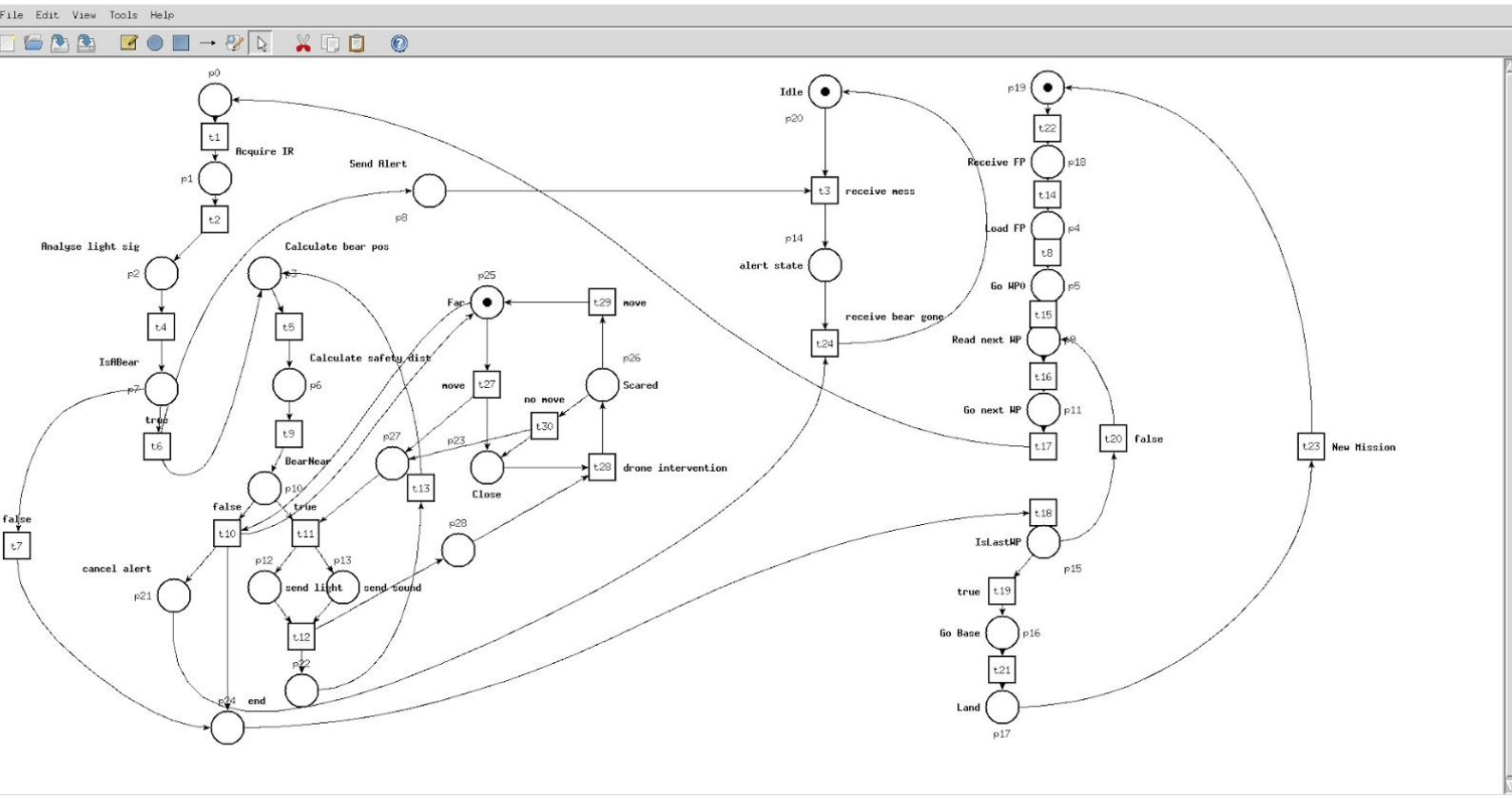
Pour modéliser l'ours, nous avons choisi un modèle linéaire simple à plusieurs états pertinents (**Far**, **Close** et **Scared**). À noter que nous avons décidé par la suite d'ajouter des transitions supplémentaires entre ceux-ci pour garantir le synchronisme avec le système global.

V. Modèle du berger



Pour modéliser le berger, nous avons choisi un réseau du même type que celui de l'ours, avec seulement 2 états, **Idle** et **Alert**.
De même, des transitions supplémentaire seront ajoutées lors de la réconciliation entre les différents réseaux pour garantir le synchronisme du système global.

VI. Modèle global

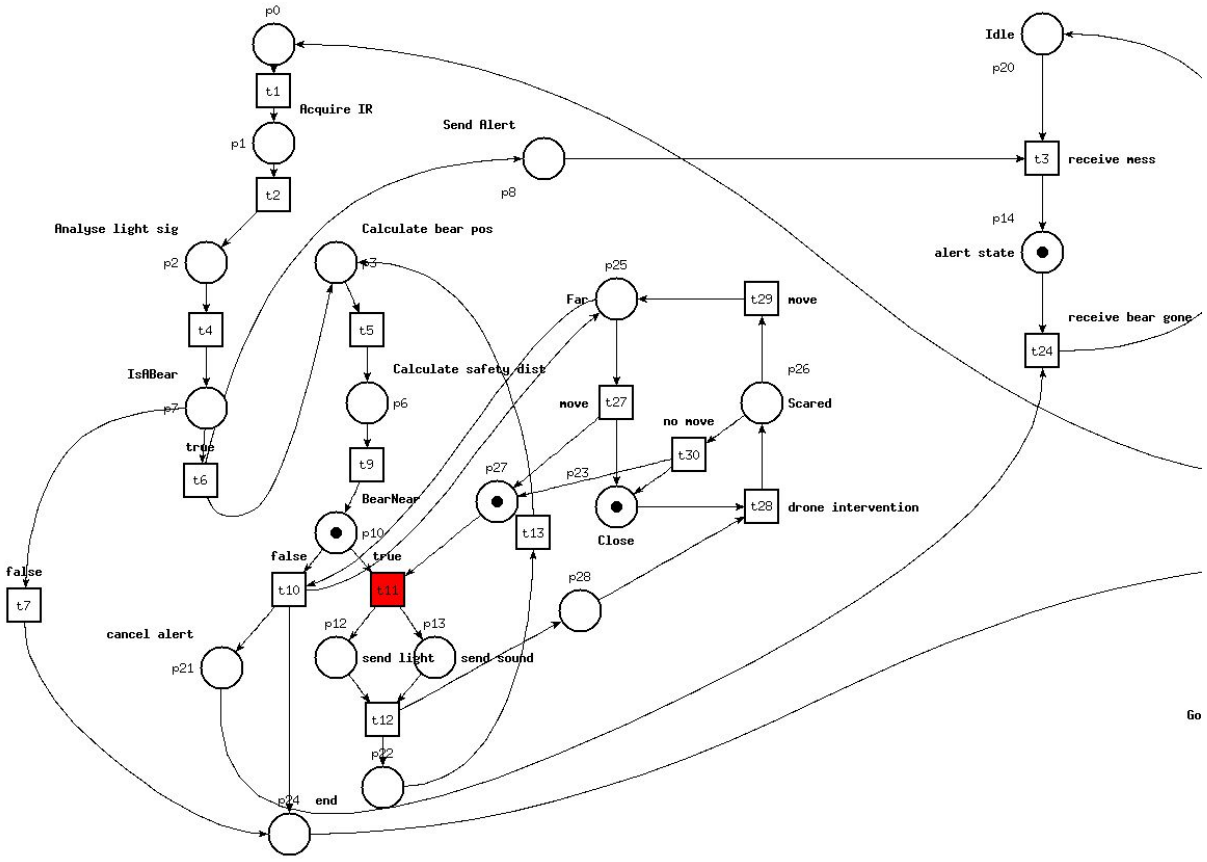


Après réconciliation des différents réseaux, nous avons du établir les points de communications et de synchronisation des différents systèmes.

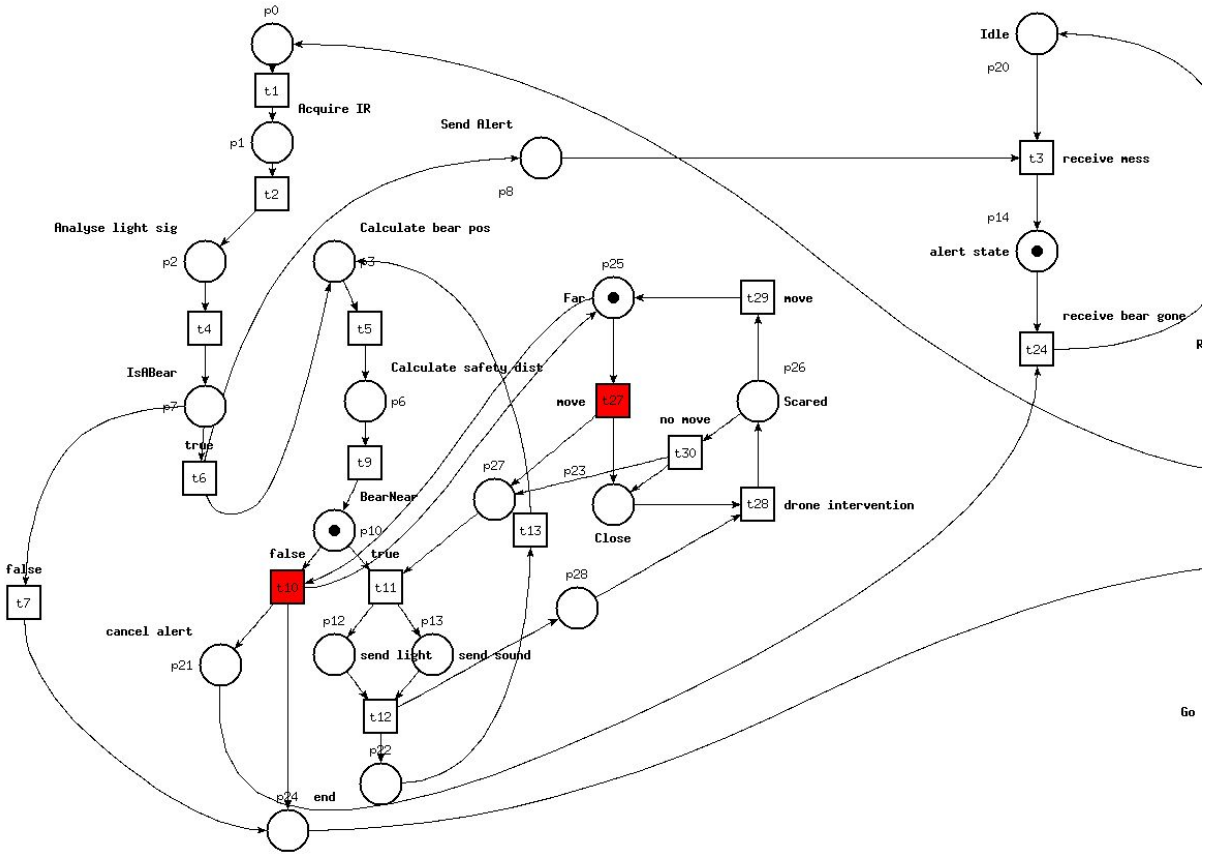
Pour le berger, initialisé dans l'état **Idle**, de manière simple, nous attendons un jeton par la place **SendAlert** avant de passer dans l'état **Alert**. De la même manière, nous interrompons cet état d'alerte quand l'ours s'est éloigné des troupeaux.

Nous initialisons l'ours dans l'état **Far**, et ses déplacements vont conditionner les possibilités de décision du protocole de surveillance, afin qu'aucun scénario ne soit possiblement erroné : nous pouvons garantir, à partir du moment où l'ours est détecté, que la bonne branche décisionnelle sera empruntée.

Pour ce faire, le rapprochement des troupeaux, par l'action **move**, active l'accès à la branche **BearNear->true**. La fin de l'intervention du drone amène l'ours dans l'état **Scared**, d'où il peut soit rester près soit s'éloigner, déclenchant ou non l'émission d'un jeton **BearNear->true**.
Cf fig ci-dessous



Il est nécessaire pour que le drone emprunte la branche **BearNear->false**, que l'ours soit dans l'état **Far**, grâce à la double transition entre **BearNear->false** et **Far**.
Cf fig ci-dessous



Conclusion

En partant d'un système relativement simple, l'UAV, et de sa mission, nous avons obtenu 2 réseaux de Pétri plutôt simple, et relativement faciles à imbriquer (puisque la surveillance était en fait une étape de la mission globale de l'UAV) grâce à la présence d'un unique jeton.

Cependant, dès que nous avons ajouté des acteurs ayant chacun leur machine à état et un jeton, nous avons observé d'une part la difficulté de faire communiquer tous ces acteurs (apparition de jetons, blocages), mais aussi de les synchroniser, afin que les actions des uns dépendent de l'état des autres.

De quelques réseaux très simples, on aboutit facilement à des réseaux de taille conséquente,, la complexité des tâches de communications n'étant pas vraiment linéaire, comme nous avons pu le voir lors de ce bureau d'étude.