Protocole de déplacement pour robots à base d'Arduino

Complexité

https://github.com/GRnice/ComplexiteDrone

M1 IFI/RIF

23 juin 2016

Résumé

Ce document décrit le travail effectué tant au niveau de la conception de la formation et du protocole de déplacements des robots que de l'implémentation. Il décrit l'évolution possible du projet ainsi que les personnes ayant participés à sa réalisation.

1 Protocole

1.1 Initialisation

La phase d'initialisation, à l'allumage des robots, se déroule en trois étapes :

- Tirage d'un identifiant unique ;
- Demander s'il existe déjà un chef, s'il n'y en a pas, le robot se considère comme étant le chef ;
- Envoyer son identifiant unique au chef ;
- *Facultatif* : Vérifier s'il existe plusieurs chefs, si oui celui avec l'identifiant le plus grand garde son statut de chef.

Pour finir le chef envoie à chaque robot une paire de nombres composée de l'identifiant du robot et de sa position dans la formation.

1.2 Formation

Les robots se positionnent en triangle, avec le chef en tête et deux esclaves derrière lui.
Chaque esclave est suivit par deux esclaves jusqu'à la fin de la formation.
Les robots sont séparés par une distance de $50 \pm 10\%$ cm.

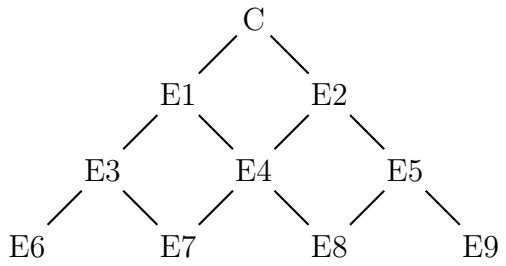

FIGURE 1 – Formation avec dix robots

1.3 Mouvement

Lors du déplacement, le robot avec le rôle de chef respecte les conditions suivantes :

- Avancer en ligne droite ;
- S'il détecte un obstacle, l'éviter ;
- Communiquer ses déplacements aux autres robots.

Le reste des robots obéissent aux règles ci-dessous :

— Écouter les ordres de déplacements du chef ;
— Avancer en fonction de l'ordre reçu et de leur position dans la formation

1.4 Format de communication

0	0
1	len
CRC	
...	
end	end

FIGURE 2 – Modèle de message

Le format de communication se présente comme une trame qui possède une partie fixe et variable avec les éléments suivants :

— *0 0* : début de la trame ;
— *1* : type de trame ;
— *len* : taille de la trame ;
— *CRC* : Cyclic redundancy check, XOR sur chaque niveau de la trame (hors CRC) ;
— *...* : corps du message ;
— *end end* : fin de la trame ;

La trame fait une taille de 32 octets.

1.5 Déroulement des communications

Le schéma des communications commence toujours par un robot qui envoie "chef", ensuite en fonction de la réponse qu'il reçoit, les communications empruntent une branche de l'arbre des communications. Les cases bleus sont les messages envoyés et les cases rouges sont les messages reçu par le robot.

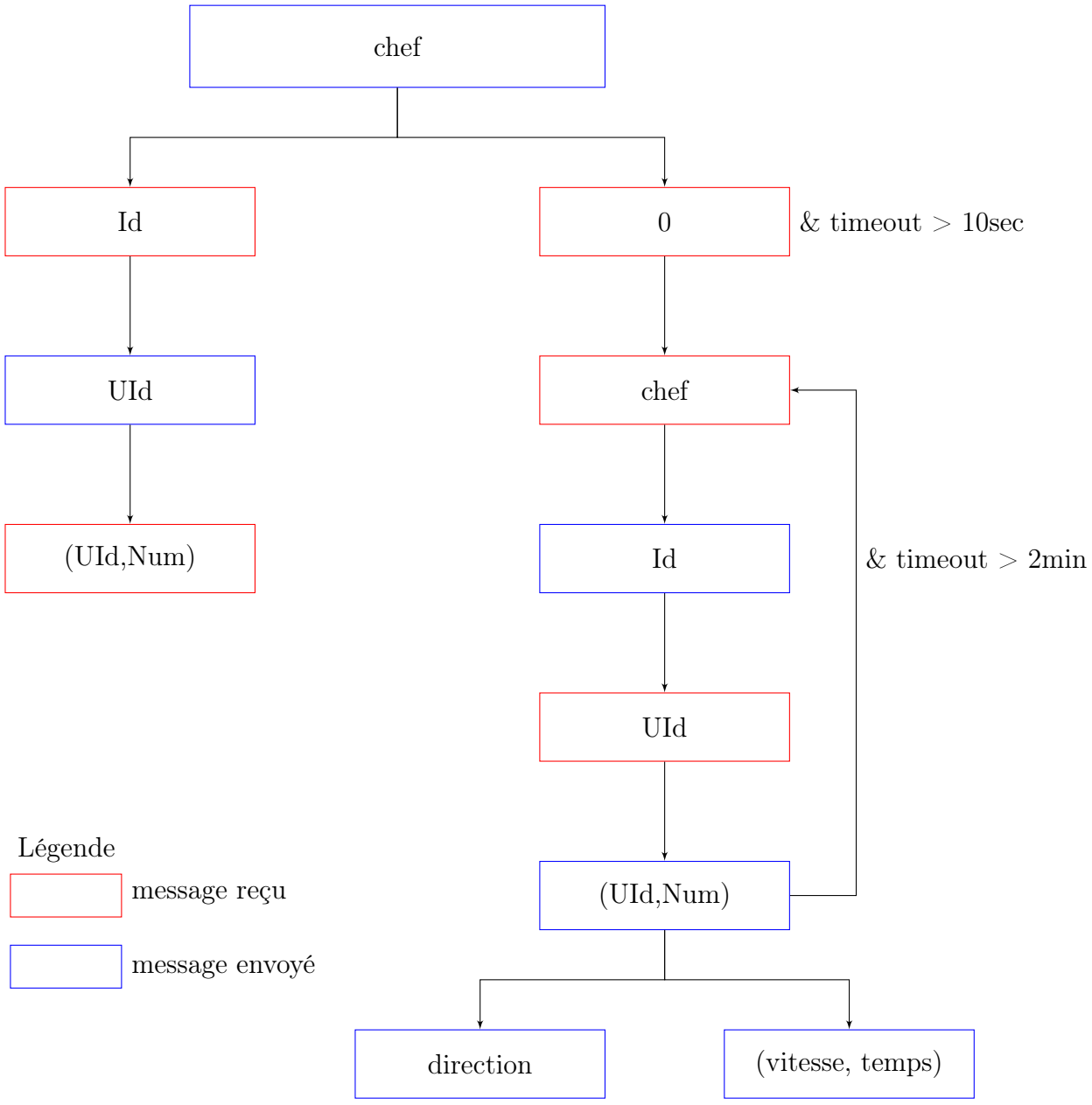

FIGURE 3 – Déroulement des communications du point de vue d'un robot

2 Implémentation

2.1 Machine à états

Le code à été implémenté sous forme de machine à états, avec les trois états suivants :

— ETAT_INIT ;
— ETAT_GO ;
— ETAT_STOP.

Le robot démarre dans l'état *ETAT_INIT*, dans lequel il participe à l'élection du chef. À la fin de la phase d'initialisation, il passe dans l'état *ETAT_GO*. Il est important de noter que la phase d'initialisation du chef est la plus longue, en effet il partira lorsque l'ensemble des suiveurs sera pret et en position, à une distance minimale de leur prédécesseur.

Dès que le chef détecte un mouvement, il envoie un message *stop*, en attendant qu'il choisisse la manière de l'éviter, les suiveurs passent à l'état *ETAT_STOP* et leur moteur s'arrête. Quand le chef trouve une direction libre, il envoie un message aux suiveurs pour qu'ils passent à l'état *ETAT_GO*.

La procédure *go* est elle aussi implémentée à partir d'une machine à états. Ainsi il n'est plus nécessaire pour le chef d'indiquer à chaque tour de boucle aux suiveurs de rester dans l'état *ETAT_GO* et de continuer à avancer. Cela libère donc du temps de communication et réduit la redondance des messages.

2.2 Messages

Les messages sont envoyés uniquement par le maître (ou chef), seul les esclaves (ou suiveurs) peuvent donc les écouter. Ils sont dans une logique simplex avec un mode de communication en broadcast.

2.3 Maître

Le maître peut effectuer deux actions :

— Envoyer un message à chaque fois qu'il effectue un mouvement, c'est-à-dire à chqque tour de boucle ;
— Vérifier la réception d'un message à chaque tour de boucle.

2.4 Esclave

L'esclave peut effectuer l'action suivante :

— Vérifier la réception d'un message à chaque tour de boucle.

3 Participants

— **Julien Audibert**, réflexions à la base du protocole de communication et conception de la formation en pyramide, code de base de détection et évitement des obstacles ;
— **Thomas Grillo**, création de la librairie *com*, implémentant le protocole de communication, de la documentation *doxygen*, de l'implémentation en machine à états, co-création des procédures *go* et *suivre*.
— **Rémy Giangrasso**, auteur de la librairie de communication hardware, débuggage partiel de la librairie *com* ;
— **Rémy Garcia**, réflexions sur la conception, rédaction du rapport ;
— **Clément Lavoisier**, code de base de détection et évitement des obstacles ;
— **Mathias Ellapin**, réflexions sur la conception, co-création des procédures *go* et *suivre* ;

Autres participants : Arthur Finkelstein, Christophe Fagard, Diana Resmerita, Djoé Denne, Dominique Dib, Kévin Caucheteur, Mehdi El Hajami, Mohamed Limam.