POLYTECH GRENOBLE

SUN Bin – ZEGAOUI Taqiyéddine - ELLAPIN Jordan
vous présentent

mindstorms

LEJOS
Java for LEGO Mindstorms

RICM4 – Promotion 2015/2017 06/04/2016

Table des matières

Un tournoi ?

Ce projet est en relation directe avec le tournoi « Persycup » édition 2016. Ce tournoi se déroule le 19 mai 2016 sur le campus de Saint-Martin d'Hères. Pendant cet évènement se déroulent des affrontements entre robots EV3 LEGO Mindstorm, des briques programmables implantées sur des LEGO en forme de robots. Ces robots sont préparés à l'avance par divers groupes d'étudiants venant de différentes écoles de Grenoble.

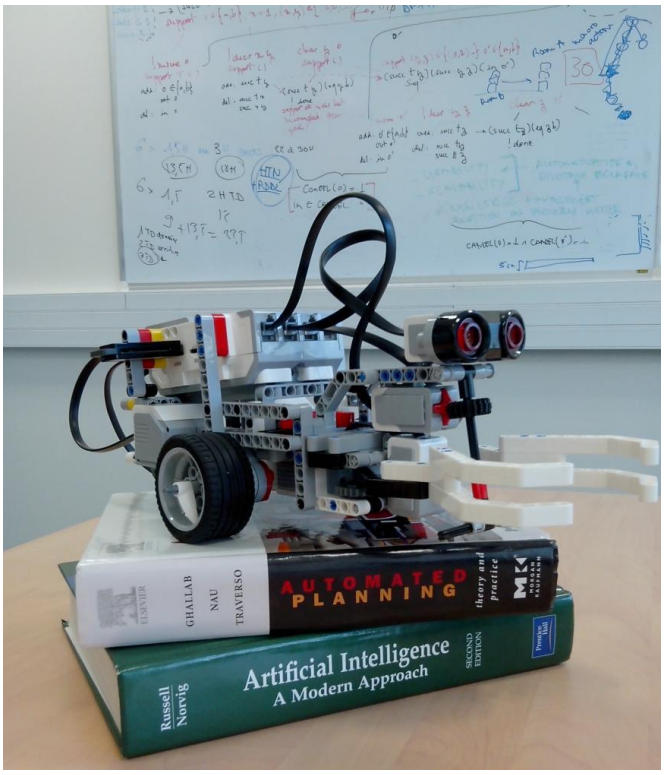

Figure 1 - LEGO EV3 Mindstorm

En ce qui concerne notre groupe, nous avons eu 12 semaines pour préparer cet évènement. Nous avons été mis en relation avec notre tuteur M. Stéphane CURABA pour nous épauler durant cette aventure.

Environnement

Le robot est livré avec une brique programmable. Le système d'exploitation installé dessus s'appelle Lejos et permet de lancer des applications codées en java directement sur cette brique. Nous avons choisi de travailler sous l'environnement Eclipse couplé au plugin Lejos for EV3 qui permet d'importer toute la librairie EV3.

Lejos for EV3

Ce plugin s'installe directement depuis Eclipse, rubrique « Install new software ». Eclipse se voit doté d'un nouveau menu d'options qui permet de paramétrer la communication avec le robot.

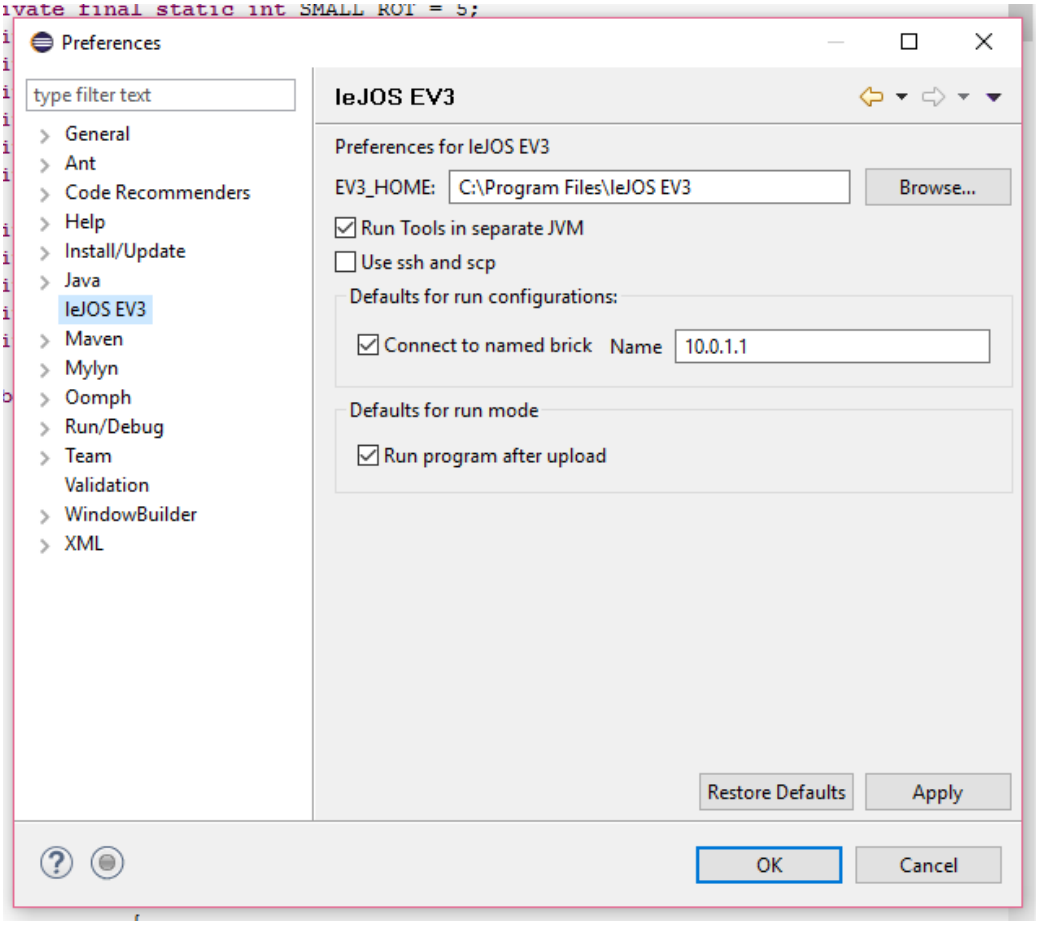

Figure 2 - Menu option LEJOS EV3

Stratégie

Le robot doit interagir avec son environnement de manière autonome. On parle de programmation multi-agents. La stratégie à mettre en place consiste à définir le comportement que le robot doit adopter en fonction de cet environnement. On doit dans un premier temps définir quel est le but premier du robot.

Quel est le but premier du robot dans notre cas ?

- ⇨ Gagner le match

Cela revient à définir une stratégie pour gagner un match. On doit d'abord s'informer sur le déroulement d'un match. En effet, quelles sont les règles d'un match ?

Règles du match

Le terrain

C'est une surface rectangulaire de 3 mètres sur 2 mètres. Il est entouré d'une bordure rigide d'une hauteur de 15cm. Les zones d'en-but sont d'une profondeur de 30cm. Chaque zone délimitée par des lignes de couleur a une dimension de 50cm sur 60cm. Les « R » représentent les positions initiales possibles de chaque robot.

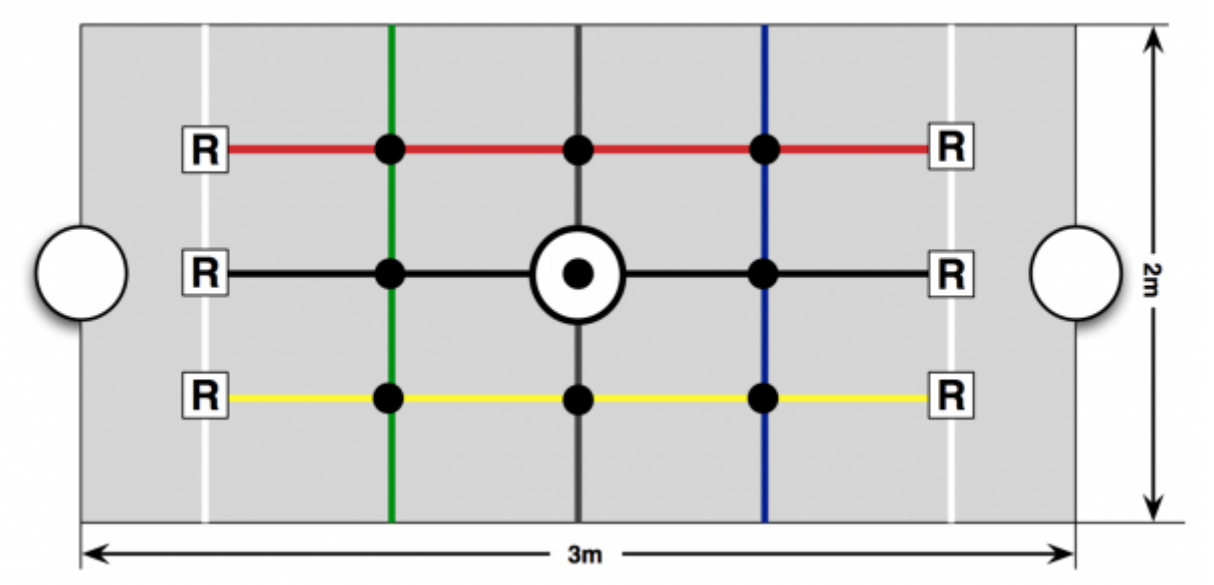

Figure 3 - Terrain de jeu

Cette année, les règles ont changé et ajoutent cette fois une caméra dont le champ de vision surplombe le terrain. Elle permet de connaître la position des palets sur le terrain.

Caméra

Elle envoie constamment la position de tous les palets situés sur le terrain. L'envoie se fait par paquets UDP.

- ⇨ Etablir une connexion UDP avec la caméra
- ⇨ Récupérer les paquets de la forme :
  - IDPalet;PosX;PosY;
  - Tous les palets sont donnés dans un seul paquet
- ⇨ Créer une structure Palet (classe Palet.java) regroupant :
  - l'identifiant du Palet
  - sa coordonnée en X
  - sa coordonnée en Y.

```
17          int port = 8888;
18
19          // Create a socket to listen on the port.
20          DatagramSocket dsocket = new DatagramSocket(port);
21
22          // Create a buffer to read datagrams into. If a
23          // packet is larger than this buffer, the
24          // excess will simply be discarded!
25          byte[] buffer = new byte[2048];
26
27          // Create a packet to receive data into the buffer
28          DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
29
30          // Now loop forever, waiting to receive packets and printing them.
31          while (true)
32          {
33            // Wait to receive a datagram
34            dsocket.receive(packet);
35
36            // Convert the contents to a string, and display them
37            String msg = new String(buffer, 0, packet.getLength());
38            //System.out.println(packet.getAddress().getHostName() + ": "
39            //    + msg);
40
41            String[] palets = msg.split("\n");
42
43            for (int i = 0; i < palets.length; i++)
44            {
45                String[] coord = palets[i].split(";");
46                int x = Integer.parseInt(coord[1]);
47                int y = Integer.parseInt(coord[2]);
48
49                System.out.println(Integer.toString(x) + " / " + Integer.toString(x) );
50            }
51
52
53            // Reset the length of the packet before reusing it.
54            packet.setLength(buffer.length);
55          }
56
57        }
58        catch (Exception e)
59        {
60          System.err.println(e);
61        }
```

Figure 4 - Connexion UDP avec la caméra

Le but du match est simple : marquer plus de palets que le robot adverse.

Un match comprend trois manches de 5 minutes avec des pauses intermédiaires de 5 minutes. Au cours d'une pause, une équipe peut changer le programme de son robot.

Remarques

1) Comment fait-on la différence entre les Palets qui ont été marqués et les autres que l'on doit attraper ?

⇨ On rajoute un champ « noter » pour la structure Palet pour connaître l'état du Palet.
⇨ Un Palet est donc à présent représenté par :
- Sa coordonnée en X
- Sa coordonnée en Y
- Son état « noter » (booléen)

2) Comment choisir son repère ?

Le repère choisi par la caméra est cartésien. On doit donc adapter toutes les méthodes à ce type de repère. En effet, les primitives données par la librairie EV3 se placent dans un repère polaire.

⇨ Changement de repère (polaire -> cartésien)

Mais, où se trouve l'origine du repère sur le terrain ? On peut se demander quel point du terrain sera désigné comme origine pour notre repère. Il faut donc prévoir ce détail (qui a une très grande importance).

⇨ Au début de la partie, choisir l'origine du repère

3) Peut-on marquer plusieurs fois un même Palet ?

Réponse : Non, on peut rapprocher cette règle avec celles du football par exemple.

Machine à états

Le robot devra questionner la caméra pour connaître la position des palets. Il faut maintenant savoir vers lequel il doit se diriger. On doit réfléchir à toutes les situations possibles dans lesquelles le robot peut se trouver. Cela revient à dessiner un automate pour toutes les transitions possibles d'une situation à une autre.

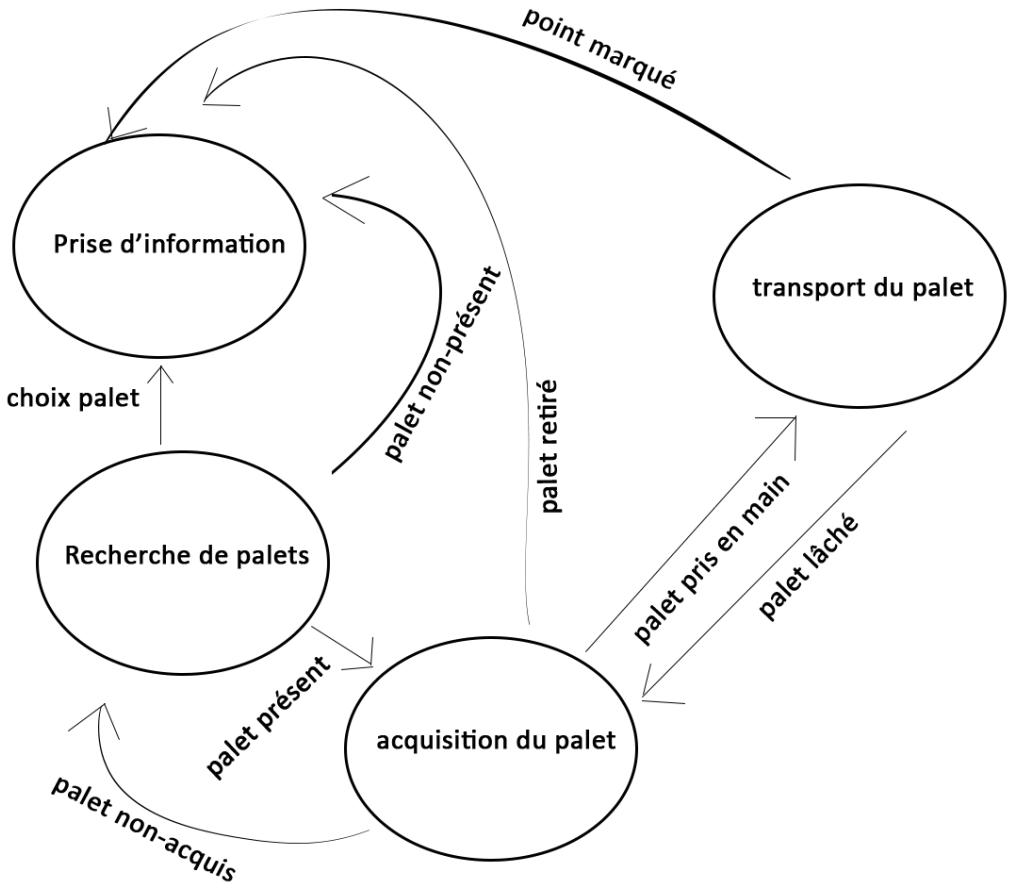

Figure 5 - Automate stratégie

Conclusion

Nous sommes assurés que notre robot agit selon la stratégie que nous avons établie. Cependant, n'ayant aucun robot adverse pour simuler un match, nous ne sommes pas en mesure de vérifier le déroulement de cette stratégie en conditions réelles.

Perspectives

Gagner le tournoi !

Plus sérieusement, on dispose d'à peu près 1 mois jusqu'au début de la compétition. Cela nous laisse encore le temps de faire des tests sur notre stratégie notamment en situation de match. Il faudra donc essayer d'acquérir un second robot pour effectuer ces tests.

Sources

Site officiel de la compétition : http://persycup.imag.fr

Site de LEJOS : http://www.lejos.org/ev3.php

Site du constructeur LEGO : http://lego.com/fr-fr/mindstorms/about-ev3