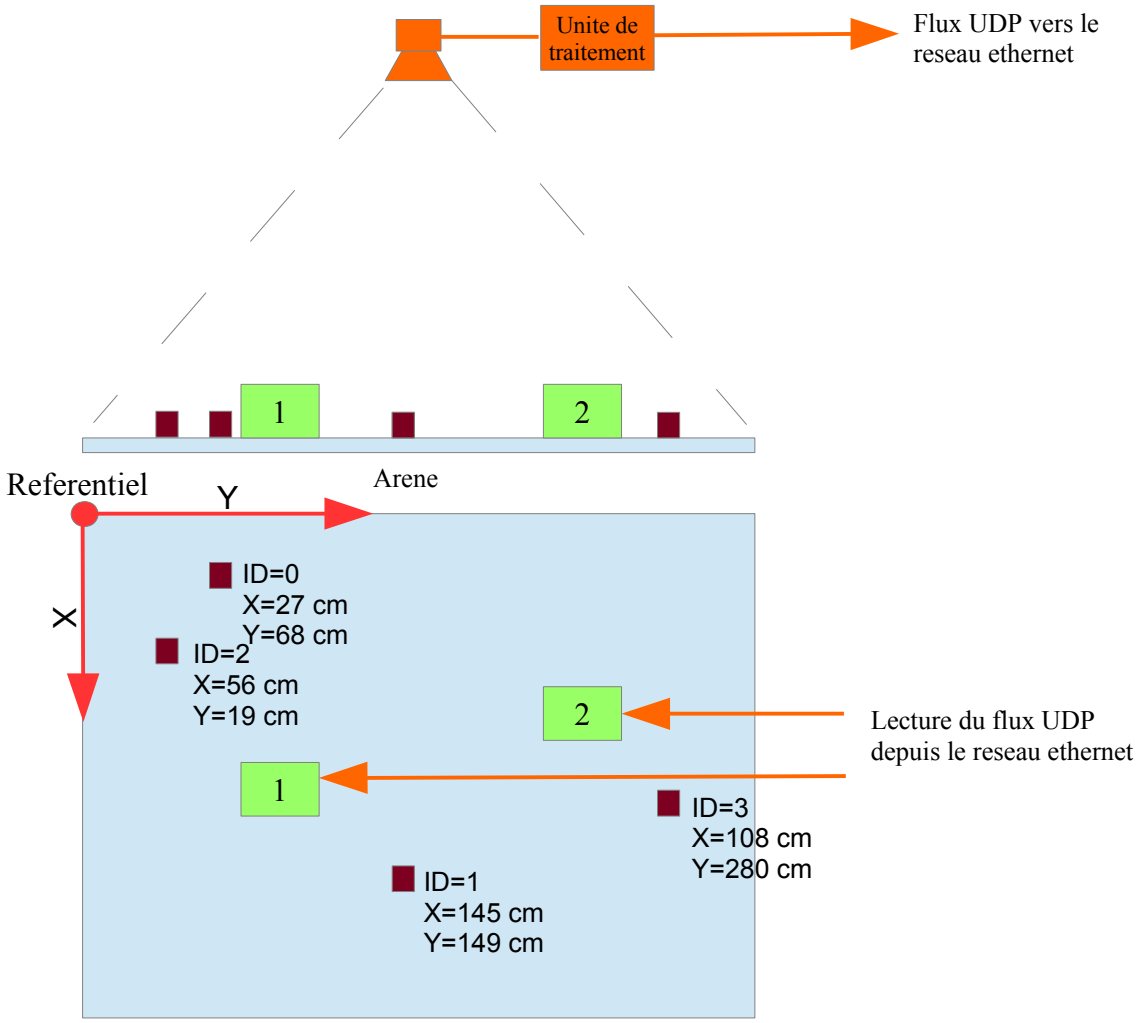
PROTOCOLE DE SUIVI DES PALETS

Principe

L'arène de la ligue Mindstorms est équipée d'une camera en surplomb qui détecte pendant les matchs, en temps réel, les positions des palets et diffuse les diffuse via un flux UDP lisible par les robots situes sur l'aire de jeu. Ceux-ci peuvent donc les exploiter dans leur stratégie si ils le souhaitent.



Choix de la technologie de détection

La camera est couple a un projecteur a leds, tous 2 infra-rouges. La lumière IR émise par le projecteur est reflétée par les palets car ceux-ci sont équipes de stickers réfléchissant et est renvoyée vers la camera. Celle-ci voit donc les palets comme des spots lumineux et peut aisément les distinguer dans l'image, quelque soit la lumière ambiante.

Structure des donnees

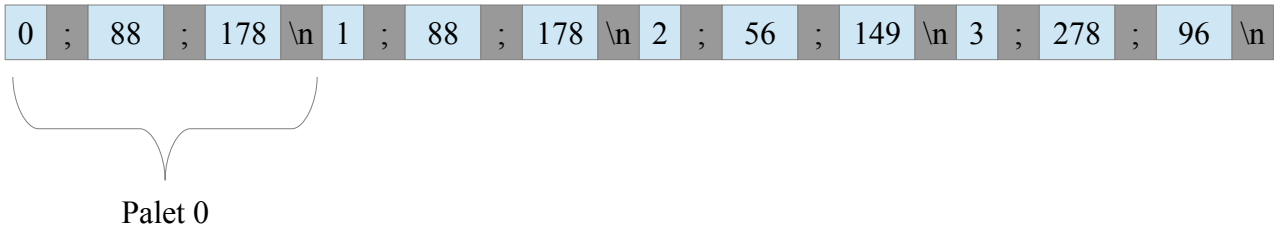
Le flux UDP diffuse en continu une trame de données **T** contenant les informations sérialisées des 9 palets et des 2 robots tant qu'ils sont présents sur l'aire de jeu, soit 11 structures séparées par des caractères de retours a la ligne.

Chaque structure contient 3 éléments séparés par des points virgules :

- son ID (entier) (index arbitraire)
- son abscisse sur la table de jeu, exprimée en centimètres (entier)
- son ordonnée sur la table de jeu, exprimée en centimètres (entier)

L'origine du référentiel est l'angle de la table matérialisé repère 2 flèches rouges.

Exemple de trame **T** :



Propriétés du flux

Point d'accès wifi pour connecter les robots : SSID « PERSYCUP » / password « persycup » (mode DHCP)

Le flux UDP est émis en broadcast sur le sous réseau 192.168.1.255 sur le port 8888.

Il y a également un flux de test accessible par un navigateur a l'adresse http://192.168.1.1:8080

PERSYVAL-Lab

Exemple du code de récupération des données

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class EV3Client {
  public static void main(String args[])
  {
    try
    {
      int port = 8888;

      // Create a socket to listen on the port.
      DatagramSocket dsocket = new DatagramSocket(port);

      // Create a buffer to read datagrams into. If a
      // packet is larger than this buffer, the
      // excess will simply be discarded!
      byte[] buffer = new byte[2048];

      // Create a packet to receive data into the buffer
      DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

      // Now loop forever, waiting to receive packets and printing them.
      while (true)
      {
        // Wait to receive a datagram
        dsocket.receive(packet);

        // Convert the contents to a string, and display them
        String msg = new String(buffer, 0, packet.getLength());
                        System.out.println("packet.getLength = " + packet.getLength() + " msg = " + msg);

        String[] palets = msg.split("\n");

        for (int i = 0; i < palets.length; i++)
        {
              String[] coord = palets[i].split(";");
              if(coord.length >2)
              {
                        int x = Integer.parseInt(coord[1]);
                        int y = Integer.parseInt(coord[2]);
                        System.out.println(i + " / " + Integer.toString(x) + " / " + Integer.toString(y) );
              }
        }

        // Reset the length of the packet before reusing it.
        packet.setLength(buffer.length);
        packet.setData(buffer);
      }
      dsocket.close();

    }
    catch (Exception e)
    {
      System.err.println(e);
    }
  }
}
```

T\l\charger ce document

web

https://www.dropbox.com/s/7sr2zovodsfnse5/Protocole%20Suivi%20Palets.pdf?dl=0