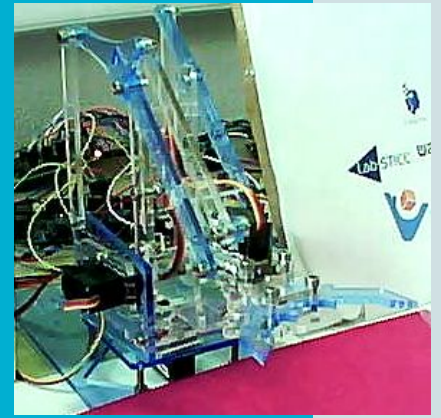




LEAP
M O T I O N



Ismael Mkenini
Harshil Sethi
Mohamed-Mehdi Adlal
Emmanuel Chêne



Iut Nice Sophia Antipolis
LP SIL IDSE

Objectifs :

Contrôler à distance les mouvements et gestes d'un bras robotisé équipé de ServoMoteur à partir d'un leapMotion.

-Support technique : Couplage leap Motion + Connexion Internet + Raspberry Pi/Robot Pince Arduino.

Exemple d'utilisation :

Rotation gauche :



NICE

Internet

Rotation gauche :



BREST

Répartition des tâches :

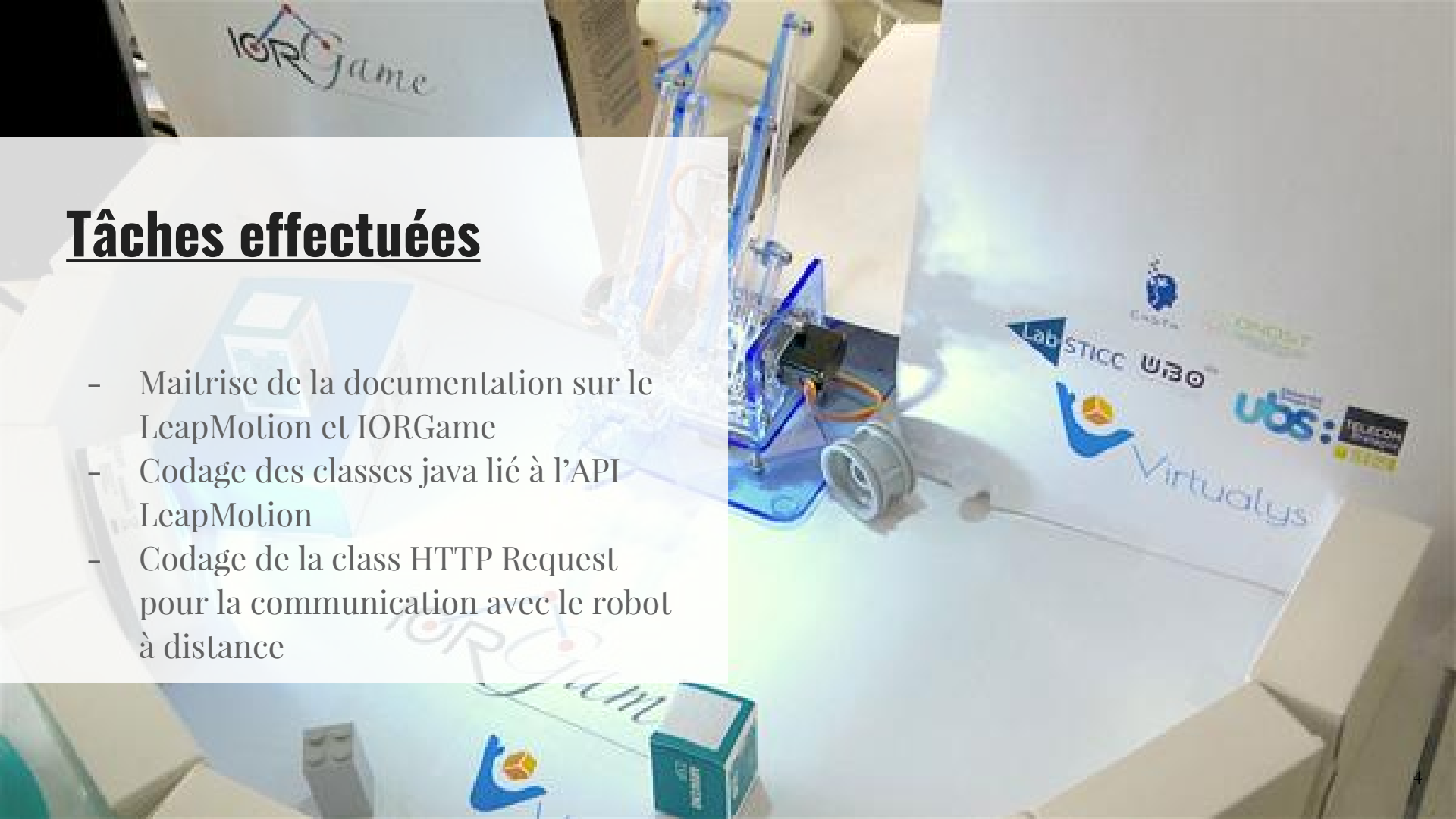
Les tâches ont été réparties en 2 groupe de 2 personnes :

- Un groupe qui s'est occupé du LeapMotion:
 - Tester les récupération des mouvements du leap motion.
 - Associer les commandes du leap motion au commandes du bras robotisé

- Un autre qui s'est occupé de l'envoi des trames au robot
 - Traitement des informations reçues du LeapMotion,
 - Exécution des actions qui ont été auparavant traitées.

Tâches effectuées

- Maitrise de la documentation sur le LeapMotion et IORGame
- Codage des classes java lié à l'API LeapMotion
- Codage de la class HTTP Request pour la communication avec le robot à distance



Fonctionnement général

Le LeapMotion :

Le boîtier comprend une caméra style webcam ainsi que 3 LED infrarouges lui permettant une vue en relief. Les données collectées sont ainsi transmises par câble USB au pc. Un programme adéquat pourra lire les données, les traduire et les envoyer à la main robotisé par bluetooth.

IORGame :

Depuis un téléphone mobile ou une tablette, il est possible de prendre le contrôle d'un bras articulé à distance via internet. Il faut télécharger l'application sur le Google play store ou l'App Store.

Fonctionnement du bras robotisé

La main dispose des attributs qui permettent de réaliser des mouvements et des interactions avec l'application.

- L'origine du repère de la main permettant son suivi.
- Sa vitesse en mm/sec, la normal a la main.
- Un vecteur pointant du centre de la paume vers les doigts.
- Centre de la sphère.
- Sphère Distance (variation avec la forme de la main).

Le Leap ne distingue pas la main gauche de la main droite.

Il y a la possibilité de mettre plus de 2 mains mais il est conseillé d'en utiliser 2 au maximum.

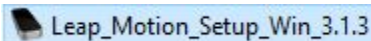
Configuration du kit de développement logiciel leapMotion :

Pré Requis :

-LeapDeveloperKit.zip ➞ <https://developer.leapmotion.com/get-started>

DOWNLOAD ORION BETA

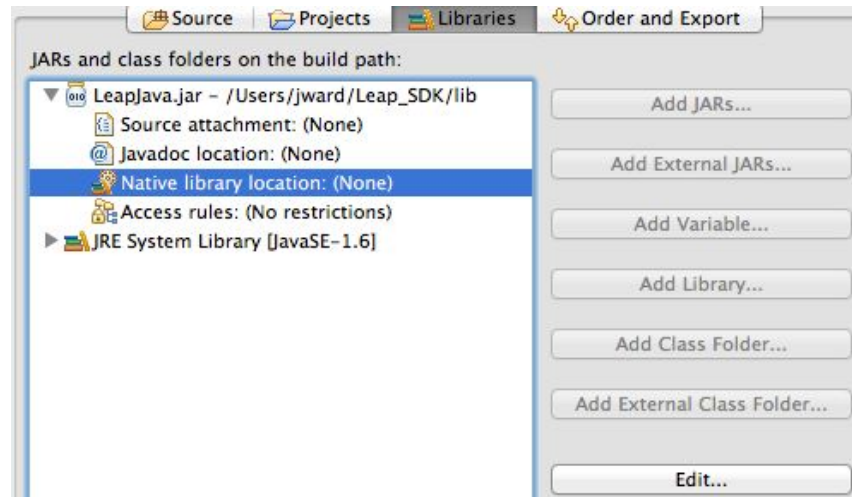
-Leap_Motion_Setup_Win_3.1.3.exe



Mise en place du projet :

-Importer la librairies LeapJava.jar dans le projet java

-Sélectionnez la bibliothèque native - > **x32/x64**



Class Controller :

class com::leapmotion::leap::Controller

->interface principale du service Leap Motion.
->permet d'accéder aux cadres de données de suivi (Frame) et aux informations de configuration. Les données peuvent être interrogées à tout moment grâce à cette class.

Accéder aux objets Frame par une instance du Controller :

```
public void onFrame(Controller controller) {  
    Frame frame = controller.frame();
```

Class Frame :

class com::leapmotion::leap::Frame

La classe Frame représente l'ensemble de données de suivi de main et de doigt détectées par le leap Motion. Le logiciel Leap Motion détecte donc les mains, les doigts en signalant leurs positions, les orientations, les gestes et les mouvements dans les cadres à la cadence du Leap Motion.

Accéder aux objets Hand par une instance frame:

```
HandList hands = frame.hands();|
```

Accéder aux objets pointable par une instance frame:

```
Pointable pointable = frame.pointables().frontmost();
```


Class Listener :

class com::leapmotion::leap::Listener

- >Permet de gérer les événements Leap Motion.
- >Crée une instance d'une sous-classe Listener et on l'attribue à l'instance Controller

```
public class SampleListener extends Listener {  
  
    // On créer un sample listener et un controller  
    SampleListener listener = new SampleListener();  
    Controller controller = new Controller();  
  
    controller.addListener(listener);  
    // On demander ensuite au listener de recevoir des événements du contrôleur  
    Listener objListener = new Listener();  
    objListener.onConnect(controller);  
}
```

Class Hand :

class com::leapmotion::leap::Hand

- >Rapporte les caractéristiques physiques d'une main détectée

Créer une instance de main Droite et main Gauche avec l'objet Frame :

```
Hand handRight = frame.hands().rightmost();  
Hand handLeft = frame.hands().leftmost();
```

Une main est décrite par :

- Sa position
- Son orientation
- Sa posture
- Son mouvement

Exemple de mouvement :



SwipeGesture



CircleGesture



ScreenTapGesture



KeyTapGesture

Récupère les doigts

```
Fingers: " + frame.fingers().count()
```

```
PointableList pointables = hand.pointables();  
FingerList fingers = hand.fingers();
```

Getting the Fingers

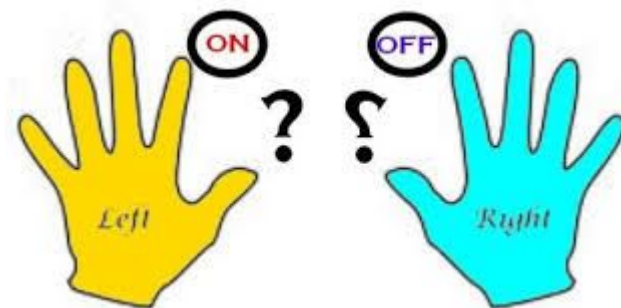
You can get the fingers associated with a hand as a list or individually using an ID obtained in a previous frame.

By list:

```
// hand is a Hand object  
List<Finger> fingers = hand.Fingers;
```

Caractéristiques :

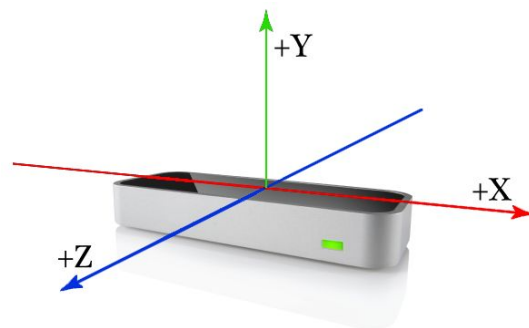
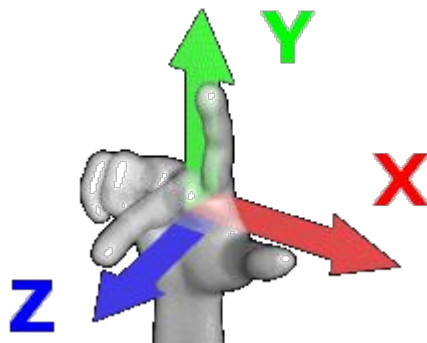
- `isRight`, `isLeft` — Whether the hand is a left or a right hand.
- `Palm Position` — The center of the palm measured in millimeters from the Leap Motion origin.
- `Palm Velocity` — The speed and movement direction of the palm in millimeters per second.
- `Palm Normal` — A vector perpendicular to the plane formed by the palm of the hand. The vector points downward out of the palm.
- `Direction` — A vector pointing from the center of the palm toward the fingers.
- `grabStrength`, `pinchStrength` — Describe the posture of the hand.



Palm Position: " + hand.palmPosition()

Palm Position: (73.3512, 50.9354, -27.085)

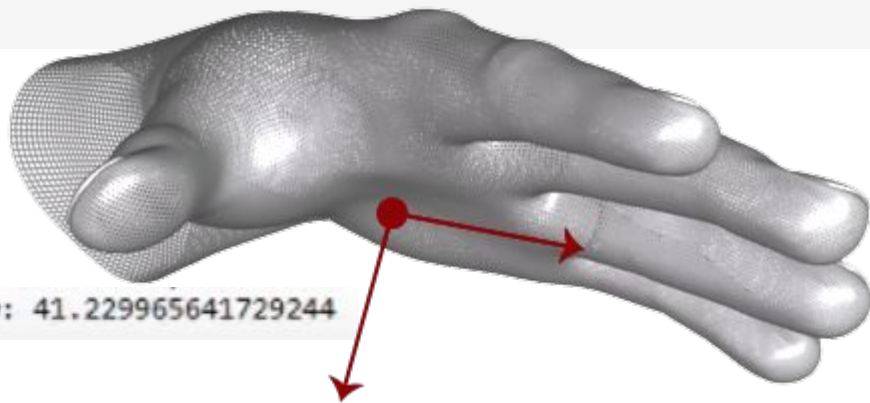
id: 70,



The [Vector](#) class defines functions for getting the pitch (angle around the x-axis), yaw (angle around the y-axis), and roll (angle around the z-axis):

```
float pitch = hand.Direction.Pitch;  
float yaw = hand.Direction.Yaw;  
float roll = hand.PalmNormal.Roll;
```

```
Vector normal = hand.palmNormal();  
Vector direction = hand.direction();
```



Pitch: 3.6484928211162893 Roll: 7.755115458169306 Yaw: 41.229965641729244

CODE :

```
try{
    if (frame.hands().count() != 0)
    {
        TimeUnit.SECONDS.sleep(1);
        if(handRight.palmPosition().getZ() > 5)
        {
            objHttp.SendCommande("RECULER");
            System.out.println("Reculer");
        }
        else if(handRight.palmPosition().getZ() < -5)
        {
            objHttp.SendCommande("AVANCER");
            System.out.println("Avancer");
        }
        else if(handRight.palmPosition().getZ() == 0)
        {
            System.out.println("Rien");
        }
    }

    if (pointable.isExtended() == true) {
        objHttp.SendCommande("OUVRIR");
        System.out.println("Poignée ouvert");
    }else if (pointable.isExtended() == false){
        objHttp.SendCommande("FERMER");
        System.out.println("Poignée fermer");
    }
}

if(handRight.palmPosition().getY() > 100.000){
    objHttp.SendCommande("MONTER");
    System.out.println("hauteur main : haut");
}else if(handRight.palmPosition().getY() < 100.000 && handRight.palmPosition().getY() > 0.000){
    objHttp.SendCommande("DESCENDRE");
    System.out.println("hauteur main : bas");
}
}
} catch(Exception e){
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

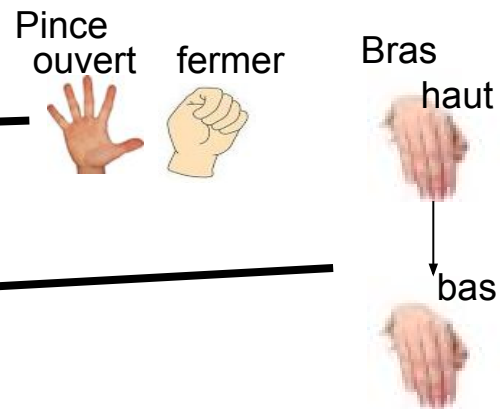
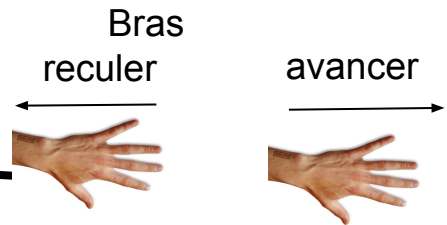
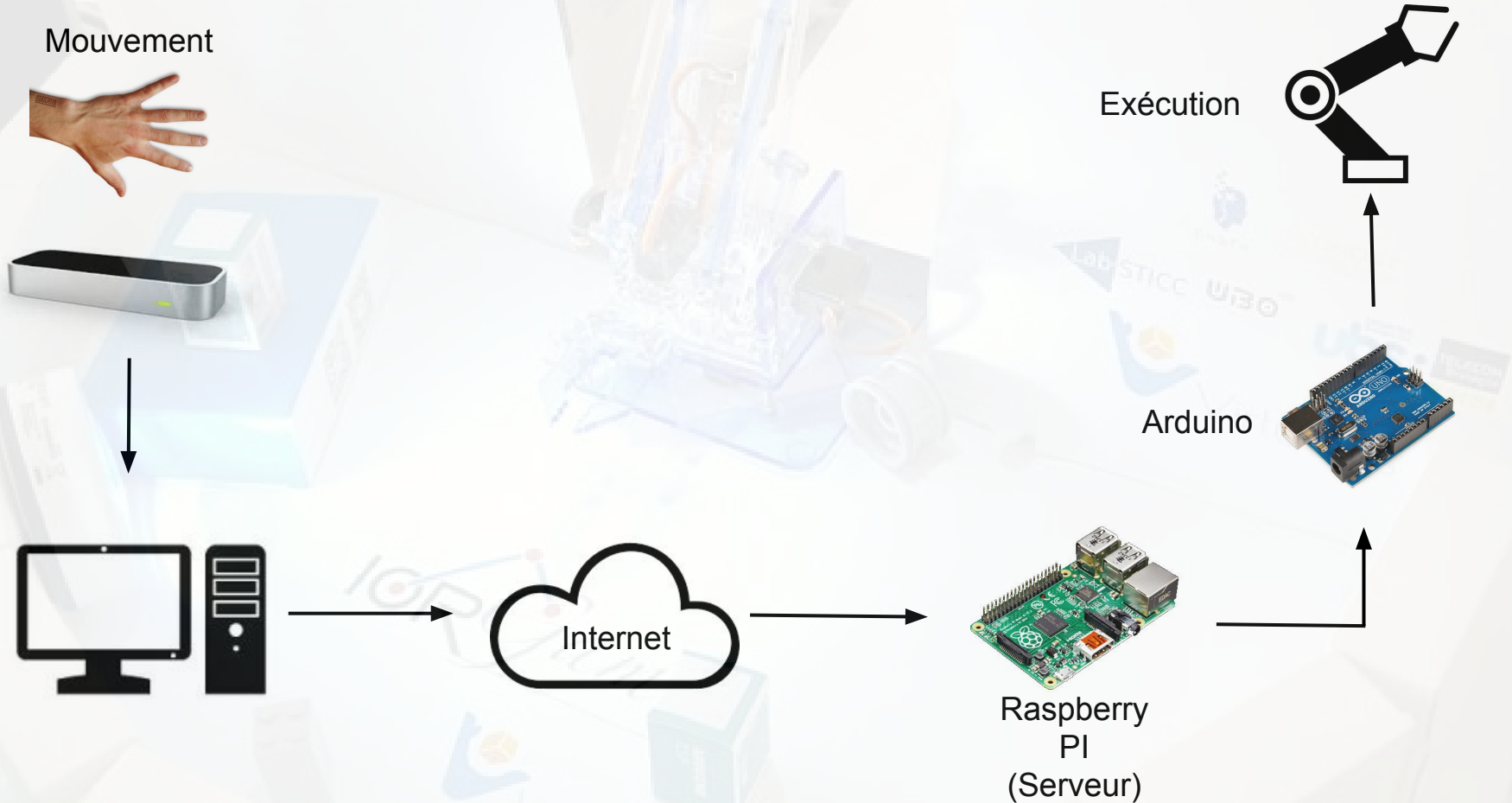


Schéma de communication :



Raspberry Pi

- Nano-ordinateur
- Sous distribution Linux
- Utilisé comme serveur web



Arduino

- Microcontrôleur programmable
- Analyse et produit des signaux électriques
- Communique avec le Raspberry Pi



Bras robotisé

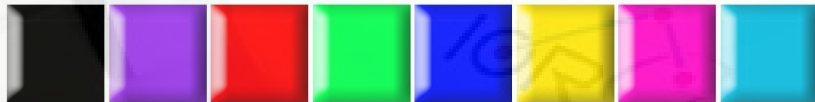
- 8 ordres différents
- Retour vidéo en direct (Caméra IP)
- Différents objets présents sur le plateau de jeu pour les manipuler

Organisation du travail

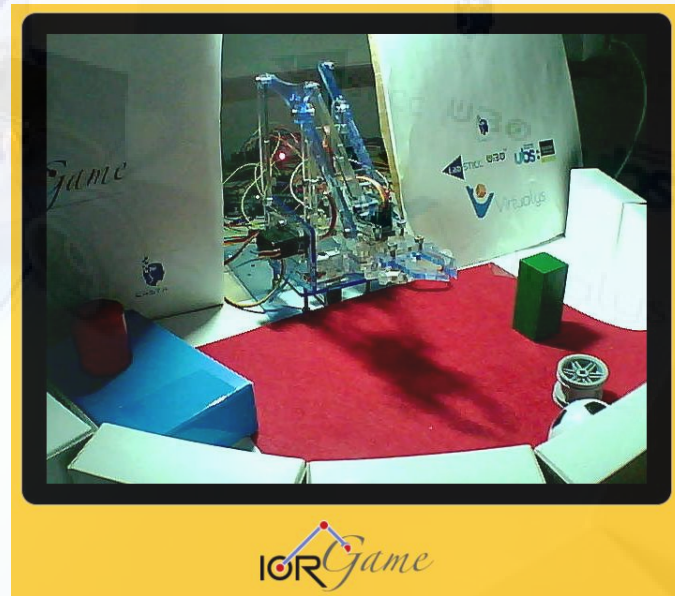
- Envoie de commandes pré-enregistrées
- Envoie de commandes avec le clavier
- Contrôle du bras grâce au LeapMotion

Outils à disposition

Interface Web



Caméra IP



Envoie des données

- Sous forme de requête HTTP :

<http://iotserver.univ-brest.fr/robot.php>

- 3 paramètres nécessaires

Avancer : $x_1=0?x_2=0?x_3=1$

Monter : $x_1=0?x_2=1?x_3=1$

Gestion des actions en Java

```
if (e.getKeyCode() == KeyEvent.VK_UP) {  
    url = "http://iotserver.univ-brest.fr/robot.php?x1=0&x2=0&x3=1";  
    commande = "UP";  
  
} else if (e.getKeyCode() == KeyEvent.VK_DOWN) {  
    url = "http://iotserver.univ-brest.fr/robot.php?x1=1&x2=0&x3=1";  
    commande = "DOWN";  
  
} else if (e.getKeyCode() == KeyEvent.VK_LEFT) {  
    url = "http://iotserver.univ-brest.fr/robot.php?x1=0&x2=1&x3=0";  
    commande = "LEFT";  
  
} else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {  
    url = "http://iotserver.univ-brest.fr/robot.php?x1=1&x2=1&x3=0";  
    commande = "RIGHT";  
  
} else if (e.getKeyCode() == KeyEvent.VK_SPACE) {  
    url = "http://iotserver.univ-brest.fr/robot.php?x1=0&x2=1&x3=1";  
    commande = "SPACE";  
  
} else if (e.getKeyCode() == KeyEvent.VK_CONTROL) {  
    url = "http://iotserver.univ-brest.fr/robot.php?x1=1&x2=1&x3=1";  
    commande = "CONTROL";  
  
}
```

Envoie des URL en Java

```
URL obj = new URL(URL);
URLConnection con = (URLConnection) obj.openConnection();

//Add request header
con.setRequestMethod("POST");
con.setRequestProperty("User-Agent", USER_AGENT);
con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");

// Send request
con.setDoOutput(true);
DataOutputStream wr = new DataOutputStream(con.getOutputStream());
wr.flush();
wr.close();


int responseCode = con.getResponseCode();
System.out.println("\nSending 'POST' request to URL : " + URL);
System.out.println("Post parameters : " + order);
System.out.println("Response Code : " + responseCode);
```


Gestions des actions en Android

```
public void robotright(View v) {  
    robotAction("110");  
    System.out.println("robotAction(\"110\");");  
}  
  
public void robothlu(View v) {  
    robotAction("001");  
    System.out.println("robotAction(\"001\");");  
}  
  
public void robothld(View v) {  
    robotAction("101");  
    System.out.println("robotAction(\"101\");");  
}  
  
public void roboth2u(View v) {  
    robotAction("011");  
    System.out.println("robotAction(\"011\");");  
}
```


Envoie des URL en Android

```
public void robotAction(String s) {  
    try {  
        URL curl = new URL(this.url + "robot.php?x1=" + s.charAt(0) + "&x2=" + s.charAt(1) + "&x3=" + s.charAt(2));  
        for (int i = 0; i < 1; i++) {  
            new WebCommand().execute(new URL[]{curl, null, null});  
        }  
    } catch (MalformedURLException e) {  
    }  
}
```



Problèmes rencontrés