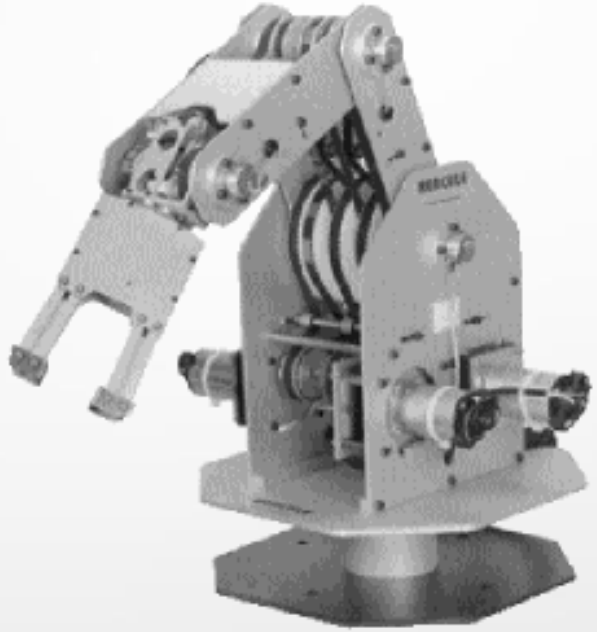
BRAS ROBOTIQUE HERCULE 2000



Alexandre Filipe

SOMMAIRE

- Présentation du projet
- Cahier des charges
- Présentation des diagramme UML commun
- Présentation des diagrammes UML personnel
- Matériels utilisés
- Choix des OS, IDE et langages utilisés
- Diagramme de gantt
- Présentation de l'IHM
- Questions

PRESENTATION DU PROJET

- Manipulation de produits potentiellement dangereux
- Deux modes de pilotage
  - -Poste de pilotage (souris , joystick), visible avec la camera.
  - -Tablette
- Gestion de la priorité

CAHIER DES CHARGES

PC de commande :

- IHM Test
- Mode apprentissage
- Mode automatique
- Serveur
- Gestion de la priorité
- Base de données

CAHIER DES CHARGES

PC de pilotage :

- IHM
- Client
- Coder le joystick
- Visionner le robot via la caméra
- Conversion des données

CAHIER DES CHARGES

Tablette de pilotage :

- Application tablette :
  - -Mode gyroscopique
  - -Mode commande
- Client

DIAGRAMME DE DEPLOIEMENT

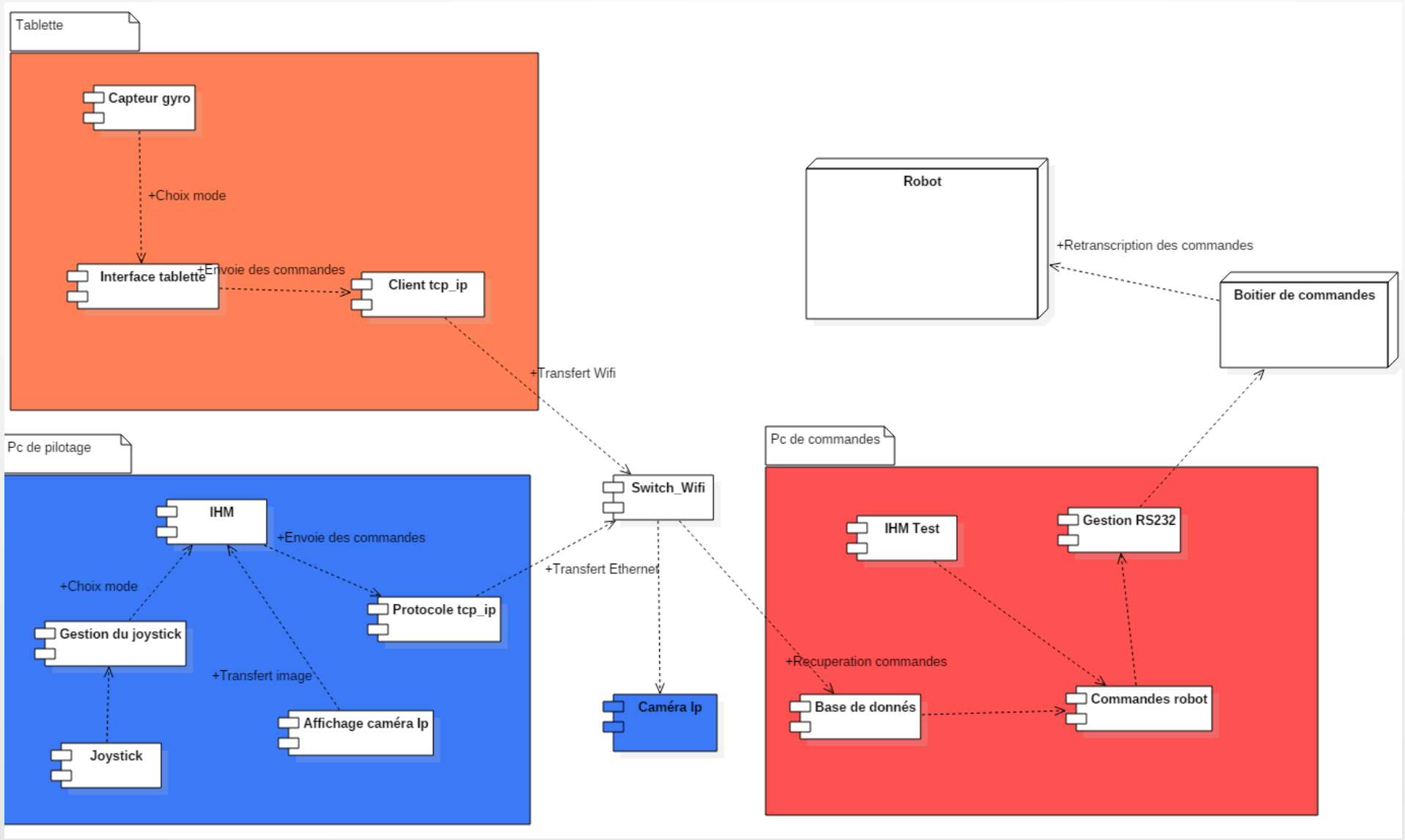


DIAGRAMME DE CAS D'UTILISATION

Générale :

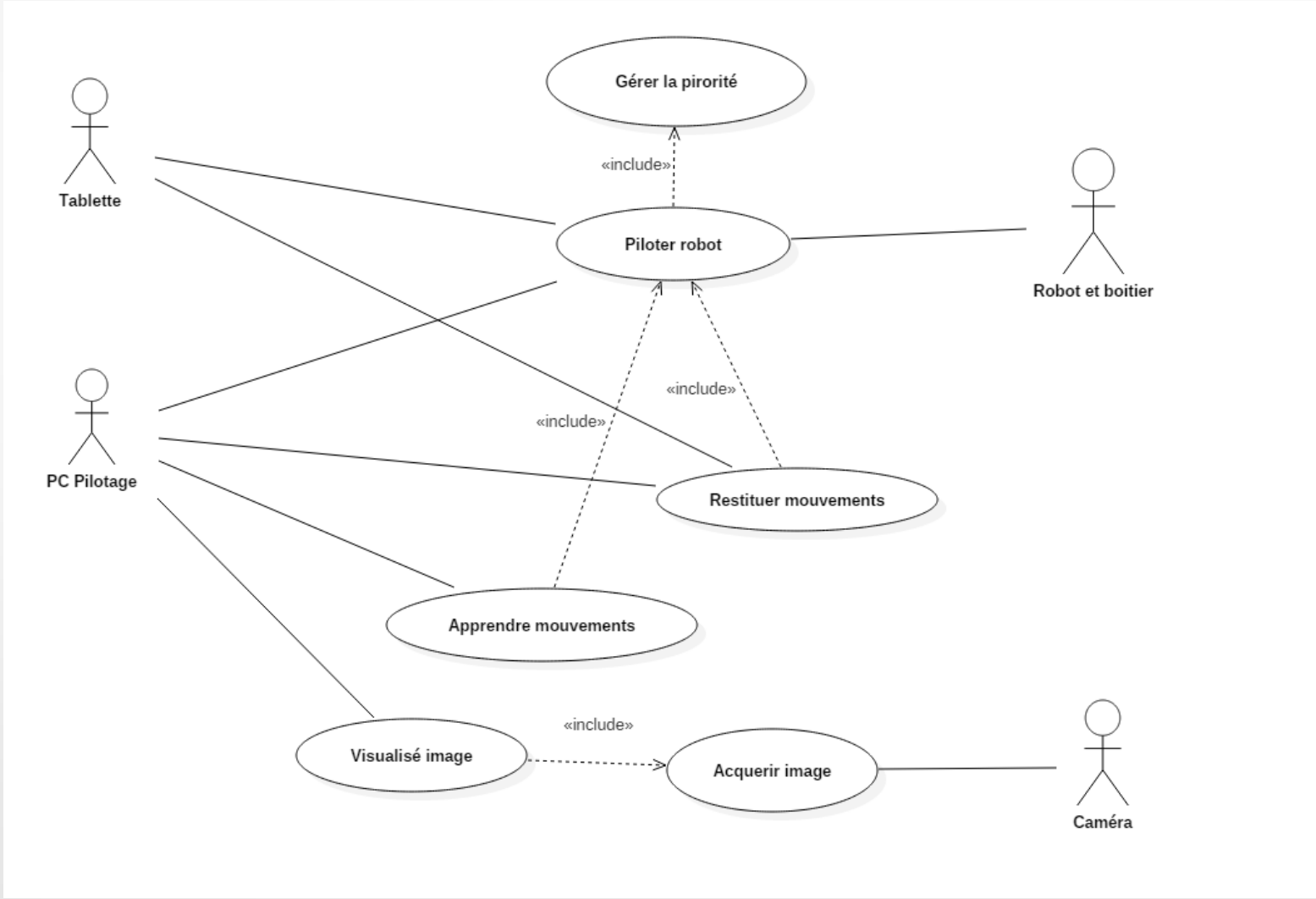


DIAGRAMME DE CAS D'UTILISATION

PC commande :

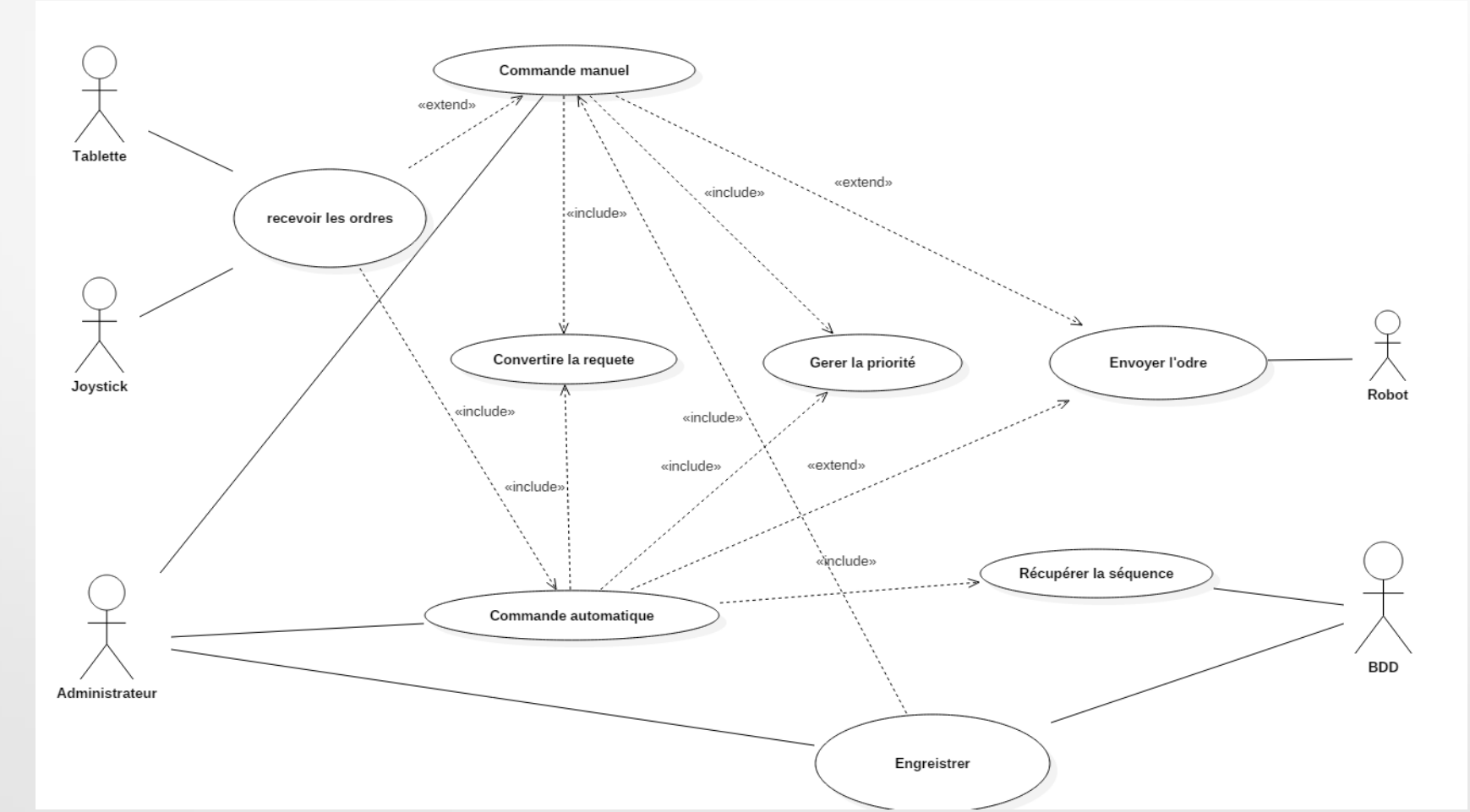


DIAGRAMME DE SEQUENCE

PC commande :

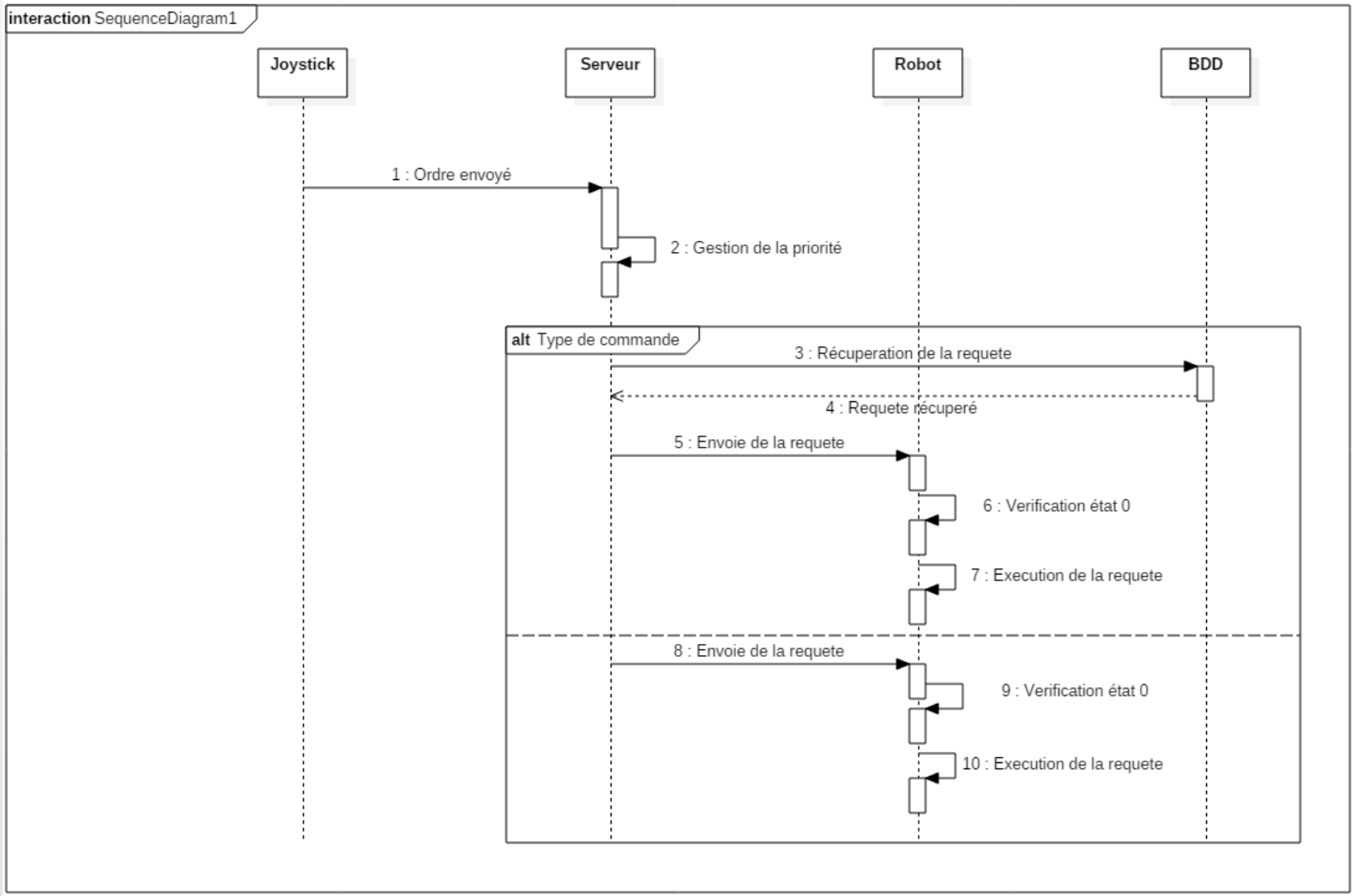


DIAGRAMME DE SEQUENCE

Base de données:

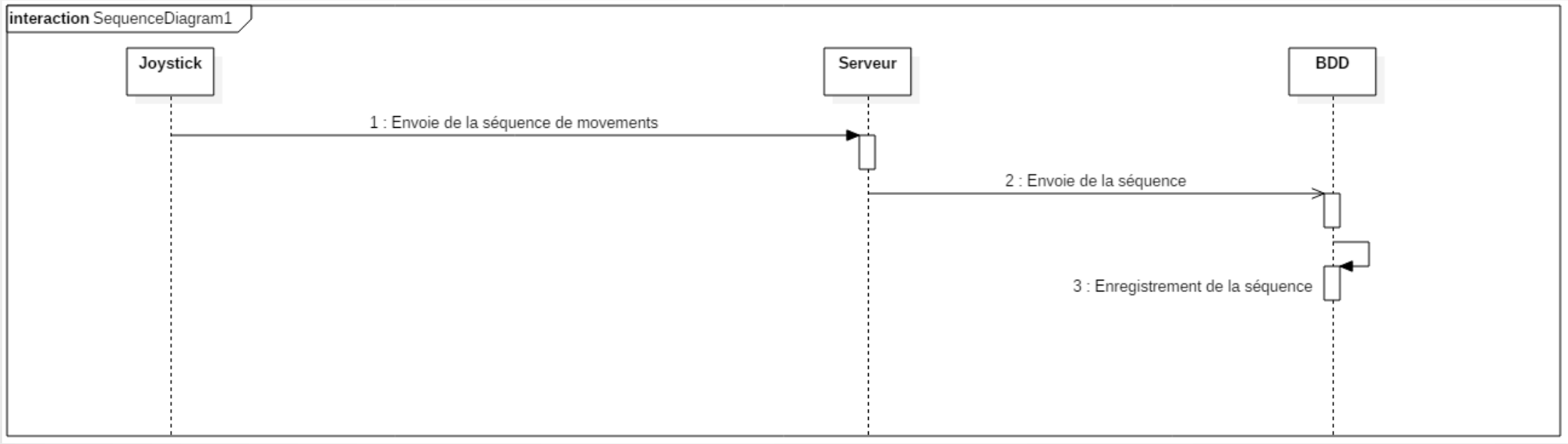


DIAGRAMME DE TABLE SQL

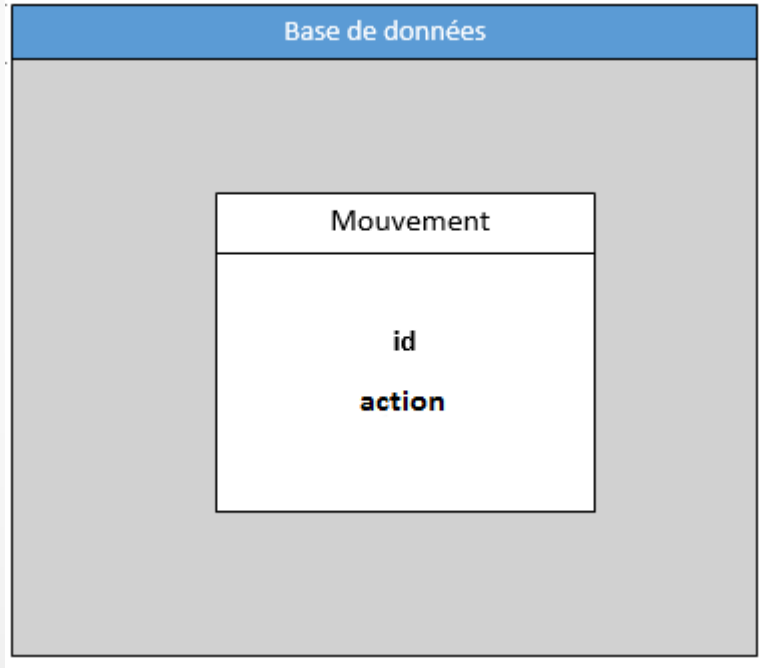
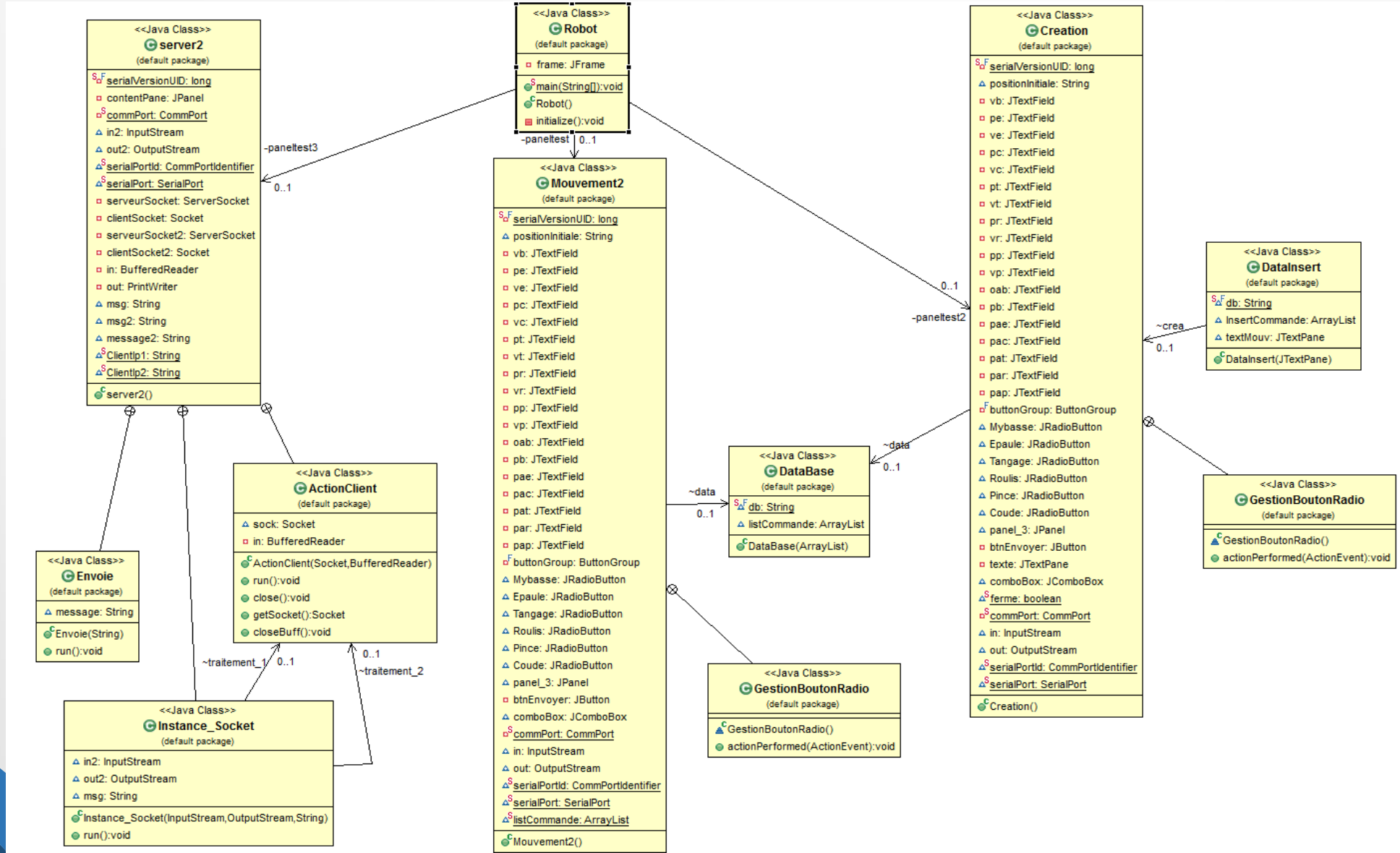
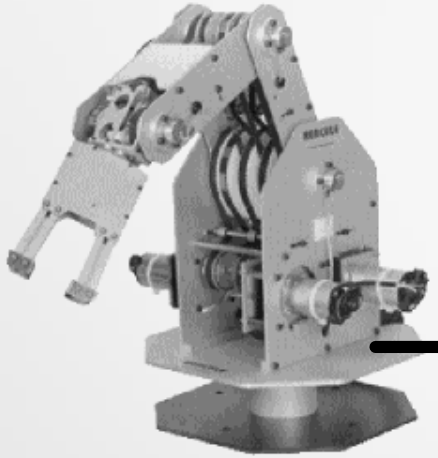


DIAGRAMME DE SEQUENCE



MATERIELS UTILISES.



Câble RS323

CHOIX DES OS, IDE ET LANGAGES CHOISIS.

OS :

Langages :

IDE :

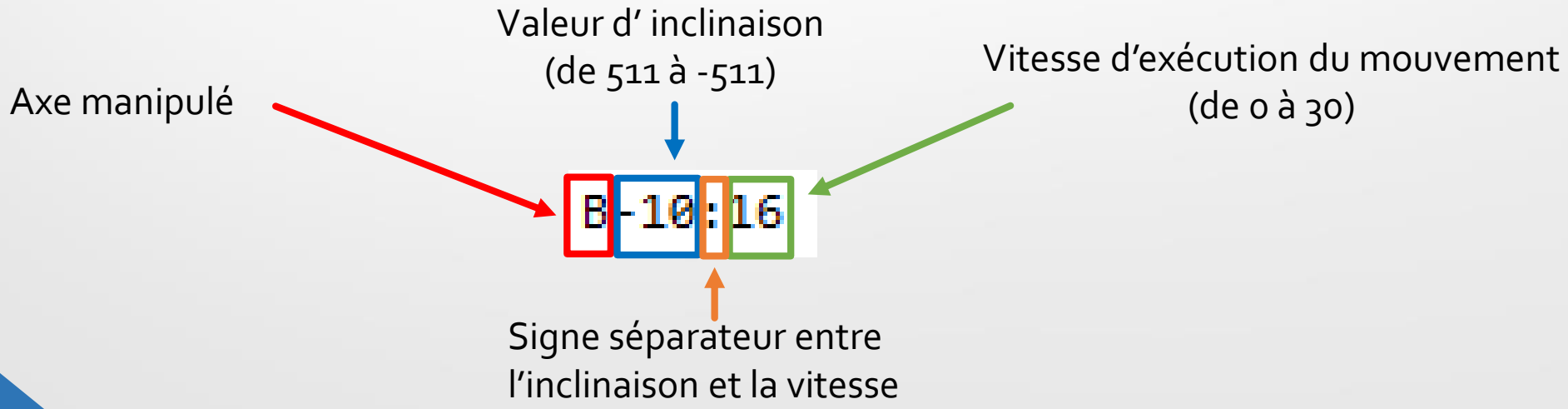
PROTOCOLE D'ECHANGE DE DONNEES

- Protocole TCP

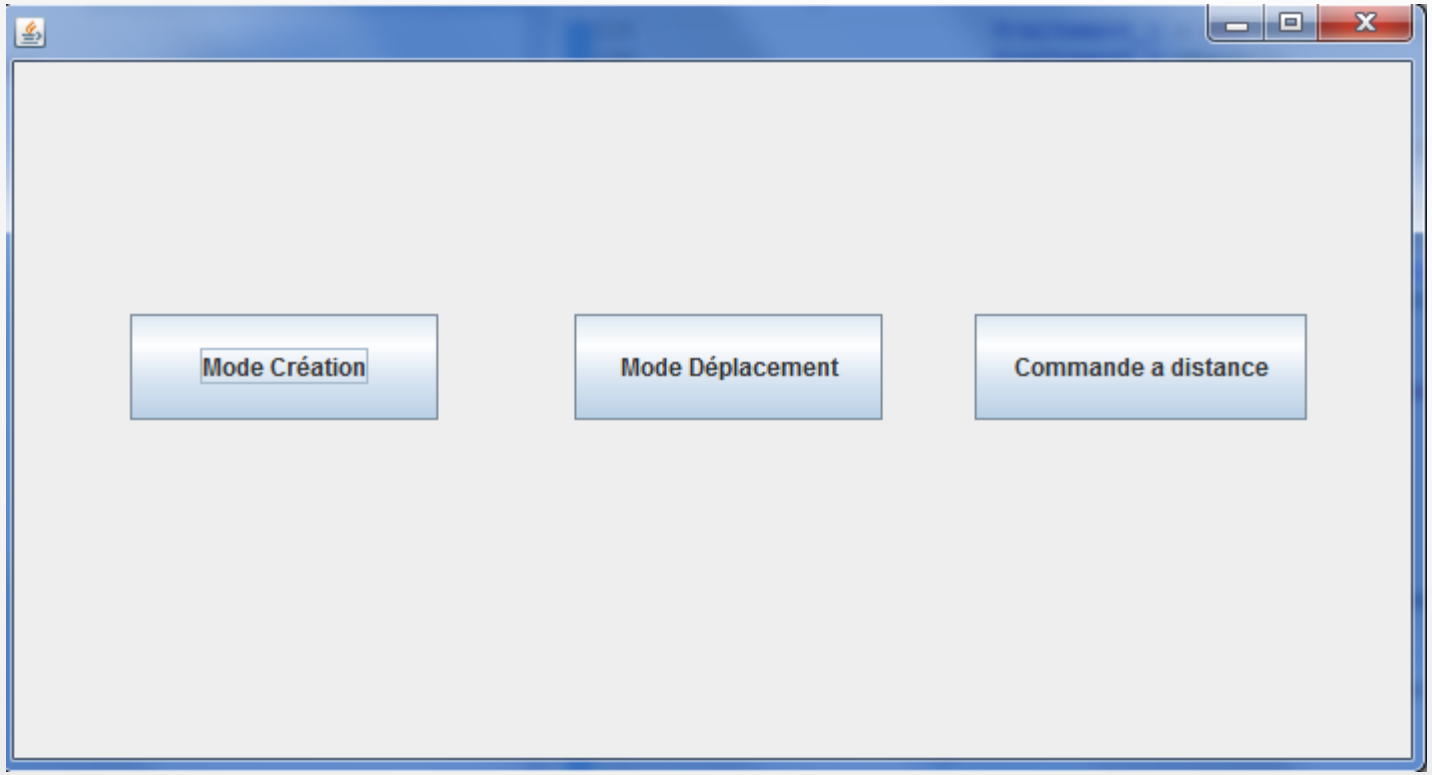
Exemple de données reçu :

```
valeur1 recu: Loic
valeur recu: B-10:16
valeur1 recu: B-10:16
valeur recu: B-10:16
```

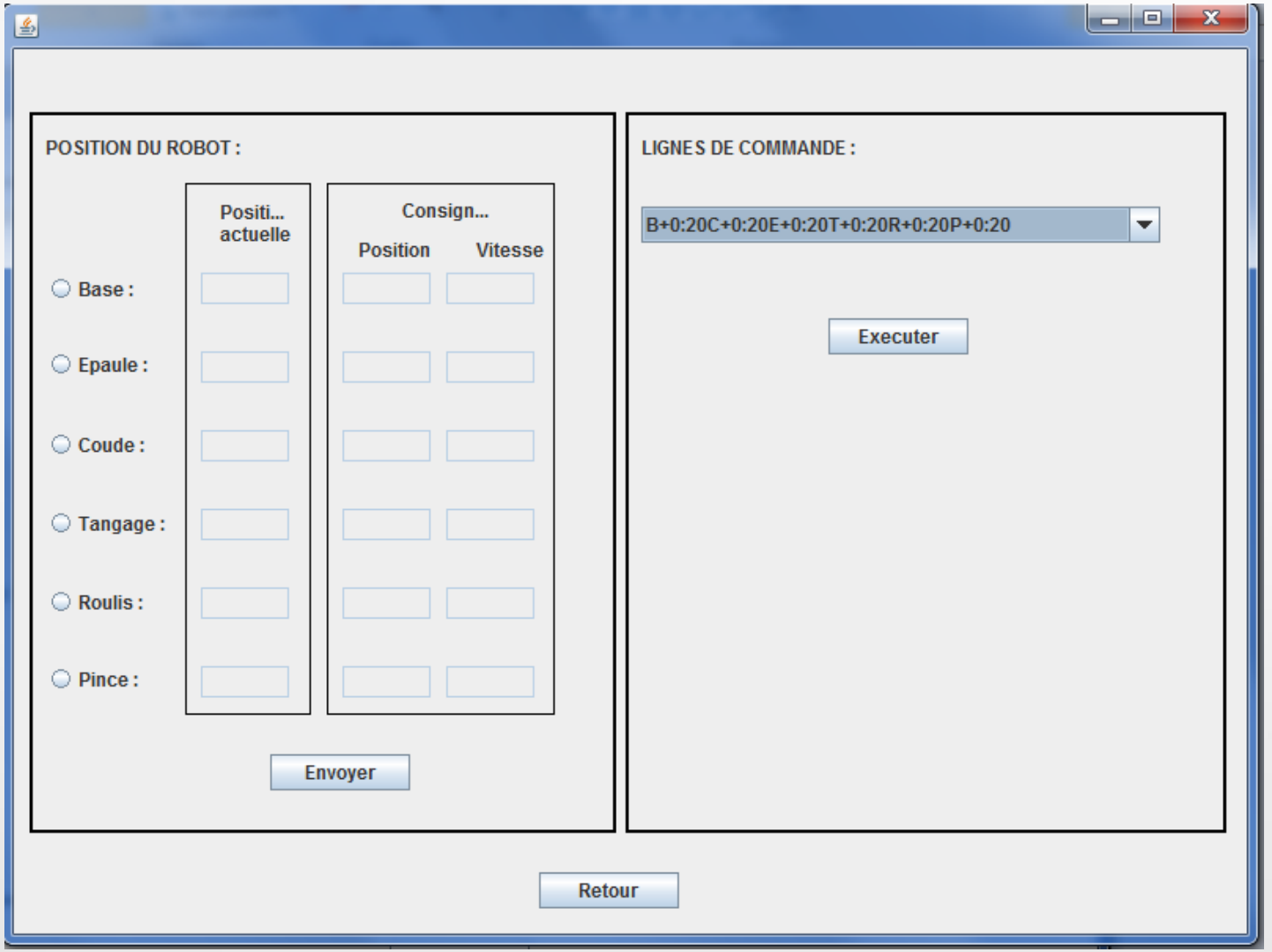
Explication :



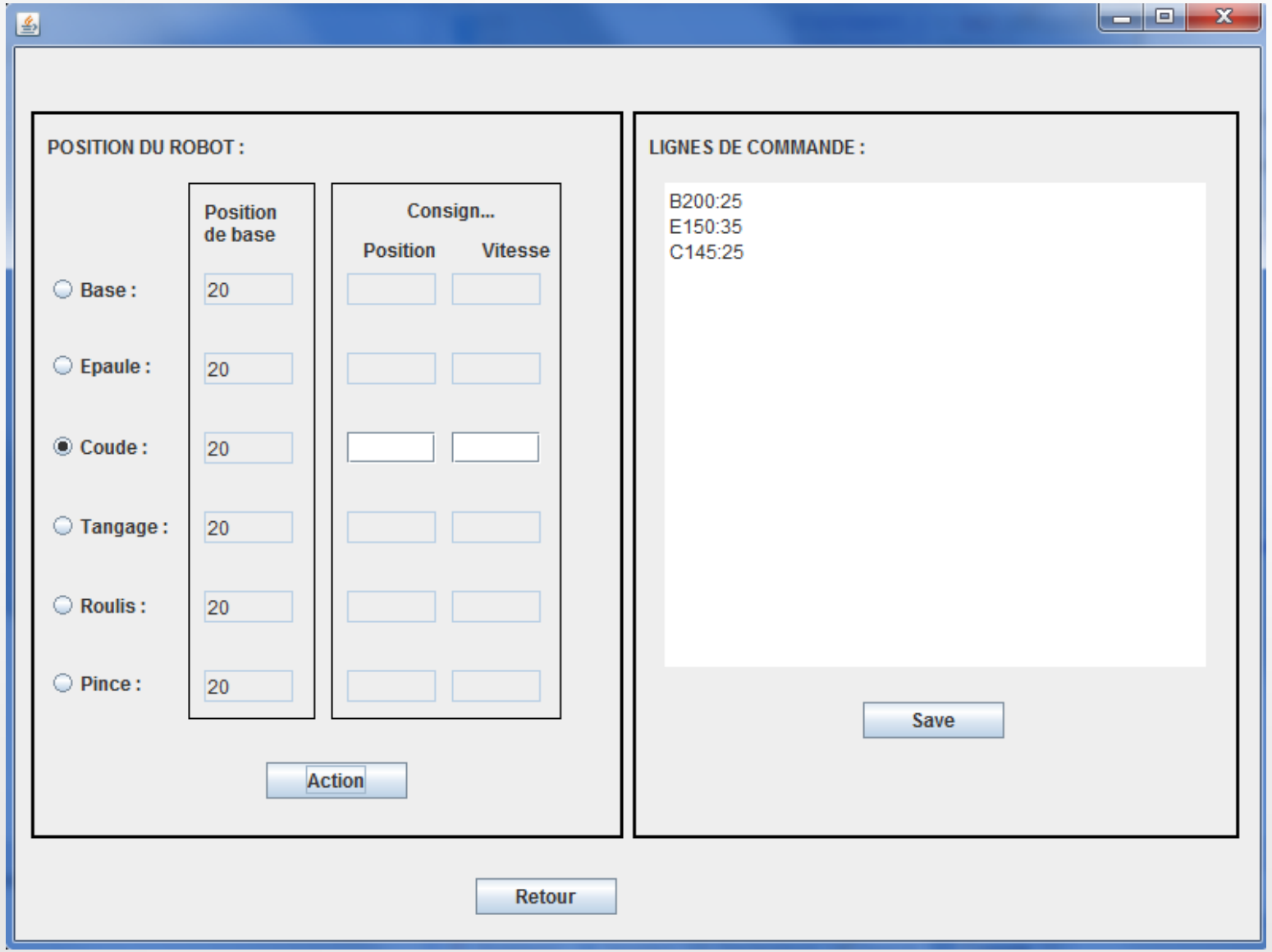
PRESENTATION DE L'IHM



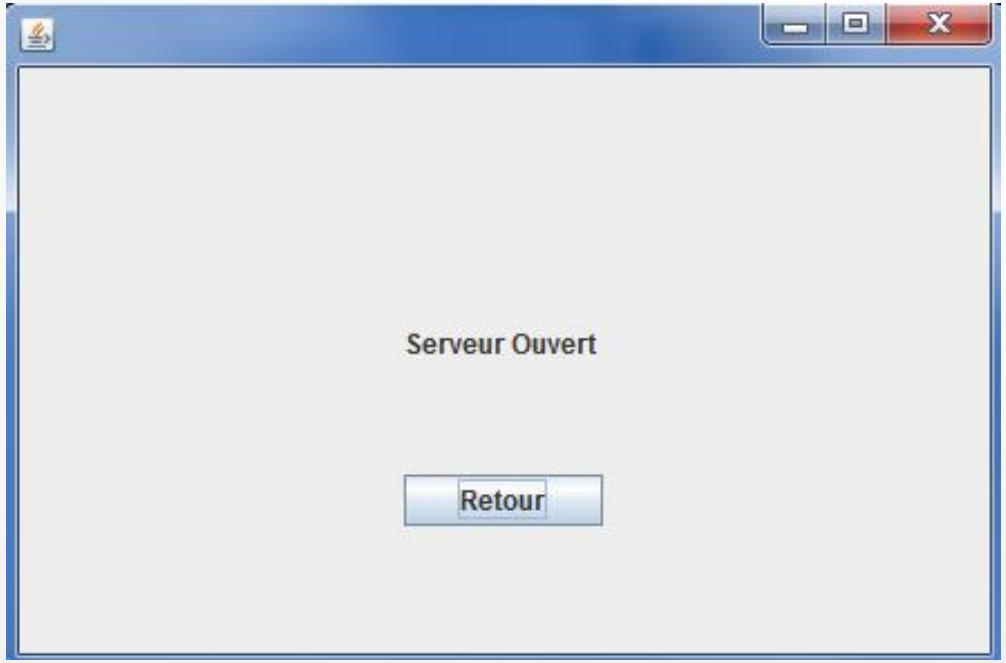
PRESENTATION DE L'IHM



PRESENTATION DE L'IHM



PRESENTATION DE L'IHM



ANNEXES

DataBase

```java
import java.sql.*;
import java.util.ArrayList;

// Notice, do not import com.mysql.
public class DataBase  {
    static final String db = "hercule";
    ArrayList listCommande;


    public DataBase(ArrayList listCommande){
        Connection con = null;
        Statement requete = null;
        ResultSet resultat = null;
        String req = "SELECT * FROM mouvement ; ";
        this.listCommande = listCommande ;



        try {
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost/" + db
```

```java
        } catch (Exception e) {
            System.out.println("Erreur connection !!! ");
            e.printStackTrace();
        }
        try {
            con.close();
            System.out.println("Base de données " + db + " fermee");
        } catch (Exception e) {
        }

        System.out.println("ListCommande :");
        for(int i=0;i<listCommande.size();i++) {
            System.out.println(""+listCommande.get(i));
        }
        //System.exit(0);
    }
}
```

Port COM

```
Enumeration enumComm;

enumComm = CommPortIdentifier.getPortIdentifiers();

while (enumComm.hasMoreElements()) {
    serialPortId = (CommPortIdentifier) enumComm.nextElement();
    if (serialPortId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
        System.out.println(serialPortId.getName());
    }
}

System.out.println("Program Finished Sucessfully");
try {
    serialPortId = CommPortIdentifier.getPortIdentifier("COM2");
    System.out.println(" portName : " + serialPortId.getName());
} catch (NoSuchPortException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

try {
    commPort = serialPortId.open(serialPortId.getName(), 2000);
} catch (PortInUseException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}

if (commPort instanceof SerialPort) {
    serialPort = (SerialPort) commPort;
    try {
        serialPort.setSerialPortParams(9600, SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
    } catch (UnsupportedCommOperationException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}

try {
    serialPort = (SerialPort) commPort;
    out = serialPort.getOutputStream();
    in = serialPort.getInputStream();
```

Mode apprentissage

```
String recPB = pb.getText();
String recVB = vb.getText();
String commandeB = "B" + recPB + ":" + recVB;
System.out.println("commandeB = " + commandeB);

String recPE = pe.getText();
String recVE = ve.getText();
String commandeE = "E" + recPE + ":" + recVE;
System.out.println("commandeE = " + commandeE);

String recPC = pc.getText();
String recVC = vc.getText();
String commandeC = "C" + recPC + ":" + recVC;
System.out.println("commandeC = " + commandeC);

String recPT = pt.getText();
String recVT = vt.getText();
String commandeT = "T" + recPT + ":" + recVT;
System.out.println("commandeT = " + commandeT);

String recPR = pr.getText();
String recVR = vr.getText();
String commandeR = "R" + recPR + ":" + recVR;
System.out.println("commandeR = " + commandeR);

String recPP = pp.getText();
String recVP = vp.getText();
String commandeP = "P" + recPP + ":" + recVP;
System.out.println("commandeP = " + commandeP);
```

Restitution des séquences de mouvements

```
data = new DataBase(listCommande);
for (int i = 0; i < listCommande.size(); i++) {
    comboBox.addItem(listCommande.get(i));

}
System.out.print("liste : " + listCommande);

JButton btnExecuter = new JButton("Executer");
btnExecuter.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {

        System.out.println("\n" + comboBox.getSelectedItem());
        String element;
        element = (String) comboBox.getSelectedItem();
        try {
            out.write(element.getBytes());
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        try {
            out.write(" \n".getBytes());
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        try {
            out.flush();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
});
```