Apéndice D. Recepción de Bytes en el RCX

Clase DirectControl.java:

```
import josx.platform.rcx.*;
import java.io.IOException;

public class DirectControl extends Thread implements ActionCode{

    private static int pOpCode1, pOpCode2;
    byte[] input;

    /**
     * Método que inicia el thread para que siempre esté en la espera
     * de mensajes infrarrojos.
     */
    public void run(){
      input= new byte[2];
      while(true) {
            if (Serial.isPacketAvailable()) { //Si se envió un paquete
                  Serial.readPacket(input);//Se recibe el paquete
                  parseInstructions(input);//Se traducen los bytes
                        //recibidos en instrucciones para el RCX.
            }
      }
    }

    /**
     * Método que convierte el arreglo de bytes recibido en una
     * instrucción que el RCX entienda.
     * @param input byte[] Arreglo de dos bytes que contiene el ID del
     * robot y el número de identificación de la acción.
     */
    public static void parseInstructions(byte[] input) {
      pOpCode1 = input[0] & 0xFF;   //ID
      pOpCode2 = input[1] & 0xFF;   //Acción
      if (pOpCode1 == ROBOT_ID) { //Comparación con el ID del robot
            RandomMovement.running = false; //Se detienen los demás
            RobotBehavior.autonomous = false;//comportamientos
            RobotBehavior.maxPower(); //Mayor fuerza para mover el balón
            if (pOpCode2 == ACTION_FORWARD) { //Búsqueda de la acción que
                  RobotBehavior.goForward(MOVE_TIME); //se quiere
                  RobotBehavior.stopMotors();
            } else if (pOpCode2 == ACTION_BACKWARD) {
                  RobotBehavior.goBackward(MOVE_TIME);
                  RobotBehavior.stopMotors();
            } else if (pOpCode2 == ACTION_TURN_LEFT) {
                  RobotBehavior.turnLeft(TURN_TIME);
                  RobotBehavior.stopMotors();
            } else if (pOpCode2 == ACTION_TURN_RIGHT) {
                  RobotBehavior.turnRight(TURN_TIME);
                  RobotBehavior.stopMotors();
            } else if (pOpCode2 == ACTION_ROTATE_LEFT) {
```

```
                    RobotBehavior.rotateLeft(ROTATE_TIME);
                    RobotBehavior.stopMotors();
                } else if (pOpCode2 == ACTION_ROTATE_RIGHT) {
                    RobotBehavior.rotateRight(ROTATE_TIME);
                    RobotBehavior.stopMotors();
                } else if (pOpCode2 == ACTION_SHOOT) {
                    RobotBehavior.shoot();
                } else if (pOpCode2 == ACTION_OPEN_CLAMPS) {
                    if(!RobotBehavior.clamps_open){
                        RobotBehavior.motorBOpen(CLAMPS_TIME);
                    }
                } else if (pOpCode2 == ACTION_CLOSE_CLAMPS) {
                    if(RobotBehavior.clamps_open){
                        RobotBehavior.motorBClose(CLAMPS_TIME);
                    }
                } else if (pOpCode2 == ACTION_AUTONOMOUS_MODE) {
                    RobotBehavior.normalPower();
                    RandomMovement.running = true;      //Regresa al
                    RobotBehavior.autonomous = true; //comportamiento
                    Pressure.hasCollided = true; //autónomo del robot
                } else if (pOpCode2 == ACTION_STOP) {
                    RobotBehavior.stopMotors();
                }
            }
        }
    }
```