APÉNDICES

APÉNDICE 1

CÓDIGO EN leJOS DE LAS PRINCIPALES CLASES IMPLEMENTADAS

1.1 Clase: robotPrueba1

```
import josx.robotics.*;
import josx.platform.rcx.*;

/**
 *  Clase principal del proyecto. Aquí se inicializan todos
los comportamientos incluidos.
 *@author     Laura Contreras Ortega
 *@created    25 de noviembre de 2003    */

public class robotPrueba1 {

     /**
      *  Variable que indica el nivel de hambre del robot */
     public static int Hambre = 0;

     /**
      *  Variable que indica la presencia de un predador en
el ambiente. False cuando no hay predador presente, true
cuando si lo hay. */
     public static boolean Miedo = false;

     /**
      *     Variable que indica el nivel de deseo de
apareamiento del robot */
     public static int Aparear = 0;

     /**
      *   La parte que inicia todo el funcionamiento del
robot.  Aquí se declaran todos los comportamientos que se van
a incluir y se declara el arreglo que los contendrá, la
posición en el arreglo indica la prioridad de los
comportamientos.
      *@param  args  Los argumentos siempre son vacíos. */

     public static void main(String[] args) {
          TextLCD.print("LAU");
          Sensor.S2.setTypeAndMode(3, 128);
          Sensor.S2.activate();

          caminar move = new caminar();
          move.definePotencia();

          Behavior b1 = new DriveForward();
```

```
            Behavior b2 = new golpeSensor1();
            Behavior b3 = new golpeSensor3();
            Behavior b4 = new cambiarDireccion();
            Behavior b5 = new incrementaVariables();
            Behavior b6 = new hayPredador();
            Behavior[] behArray = {b1, b2, b3, b5, b4, b6};
            Arbitrator arby = new Arbitrator(behArray);
            arby.start();
      }
}
```

1.2 Clase: golpeSensor1

```
import josx.robotics.*;
import josx.platform.rcx.*;

/**
 *   Esta clase implementa la reacción que el robot tendrá
cuando el sensor conectado al puerto de entrada 1 sea
liberado.
 *@author     Laura Contreras Ortega
 *@created    25 de noviembre de 2003   */

public class golpeSensor1 implements Behavior {
      caminar movimiento = new caminar();
      reaccionEstimulo reaccion = new reaccionEstimulo();
      int valorSensor = 0;
      int romperCiclo = 0;

      /**
       *  Este método define cuando este comportamiento entra
en acción. Esto depende del valor actual del sensor conectado
al puerto de entrada 1. (Presionado ó liberado)
       *@return     Un boolean que será true cuando el robot
hace contacto con algún objeto      */

      public boolean takeControl() {
            if (Sensor.S1.readBooleanValue() == false) {
                  return true;
            } else {
                  return false;
            }
      }

      /**
       *  Los motores se detienen cuando otro comportamiento
suprime al actual. */
      public void suppress() {
            movimiento.alto();
      }

      /**
       *    Cuando el sensor es liberado se controlan los
motores de manera que el robot pueda acercarse e identificar
```

el color del estímulo presente. Esto se hace por ciclos ya que no se sabe que tan cerca o lejos del estímulo hizo contacto la antena. Una vez que se cree que está colocado frente al estímulo toma una medición de luz, la cual se pasa al método reconoceEstímulo de la clase reaccionEstímulo. */

```
     public void action() {
          LCD.showNumber(222);
          movimiento.esperarPor(500);
          valorSensor = Sensor.S2.readValue();
          LCD.showNumber(valorSensor);
          movimiento.esperarPor(600);
          do {
               romperCiclo = 0;
               movimiento.atrasPor(500);
               movimiento.izquierdaPor(220);
               do {
                    romperCiclo++;
                    movimiento.adelante();
               } while (Sensor.S1.readBooleanValue() == true
&& romperCiclo < 190);
               movimiento.alto();
               valorSensor = Sensor.S2.readValue();
               LCD.showNumber(valorSensor);
               movimiento.esperarPor(1000);
          } while (Sensor.S1.readBooleanValue() == false &&
romperCiclo < 190);
          //Fin del for
          reaccion.reconoceEstimulo(valorSensor);
     }

}
```

1.3 Clase: incrementaVariables

```
import josx.robotics.*;
import josx.platform.rcx.*;
import josx.util.*;

/**
 *   En esta clase se define un comportamiento para que las
variables motivacionales del robot sean incrementadas
aproximadamente cada 5 segundos.
 *@author     Laura Contreras Ortega
 *@created    25 de noviembre de 2003
 */
public class incrementaVariables implements TimerListener,
Behavior {
     caminar movimiento = new caminar();
     boolean tiempoFuera;

     /**
```

```
      *  Constructor de la clase. Se inicializa la variable
tiempoFuera con valor false y se define el Timer para
controlar este comportamiento. Se inicia     */
     public incrementaVariables() {
          tiempoFuera = false;
          Timer tmr = new Timer(5000, this);
          tmr.start();
     }

     /**
      *   En este método se indica que se ha superado el
tiempo definido en el constructor. */
     public void timedOut() {
          tiempoFuera = true;
     }

     /**
      *  Este método indica cuando es que este comportamiento
entra en ejecución. Esto lo decide en base al valor de
tiempoFuera.
      *@return      Devuelve un boolean correspondiente al
valor de la variable tiempoFuera.   */
     public boolean takeControl() {
          return tiempoFuera;
     }

     /**
      *    Se detienen los motores cuando este método es
suprimido por otro de mayor prioridad    */
     public void suppress() {
          movimiento.alto();
     }

     /**
      *    En este método se incrementan las variables
motivacionales del robot. La variable hambre se incrementa al
doble que la variable Aparear.      */
     public void action() {
          TextLCD.print("VAR");
          robotPrueba1.Hambre = robotPrueba1.Hambre + 2;
          robotPrueba1.Aparear = robotPrueba1.Aparear + 1;
          LCD.showNumber(robotPrueba1.Hambre);
          movimiento.esperarPor(1000);
          tiempoFuera = false;
     }
}
```

1.4 Clase: reaccionaEstimulo

```
import josx.robotics.*;
import josx.platform.rcx.*;

/**
```

```
 *  En esta clase se maneja la reacción del robot en base al
estímulo presente en el ambiente. Se identifica el color del
estímulo,  se  considera  el  valor  de  las  variables
motivacionales y se decide qué hacer.
 *@author     Laura Contreras Ortega
 *@created    25 de noviembre de 2003    */

public class reaccionEstimulo {
     caminar movimiento = new caminar();

     /**
      *   Dentro de este método se distingue el estímulo
presente en el ambiente en base a su color. Si es menor que
30 es una lata negra    (representa comida).Este estímulo se
toma cuando la variable de hambre tiene un valor mayor que 20
y si su valor también es mayor que el de la variable Aparear.
Para identificar cuando el estímulo ha sido tomado el robot
hace un ruido (diferente a un simple beep). Si es mayor que
40 es una lata blanca   (representa predador). La variable
booleana Miedo cambia su valor a true e inmediatamente este
comportamiento es suprimido y toma el control otra conducta.
El robot emite un ruido que es una secuencia de beeps. Si es
menor que 40 pero mayor que 30 es una lata verde (representa
pareja). Cuando se topa con una pareja el robot toma el
estímulo si el valor de la variable Aparear es mayor que el
de la variable Hambre y también si el valor de ésta es menor
que 70. El que sea mayor indica que el robot está muriendo de
hambre y debe de buscar alimento.
      *@param  valorSensor  Este método recibe como parámetro
el valor registrado por el sensor de luz al acercarse al
estímulo. Se usará para definir el color del estímulo. */
     public void reconoceEstimulo(int valorSensor) {
          if (valorSensor < 30) {
               //Lata negra ==> Comida
               if    ((robotPrueba1.Hambre    >    19)    &&
(robotPrueba1.Hambre > robotPrueba1.Aparear)) {
                    TextLCD.print("COM");
                    robotPrueba1.Hambre = 0;
                    Sound.buzz();
               } else {
                    if        ((robotPrueba1.Aparear        >
robotPrueba1.Hambre) || (robotPrueba1.Hambre < 19)) {
                         Sound.beep();
                    }
               }
               movimiento.alejarObjeto();
          } else {
               if (valorSensor < 40) {
                    //Lata Verde ==> Pareja
                    if        ((robotPrueba1.Aparear        >
robotPrueba1.Hambre) && (robotPrueba1.Hambre < 70)) {
                         TextLCD.print("PAR");
                         robotPrueba1.Aparear = 0;
                         Sound.twoBeeps();
```

```
                    } else {
                         Sound.beep();
                    }
                    movimiento.alejarObjeto();
               } else {
                    if (valorSensor >= 40) {
                         //Lata Blanca ==> Predador
                         Sound.beepSequence();
                         robotPrueba1.Miedo = true;
                    }
               }
          }
     } // Fin del método reconoceEstimulo(int valorSensor)
}
```

1.5 Clase: caminar

```
import josx.platform.rcx.*;

/**
 *  Esta clase contiene diferentes métodos para controlar los
motores y observar varios movimientos en el robot.
 *@author     Laura Contreras Ortega
 *@created    25 de noviembre de 2003    */
public class caminar {
     int valor = 0;
     int num = 0;
     int opc = 0;

     /**
      *   Método para calibrar los motores y tratar que el
robot camine realmente en línea recta.  */
     public void definePotencia() {
          Motor.A.setPower(7);
          Motor.C.setPower(5);
     }

     /**
      *      Los   motores   se   mueven   hacia   el   frente
indefinidamente.     */
     public void adelante() {
          Motor.A.forward();
          Motor.C.forward();
     }

     /**
      *   Este método hace que los motores se muevan hacia
atrás por un cierto tiempo.
      *@param  tiempo  Variable que indica el tiempo por el
cual avanza el robot. 1000 ~ 1seg.      */
     public void atrasPor(long tiempo) {
          Motor.A.backward();
          Motor.C.backward();
          try {
```

```
            Thread.sleep(tiempo);
        } catch (InterruptedException ie) {
            LCD.showNumber(1);
        }
    }

    /**
     *  Este método controla los motores y hace que el robot
gire a la derecha por un cierto tiempo.
     *@param   tiempo   El tiempo por el cual girará a la
derecha el robot. 1000 ~ 1seg.     */
    public void derechaPor(int tiempo) {
        Motor.A.forward();
        Motor.C.backward();
        try {
            Thread.sleep(tiempo);
        } catch (InterruptedException ie) {
            LCD.showNumber(1);
        }
    }

    /**
     *  Este método controla los motores y hace que el robot
gire a la izquierda por un cierto tiempo.
     *@param   tiempo   El tiempo por el cual girará a la
derecha el robot. 1000 ~ 1seg.     */
    public void izquierdaPor(int tiempo) {
        Motor.A.backward();
        Motor.C.forward();
        try {
            Thread.sleep(tiempo);
        } catch (InterruptedException ie) {
            LCD.showNumber(1);
        }
    }

    /**
     *  Este método detiene el movimiento de los motores. */
    public void alto() {
        Motor.A.stop();
        Motor.C.stop();
    }

    /**
     *  Este método duerme el hilo de la ejecución el tiempo
que sea necesario.
     *@param  tiempo  Tiempo por el cual se desea dormir la
ejecución del hilo. 1000 ~ 1 seg.   */
    public void esperarPor(long tiempo) {
        try {
            Thread.sleep(tiempo);
        } catch (InterruptedException ie) {
            LCD.showNumber(1);
        }
```

```
    }

    /**
     *  Este método hace girar los motores hacia la derecha
o izquierda, según el número random generado.     */
    public void girar() {
        num = (int) (Math.random() * 10);
        opc = num % 2;
        if (opc == 0) {
            this.derechaPor(num * 300);
            //lado derecho
        } else {
            this.izquierdaPor(num * 300);
            //lado izquierdo
        }
    }

    /**
     *   Este método hace que el robot se detenga por un
momento, se haga hacia atrás y luego gire hacia algún lado.*/
    public void alejarObjeto() {
        this.esperarPor(1000);
        this.atrasPor(1500);
        this.girar();
    }

    /**
     *   Este método controla los motores de manera que el
robot camine hacia el frente simulando un zig zag.     */
    public void zigZagFrente() {
        Motor.A.forward();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException ie) {
            LCD.showNumber(1);
        }
        Motor.A.stop();
        Motor.C.forward();
        try {
            Thread.sleep(1500);
        } catch (InterruptedException ie) {
            LCD.showNumber(1);
        }
        Motor.C.stop();
    }

    /**
     *   Este método controla los motores de manera que el
robot camine hacia atrás simulando un zig zag.    */
    public void zigZagAtras() {
        Motor.A.backward();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException ie) {
```

```
            LCD.showNumber(1);
        }
        Motor.A.stop();
        Motor.C.backward();
        try {
            Thread.sleep(1500);
        } catch (InterruptedException ie) {
            LCD.showNumber(1);
        }
        Motor.C.stop();
    }

    /**
     *  Este método controla no solo los motores sino que
involucra también al sensor de luz. El robot gira
aproximadamente 360° y se va deteniendo cada cierto tiempo
para tomar la intensidad de luz en ese punto, esos valores se
guardan en un arreglo. Se recorre el arreglo para buscar el
punto más brillante pero menor que 40 (sería un posible
predador) y el robot gira de regreso al punto más brillante.
*/
    public void buscaLuz() {
        int step = 0;
        int valor = 0;
        int mayor = 0;
        int[] valores = new int[7];
        Sensor.S2.setTypeAndMode(3, 128);
        Sensor.S2.activate();
        do {
            Motor.A.forward();
            Motor.C.backward();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ie) {
                LCD.showNumber(1);
            }
            Motor.A.stop();
            Motor.C.stop();
            valor = Sensor.S2.readValue();
            valores[step] = valor;
            LCD.showNumber(valores[step]);
            step++;
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ie) {
                LCD.showNumber(1);
            }
        } while (step < 7);

        for (int k = 0; k < 7; k++) {
            valor = valores[k];
            if ((valor > mayor) && (valor < 40)) {
                mayor = valor;
                step = k;
```

```
                }
                LCD.showNumber(mayor);
            }
            step = 7 - step;
            do {

                Motor.A.backward();
                Motor.C.forward();
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException ie) {
                    LCD.showNumber(1);
                }
                //Motor.A.stop();
                //Motor.C.stop();
                step--;
            } while (step > 0);
        }

        /**
         *   Este método controla no solo los motores sino que
involucra también al sensor de luz. El robot gira
aproximadamente 360º y se va deteniendo cada cierto tiempo
para tomar la intensidad de luz en ese punto, esos valores se
guardan en un arreglo. Se recorre el arreglo para buscar el
punto más obscuro de todos y después girar de regreso a ese
punto.    */
        public void buscaSombra() {
            int step = 0;
            int valor = 0;
            int menor = 100;
            int[] valores = new int[7];
            Sensor.S2.setTypeAndMode(3, 128);
            Sensor.S2.activate();
            do {
                Motor.A.forward();
                Motor.C.backward();
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException ie) {
                    LCD.showNumber(1);
                }
                Motor.A.stop();
                Motor.C.stop();
                valor = Sensor.S2.readValue();
                valores[step] = valor;
                LCD.showNumber(valores[step]);
                step++;
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException ie) {
                    LCD.showNumber(1);
                }
            } while (step < 7);
```

```
            for (int k = 0; k < 7; k++) {
                 valor = valores[k];
                 if (valor < menor) {
                      menor = valor;
                      step = k;
                 }
                 LCD.showNumber(menor);
            }
            step = 7 - step;
            do {
                 Motor.A.backward();
                 Motor.C.forward();
                 try {
                      Thread.sleep(1000);
                 } catch (InterruptedException ie) {
                      LCD.showNumber(1);
                 }
                 //Motor.A.stop();
                 //Motor.C.stop();
                 step--;
            } while (step > 0);
       }
}
```