3 D-Senso

Este capítulo describe el diseño e implementación de D-SENSO un sistema de ayuda al diseño de redes de sensores. El capítulo está organizado de la siguiente manera. La sección 3.1 describe los componentes principales del sistema: clase `Sensor` que abstrae al sensor y sus características técnicas; la clase `CrearEspacio`, que permite especificar un espacio con respecto a su largo, ancho y altura. La sección 3.3 describe las funciones del sistema: creación de espacio, creación de rebanadas, cálculo de sensores adyacentes, cálculo de *clusters* para describir una red lógica de acuerdo al alcance de transmisión de los sensores que la constituyen. Finalmente, la sección 3.4 concluye el capítulo.

3.1 Diseño de una red de sensores

Una red de sensores, desde el punto de vista D-SENSO, de está compuesta por un número *n* de sensores que se colocan en un terreno en posiciones específicas. Los sensores se organizan lógicamente como un conjunto de clusters, donde un cluster agrupa un conjunto de sensores adyacentes, es decir que pueden comunicarse tomando en cuenta su alcance de transmisión.

El espacio representado por D-SENSO, es un espacio tridimensional. Tomamos en cuenta que un espacio puede variar en su altura, es por eso que dividimos rectangularmente dicho espacio lo cual le llamamos rebanadas, es decir que un conjunto de rebanadas forman un

terreno. Cada rebanada se la asignará una altura específica dentro de un rango, esta altura se representara con diferentes colores en escala de grises.

Una vez definido el espacio, se colocan el usuario coloca los sensores en algún punto del espacio. En D-SENSO cada sensor tiene ciertas características únicas como son un identificador, batería, capacidad, memoria y alcance, estos valores son predefinidos pero permite definir nuevos tipos de sensores.

Para poder posicionar los sensores D-SENSO crea una malla, que divide cada rebanada del espacio en celdas, esta división se calcula dividiendo el número de sensores entre el número de rebanadas que constituyen el terreno. D-SENSO presenta el espacio y las celdas de manera visual, los sensores son posicionados por el usuario las celdas, con la restricción de que en una celda no se puede posicionar más de un sensor. Una vez los sensores posicionados D-SENSO calcula para cada sensor, los sensores adyacentes para determinar qué grupo de sensores están fuertemente conectados entre sí.

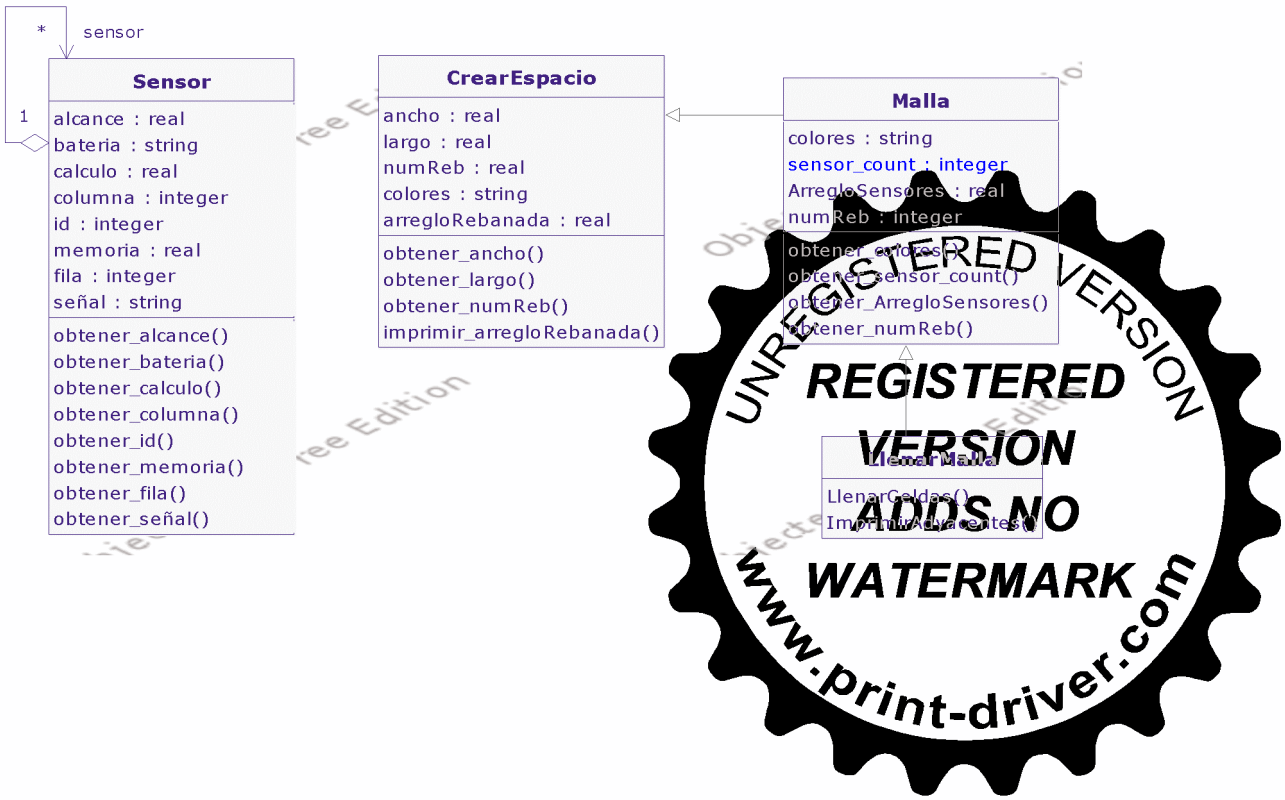

Figura 3.1 Diagrama de clases UML de las estructuras de datos de D-SENSO

La Figura 3.1 presenta el diagrama UML de las clases principales de D-SENSO : `Sensor`, `Espacio` y `Malla` asociadas a las clases que implementan las operaciones que se aplican sobre ellas (`CrearEspacio`, `LlenarMalla`).

3.1.1 Clase `Sensor`

La clase **`Sensor`** (Figura 3.2) representa un sensor en términos de sus características de los tipos de datos que maneja como la capacidad de memoria, el estado de la batería del sensor en la red, su capacidad de cálculo. Es decir, mide la capacidad de su procesador para realizar las operaciones, para definir su alcance de transmisión en la red, cada sensor tendrá un número único de registro y también pertenece a una fila y una columna dentro del espacio, específicamente en la malla.

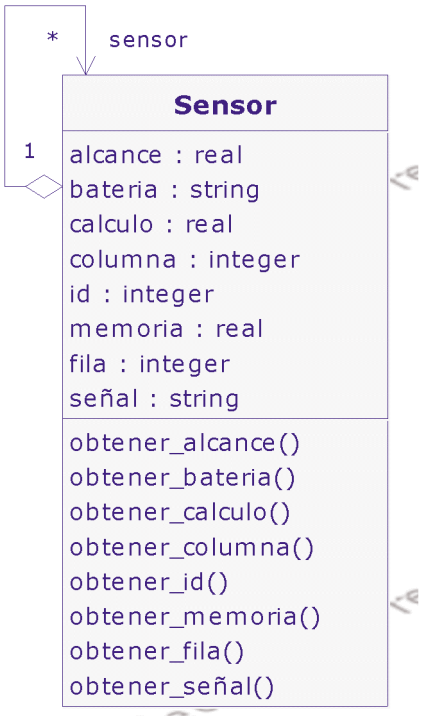

Figura 3.2 Clase `Sensor`

3.1.2 Clase `CrearEspacio`

La clase **`CrearEspacio`** representa el terreno que se va a representar para posicionar los sensores, maneja los datos de ancho y largo del espacio, así como un arreglo que contiene las rebanadas que forman dicho espacio, cada rebanada se le asigna un color para representar su altura.

De la clase **`CrearEspacio`** (Figura 3.3) se hereda la clase **`Malla`**, la cual nos sirve para posicionar los sensores dentro del espacio, cada uno en una celda. Para calcular las celdas que conforman la malla, se divide el número de sensores entre el número de rebanadas, el número resultante nos da el total de celdas que se necesitan para posicionar los sensores y para la creación de la malla. En caso de que el número resultante no sea un valor entero, se

redondea al próximo entero, para asegurarnos que por lo menos siempre hay una celda para cada sensor.

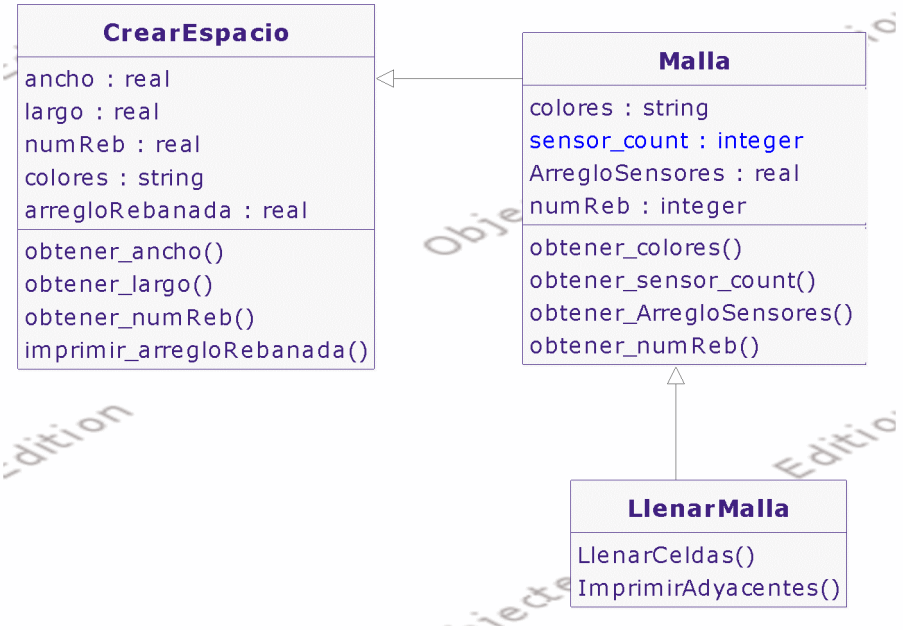

Figura 3.3 Clase `CrearEspacio`

3.1.3 Clase `TerrenoControl`

La clase **`TerrenoControl`**, es la clase principal de D-SENSO (Figura 3.4). Esta clase realiza la mayoría de las funciones, y guarda información para el sistema: el número de rebanadas se guardan en una lista, una lista es un tipo de arreglo o array. Al crear la variable `List<>` se inicializa su capacidad, que se aumenta conforme la lista va creciendo, es decir conforme al número de rebanadas. Los colores que representan la altura de las rebanadas son guardados en una matriz llamada `colors`.

```
public class TerrenoControl : Control
      {
            int slices = 1; // As cols
            int sensors_count = 1;
            int rows; // The number of rows
            int slice_width;
            int slice_height;
            List<SliceInfo> slices_areas;
            Color [] colors;
            ToolTip tooltip;

            int area_width;
            int area_height;
```

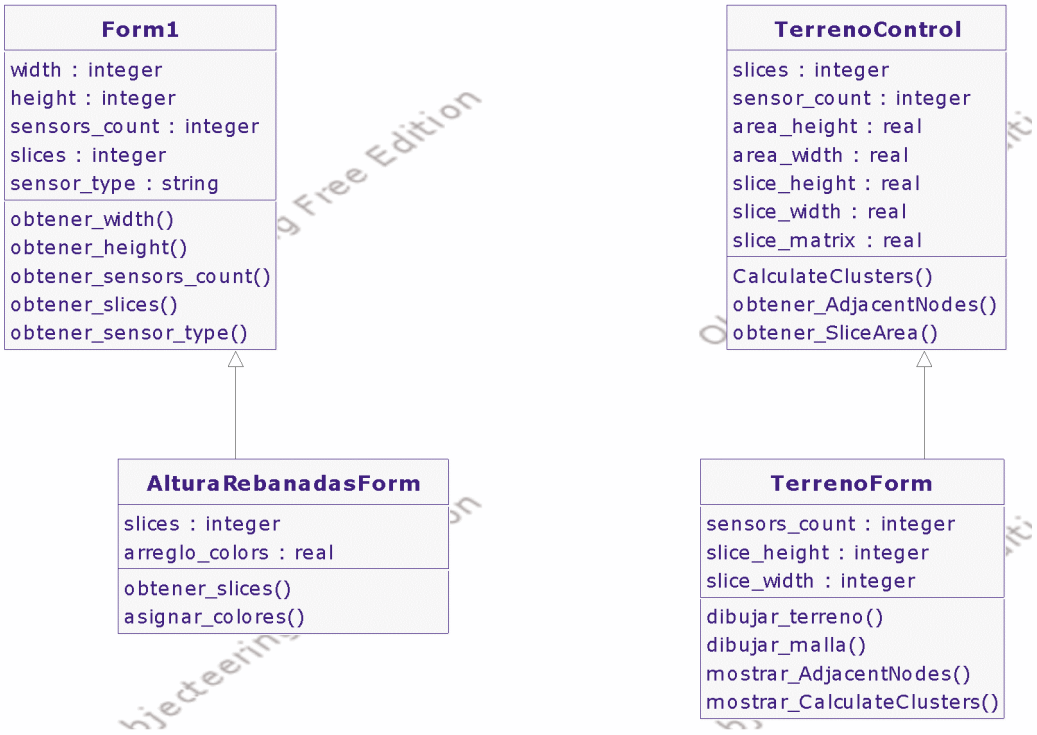

Figura 3.4 Diagrama de clases para la configuración del terreno

Una vez que tenemos definido el espacio, creamos la malla, que es una matriz bidimensional la cual representa las filas y las rebanadas, que son los espacios creados para posicionar los sensores.


```
void Calculate ()
```

```
            {
                  if (slices_areas == null)
                        slices_areas = new List<SliceInfo> ();

                  slices_areas.Clear ();

                  rows = (int)Math.Ceiling ((double)sensors_count /
(double)slices);
                  if (rows < 1)
                        rows = 1;

                  slice_matrix = new SliceInfo [slices, rows];

                  slice_width = Width / slices;
                  slice_height = Height / rows;
```

Los sensores son componentes con valores predeterminados, que se inicializan en la clase **Form1**, un componente en c#, se utiliza normalmente para referirse a un objeto reutilizable que expone una o varias interfaces a los usuarios de forma normalizada

```
public Form1 ()
            {
                  InitializeComponent ();
                  DefaultSensorInfo = new Sensor (0);
                  DefaultSensorInfo.Bateria = "100%";
                  DefaultSensorInfo.Calculo = 15;
                  DefaultSensorInfo.Memoria = 5;
                  DefaultSensorInfo.Alcance = 10;
                  DefaultSensorInfo.Senal = "Luz";
            }
```

Ya con los sensores posicionados, se calculan los *clusters* de sensores, los cuales son pintamos de un mismo color, dentro de un rango de RGB, esto es para que se puedan identificar fácilmente a simple vista.

```
Random rand = new Random ();
                  foreach (List<int> cluster in clusters) {
                        int r = rand.Next (255);
                        int g = rand.Next (255);
                        int b = rand.Next (255);
                        Color c = Color.FromArgb (r, g, b);

                        foreach (int index in cluster) {
```

```
                                // Get the label, and set the color!
                                SensorLabel label = (SensorLabel)
slices_areas [index].sensor;
                                if (label != null)
                                      label.BackColor = c;
```

3.1.4 Clase `Form1`

La clase **`Form1`** permite inicializar un terreno, a través de una forma, vamos obteniendo los datos de altura, y largo, así como el número de sensores que deseamos posicionar y por último las rebanadas en las que queremos dividir nuestro terreno. En la clase **`AlturaRebanasForm`** asignamos a cada rebanada una altura, la cual es representada por un color en escala de grises.

```
private void button1_Click (object sender, EventArgs e)
            {
                  // Step 1: colors for slices!
                  AlturaRebanadasForm color_form = new
AlturaRebanadasForm ((int)numericUpDown4.Value);
                  color_form.ShowDialog ();

                  TerrenoForm form = new TerrenoForm
((int)numericUpDown3.Value);
                  form.SetAllSensorAtributes (DefaultSensorInfo);
                  form.SensorTextLabel.Text = comboBox1.SelectedIndex ==
0 ? " Sensores Luz" : "Sensores Temperatura";
                  TerrenoControl terreno = form.Terreno;

                  terreno.Width = (int)numericUpDown1.Value * 5;
                  terreno.Height = (int)numericUpDown2.Value * 5;
                  terreno.AreaWidth = (int)numericUpDown1.Value;
                  terreno.AreaHeight = (int)numericUpDown2.Value;
                  terreno.Slices = (int)numericUpDown4.Value;
                  terreno.SetColors (color_form.GetSlicesColors ());

                  form.Show();
            }
```

3.1.5 Clases TerrenoControl y TerrenoForm

La clase **TerrenoControl** es la clase que realiza todas las operaciones que se van a efectuar en el terreno, como dibujar el terreno y hacer las operaciones del cálculo de clusters.

```
public class TerrenoControl : Control
	{
		int slices = 1; // As cols
		int sensors_count = 1;
		int rows; // The number of rows
		int slice_width;
		int slice_height;
		List<SliceInfo> slices_areas;
		Color [] colors;
		ToolTip tooltip;

		int area_width;
		int area_height;

		int last_clicked_slice;
	}
```

La clase **TerrenoForm** realiza el cálculo para representar la malla para poder posicionar los sensores dentro del espacio, por medio del evento drag & drop, además implementa ToolTip para ver las características de cada sensor cuando se posicione el mouse sobre él.

```
public TerrenoControl ()
		{
			tooltip = new ToolTip ();
			AllowDrop = true;
			Calculate ();
		}
```

3.2 Funciones

Esta sección describe las funciones principales DE D-SENSO : creación de espacio, calcular sensores adyacentes, identificación de clusters.

3.2.1 Creación de espacio

La Figura 3.5 presenta el diagrama de secuencias de la función de creación de espacio. Creamos el espacio donde se instalará la red de sensores. Se representa un espacio real con sus dimensiones, según el ancho y largo. Una vez que se tienen estos valores, se indica el número de sensores que van a componer la red y el número de rebanadas en las que se a divide el terreno.

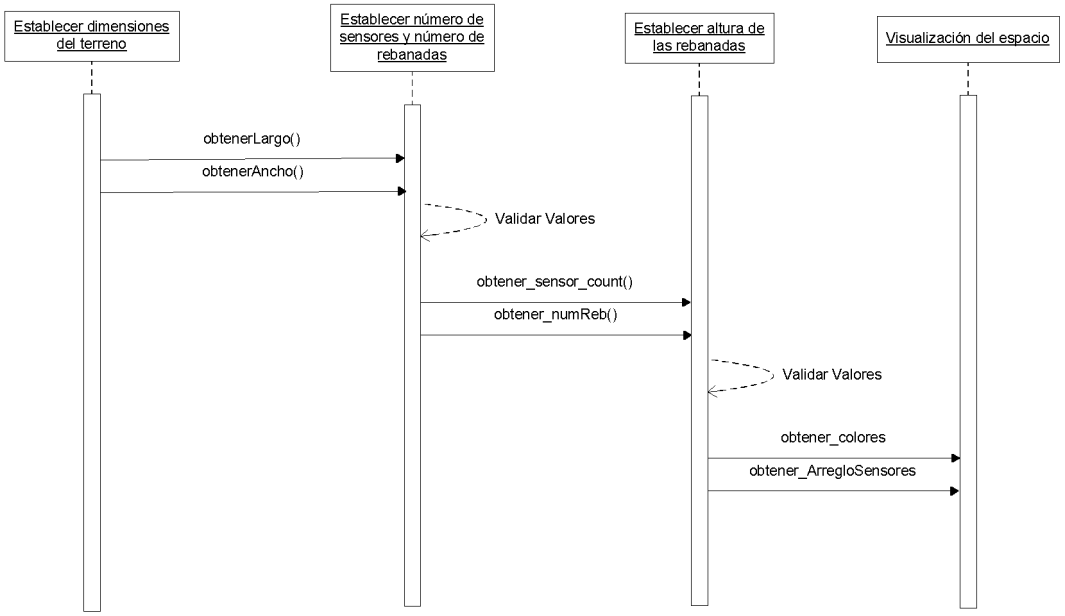

Figura 3.5 Diagrama de secuencia para la creación del espacio

Dado que suponemos que el terreno no es plano y no tiene el mismo a[illegible] todas sus [illegible]. Para cada rebanada en la que se desea dividir el espacio se [illegible] una altura y D-SENSO le asocia un color en escala de grises: blanco corresponde a [illegible] altura mínima y negro la altura máxima. De esta forma se genera el espacio que se visualiza en la interfaz gráfica del D-SENSO. El siguiente pseudocódigo implementa este pro[illegible]

```
Class CrearEspacio

{
   public string colores [,];
        float ancho, largo, numReb, arregloRebanada [,];

   public CrearEspacio ()
   {
      colores [,];
      largo = 0;
      ancho = 0;
      numReb = 0;
      arregloRebanada[,]
   }

   public CrearEspacio (string colores [,], float largo, float ancho,
float numReb, float arregloRebanada[,])
   {

  colores[,] = ({#CCCCCC}, {#999999}, {#666666}, {#333333});
  largo = largo;
  ancho = ancho ;
  numReb = numReb;
  arregloRebanada [,] = ({R1}, {R2}, … {Rn});


   }

}
```

Los valores de entrada para la creación del espacio son el largo, ancho, el número de sensores y el número de rebanadas en las que se va a dividir nuestro espacio. Esta operación se realiza de manera interactiva a través de la interfaz gráfica de D-SENSO.

3.2.2 Creación de rebanadas

La clase **`AlturaRebanadasForm`** inicializa el componente que va a representar las alturas de las rebanadas que conforman el terreno, como ya se había mencionado la altura será representada por colores en escala de grises. Estos colores se guardan en una matriz

llamada *colors*. El siguiente código ilustra la creación de las rebanadas de un terreno donde se asocia un color con una altura.

```
public partial class AlturaRebanadasForm : Form
      {
            Color [] colors;
            readonly Color DefaultGray = Color.FromArgb (255, 255, 255);
// White actually
            NumericUpDown [] numeric_updowns;

            public AlturaRebanadasForm (int slicesnum)
            {
                  InitializeComponent ();

                  colors = new Color [slicesnum];
                  numeric_updowns = new NumericUpDown [slicesnum];

                  int y = 10;
                  for (int i = 0; i < slicesnum; i++) {
                        Label label = new Label ();
                        label.Text = "Slice " + (i + 1);
                        label.Location = new Point (5, y);

                        NumericUpDown num_updown = numeric_updowns [i] =
new NumericUpDown ();
                        num_updown.Maximum = 10;
                        num_updown.Value = 0;
                        num_updown.Location = new Point (label.Right + 10,
label.Top);

                        Controls.Add (label);
                        Controls.Add (num_updown);
                        y += 30;
                  }
            }
```

3.2.3 Cálculo de sensores adyacentes

La Figura 3.6 presenta el diagrama de secuencias de la función que permite calcular los sensores adyacentes para cada sensor de la red. Cada sensor que está en el espacio, se localiza en una celda, obtenemos la ubicación del sensor, es decir en qué columna y en que

fila se encuentra, con esto calculamos los sensores que están adyacentes a su posición. Posteriormente se muestran dichos sensores.

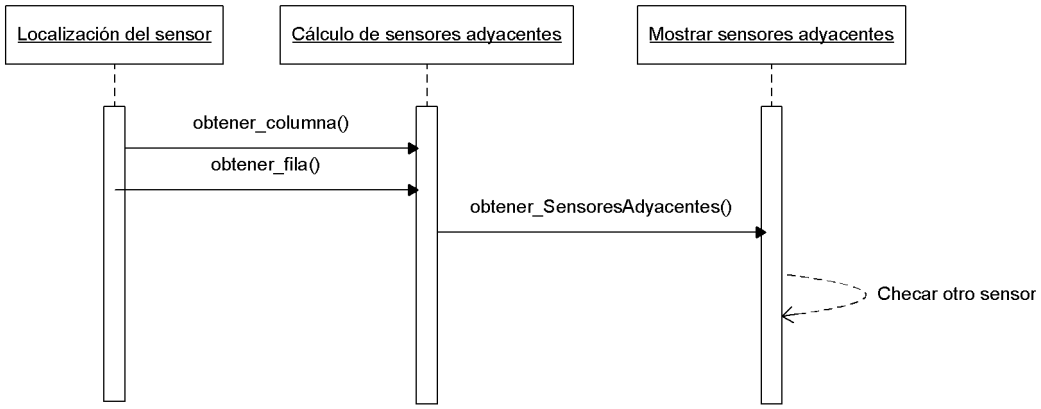

Figura 3.6 Diagrama de secuencia para calcul nodos adyacentes

El siguiente pesudocódigo muestra la implementación de este cálculo implementado por la clase `Sensor`.

```
public void ImprimirAdyacentes (Sensor [,] sensores, int n, int m, int i,
int j)
   {
bool up, down, left, right;

                  down = i > 0;
                  up = i + 1 < n;
                  left = j > 0;
                  right = j + 1 < m;

                  if (down) {
                        adj = sensores [i - 1, j];

                  }

                  if (up) {
                        adj = sensores [i + 1, j];

                  }

                  if (left) {
                        adj = sensores [i, j - 1];

                  }

                  if (right) {
                        adj = sensores [i, j + 1];

                  }
}

private void mostrarNodosAdyacentesToolStripMenuItem_Click (object
sender, EventArgs e)
            {
                  Point point = terrenoControl1.PointToClient
(Control.MousePosition);
                  int slicenum = terrenoControl1.GetSliceAt (point);
                  SliceInfo s = terrenoControl1 [slicenum];
                  if (s.sensor == null) {
                        MessageBox.Show ("No hay sensores en esta
posición");
                        return;
                  }

                  ICollection<object> sensors =
terrenoControl1.GetAdjacentNodes (slicenum);
                  StringBuilder sb = new StringBuilder ("Nodos
Adjacentes:\n");
                  foreach (object o in sensors) {
                        // hackish
                        Control c = (Control)o;
```

```
                sb.AppendLine (c.Text);
            }
```

3.2.4 Identificación de *Clusters*

La Figura 3.7 presenta el diagrama de secuencias de la función que permite identificar clusters que agrupan sensores según su capacidad de transmisión y su posición en el terreno. Al colocar los sensores en el espacio se crea una red de sensores, donde se pueden identificar los nodos clave de dicha red, sus sensores adyacentes y así definir *clusters* formados por un grupo de sensores que están fuertemente conectados. Para mostrar los *clusters* en la red se colorean de un mismo color, de este modo los podemos identificar más fácilmente dentro del espacio.

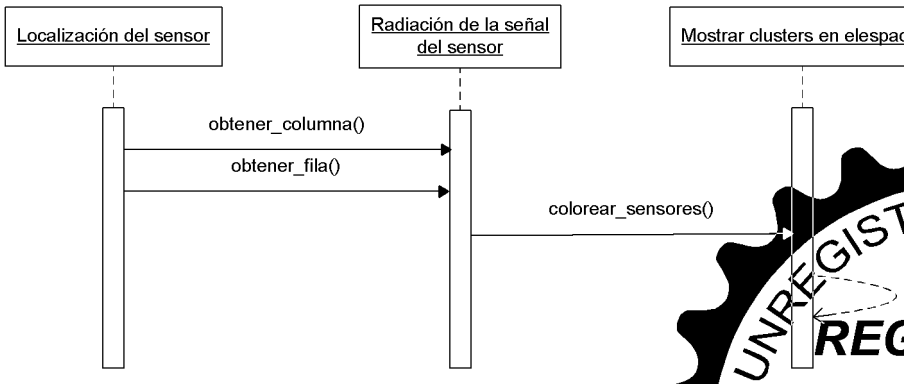

Figura 3.7 Diagrama de secuencia para identificar los nodos claves

Identificamos los *clusters* en la red, especificando el alcance de radiación de cada sensor, para después colorear el grupo de sensores identificado para su mejor visualización.

```
internal Rectangle GetSliceRegion (int index)
            {
                  Size sub_area_size = SubAreaSize;

                  SliceInfo slice = slices_areas [index];
            SensorLabel sensor_label = (SensorLabel)slice.sensor;

            Point origin;
            if (sensor_label == null)
                origin = new Point(sub_area_size.Width * slice.col,
sub_area_size.Height * slice.row);
            else
            {
                origin =
PointToClient(sensor_label.PointToScreen(Point.Empty));
                origin.X -= 60;
                origin.Y -= 60;
            }

            Rectangle rect = new Rectangle(origin, new Size(120, 120));
            return rect;
```

3.3 Conclusiones

Este capítulo presentó el diseño e implementación de D-SENSO que se basa en estructuras de datos que representan las nociones de sensor, red de sensores y terreno para implementar las funciones básicas para diseñar una red y ofrecer una visión lógica de su topología según la capacidad de transmisión de los sensores. El capítulo usa diagramas UML de clases y de secuencias para describir los componentes principales del sistema. Para las funciones principales, le capítulo da un trozos del código que las implementa.


UNREGISTERED VERSION
REGISTERED
VERSION
ADDS NO
WATERMARK
www.print-driver.com