SEP **SNEST** **DGEST**

INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

CONTROL DIFUSO PARA EL CLIMA DE UN INVERNADERO

INFORME DE RESIDENCIA PROFESIONAL

QUE PRESENTA:

FERNANDA VIRIDIANA SILVA NIGENDA
07270075

NOMBRE DEL ASESOR INTERNO: DOCTOR HÉCTOR RICARDO HERNÁNDEZ DE LEÓN

INGENIERÍA ELECTRÓNICA

TUXTLA GUTIÉRREZ, CHIAPAS. MÉXICO. JUNIO DEL 2011

CONTENIDO

RESUMEN
En este documento se presenta una introducción a la teoría y aplicación de lógica difusa en sistemas de control; así como la implementación de dispositivos móviles para mayor información al usuario. En particular, se introduce un leve cambio a la configuración tradicional de un lazo de control difuso, que permite realizar un ajuste fino o medida correctivo a un controlador ya diseñado, en forma intuitiva y relacionada con el control clásico. Para confirmar el análisis teórico se presenta una aplicación, la cual fue el motivo de la residencia; esta aplicación fue llevada a cabo en un pequeño invernadero del ITTG, en esta aplicación se desarrolla en forma detallada el diseño de un lazo de control que permita regular el clima de un invernadero, mediante el control de tres variables, humedad relativa, temperatura y luminosidad.

PALABRAS CLAVES: Lógica difusa, control automático, humedad relativa, temperatura, luminosidad, dispositivos móviles.

INTRODUCCIÒN.

La lógica difusa puede ser descrita como un sistema, interpretativo, en el cual los objetos o elementos son relacionados con conjuntos de fronteras no nítidamente definidas, otorgándoles un grado, de pertenencia relativa o graduada y no estricta como es de costumbre en la lógica tradicional. Se puede afirmar que existe una interpolación entre una frontera y otra, es decir, entre un conjunto y otro.

Esto permite sentencias de lenguaje común, se caracterizan por ser un tanto indefinidas, para interpretar el estado de las variables de cierto proceso, asignándoles en cada momento un grado de pertenencia a estos conjuntos difusos. Esto puede ser relacionado mediante operadores lógicos tradicionales con cierta medida de acción, también de naturaleza no exacta, que son diseñadas de tal manera que produzcan un cambio deseado en las variables .Esto quiere decir que se puede diseñar un controlador, que interprete en forma intuitiva el estado de ciertas variables, y en base a ello deduzca en forma lógica una actuación posible que permita llevar la variable al estado deseado.

En la práctica, esto acarrea algunos problemas, por ejemplo se puede atravesar por todo el proceso del diseño del controlado y obtener resultados pobres, obligando a revisar el diseño en su totalidad para contrarrestar esta desventaja.

Para el control del clima del invernadero del ITTG se diseñó una interfaz para mostrar de manera gráfica en la computadora las medidas realizadas de las tres variables, además se utilizaron dispositivos móviles para una mejor comunicación entre el usuario y la interfaz. El dispositivo móvil trasmisor manda un mensaje a otro dispositivo avisando que la lectura de las mediciones ha finalizado, también muestra los valores de las últimas mediciones realizadas, con el fin de proporcionar al usuario un panorama de

la situación climática del invernadero. En este invernadero se trabajó con el agave y frijol, las dos plantas son importantes para la región, ya que constituyen una fuente más de alimentación.

JUSTIFICACIÓN DE CONTROL DE CLIMA PARA UN INVERNADERO.

Normalmente en la lógica convencional tenemos un conjunto de enunciados que pueden ser verdaderos o falsos; es decir 0 y 1; pero gracias a que la lógica difusa emplea expresiones que no son totalmente ciertas ni totalmente falsas se puede aplicar la lógica tomando un valor indeterminado de veracidad dentro de un conjunto de valores cuyos extremos son la verdad absoluta o la falsedad absoluta. La lógica difusa maneja esta incertidumbre en los sistemas de control mediante grados de certeza para responder a una cuestión lógica; esto es importante porque hace que el control de clima para un invernadero se vuelva práctico y comercial; ya que la simplicidad del diseño lo vuelve de bajo costo y a la vez fácil de utilizar para los usuarios.

OBJETIVO.

Lograr un control automático de temperatura, luminosidad y humedad relativa en un invernadero, basándose en lógica difusa y con costos de producción menores que los controles comerciales.

OBJETIVOS ESPECÍFICOS:

Análisis y verificación del comportamiento de las variables características de un invernadero (temperatura, luminosidad y humedad relativa).

Implementación y pruebas de un circuito digital basado en microcontroladores que permita evaluar las variables características en el control automático de la temperatura, luminosidad y humedad relativa en un invernadero con la finalidad de proporcionar un funcionamiento eficiente basado en lógica difusa.

Crear una interfaz gráfica de comunicación entre el usuario y la computadora.

Implementación de dispositivos móviles, para facilitar la comunicación entre el usuario y la interfaz.

ALCANCES Y LIMITACIONES.

Alcances:

Se logró diseñar e implementar un circuito para evaluar las variables características en el control de invernadero.

Se pudo crear una interfaz gráfica para el usuario; que muestra los valores de las variables y las registra en una base de datos.

Se realizaron pruebas de funcionamiento con el circuito diseñado y el sistema de adquisición de datos en el laboratorio de microbiología; se logró sensar un pequeño invernadero, que contaba con agave y frijol.

Se probó el prototipo en este pequeño invernadero para verificar el funcionamiento por partes y luego en su conjunto.

Se realizó el análisis de los datos obtenidos del prototipo, para validar el prototipo y verificar su comportamiento

Se implementó el uso de dispositivos móviles para mayor interacción con el usuario.

LIMITACIONES.

El lugar donde se probó el prototipo era muy reducido así que se pudo probar a una pequeña escala. El resultado de las mediciones de las variables no difería mucho debido a que el lugar que se controló era un tanto estable; en contraposición con el clima de Chiapas; así que esto resultó ser una limitación para poder llevar a cabo una interpretación adecuada de los resultados.

HIPÓTESIS

La aplicación de lógica difusa en un invernadero facilita a los usuarios el control de variables; gracias a la lógica difusa el diseño del sistema resulta con bajos costos.

MARCO TEÓRICO

La Lógica Fuzzy o Difusa, es una lógica basada en la teoría de conjuntos que imita el comportamiento de la lógica humana. La facilidad que esto constituye alumbrará los próximos años espectaculares mejoras técnicas en los sistemas de control de nuestra sociedad.

El termino "difuso" procede de la palabra inglesa "fuzz" que sirve para denominar la pelusa que recubre el cuerpo de lo polluelos al poco de salir del huevo. Este término inglés significa "confuso, borroso, indefinido o desenfocado". Este término se traduce por "flou" en francés y "aimai" en japonés. Aunque la teoría de conjuntos difusos presente cierta complejidad, el concepto básico es fácilmente comprensible.

La lógica difusa es una alternativa de control valiosa para procesos que no pueden describirse con un modelo matemático o su desarrollo es muy complejo. El control lógico difuso utiliza una descripción del proceso mediante reglas, o heurística, que son desarrolladas a partir de un conocimiento seguro del proceso. También se utiliza esta tecnología en los sistemas expertos basados en conocimiento.

Historia de la lógica difusa.

Los conjuntos difusos fueron introducidos por primera vez en 1965; la creciente disciplina de la lógica difusa provee por sí misma un medio para acoplar estas tareas. En cierto nivel, la lógica difusa puede ser vista como un lenguaje que permite trasladar sentencias sofisticadas en lenguaje natural a un lenguaje matemático formal. Mientras la motivación original fue ayudar a manejar aspectos imprecisos del mundo real, la práctica temprana de la lógica difusa permitió el desarrollo de aplicaciones prácticas. Aparecieron numerosas publicaciones que presentaban los fundamentos básicos con aplicaciones potenciales. Esta frase marcó una fuerte necesidad de distinguir la lógica difusa de la teoría de probabilidad. Tal como la entendemos ahora, la teoría de conjuntos difusos y la teoría de probabilidad tienen diferentes tipos de incertidumbre.

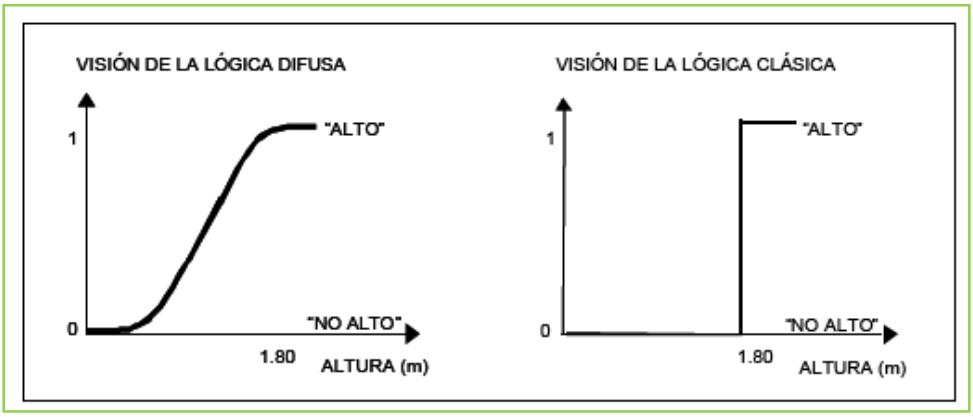
En 1994, la teoría de la lógica difusa se encontraba en la cumbre, pero esta idea no es nueva, para muchos, estuvo bajo el nombre de lógica difusa durante 25 años, pero sus orígenes se remontan hasta 2,500 años. Aún Aristóteles consideraba que existían ciertos grados de veracidad y falsedad. Platón había considerado ya grados de pertenencia.

En el siglo XVIII el filósofo y obispo anglicano Irlandés, George Berkeley y David Hume describieron que el núcleo de un concepto atrae conceptos similares. Hume en particular, creía en la lógica del sentido común, el razonamiento basado en el conocimiento que la gente adquiere en forma ordinaria mediante vivencias en el mundo. En Alemania, Immanuel Kant, consideraba que solo los matemáticos podían proveer definiciones claras, y muchos principios contradictorios no tenían solución. Por ejemplo la materia podía ser dividida infinitamente y al mismo tiempo no podía ser dividida infinitamente. Particularmente la escuela americana de la filosofía llamada pragmatismo fundada a principios de siglo por Charles Sanders Peirce, cuyas ideas se fundamentaron en estos conceptos, fue el primero en considerar ''vaguedades'', más que falso o verdadero, como forma de acercamiento al mundo y a la forma en que la gente funciona.

La idea de que la lógica produce contradicciones fue popularizada por el filósofo y matemático británico Bertrand Russell, a principios del siglo XX. Estudió las vaguedades del lenguaje, concluyendo con precisión que la vaguedad es un grado. El filósofo austríaco Ludwing Wittgenstein estudió las formas en las que una palabra puede ser empleada para muchas cosas que tienen algo en común. La primera lógica de vaguedades fue desarrollada en 1920 por el filósofo JanLukasiewicz, visualizó los conjuntos con un posible grado de pertenencia con valores de 0 y 1, después los extendió a un número infinito de valores entre 0 y 1. En los años sesentas, LoftiZadeh inventó la lógica difusa, que combina los conceptos de la lógica y de los conjuntos de Lukasiewicz mediante la definición de grados de pertenencia.

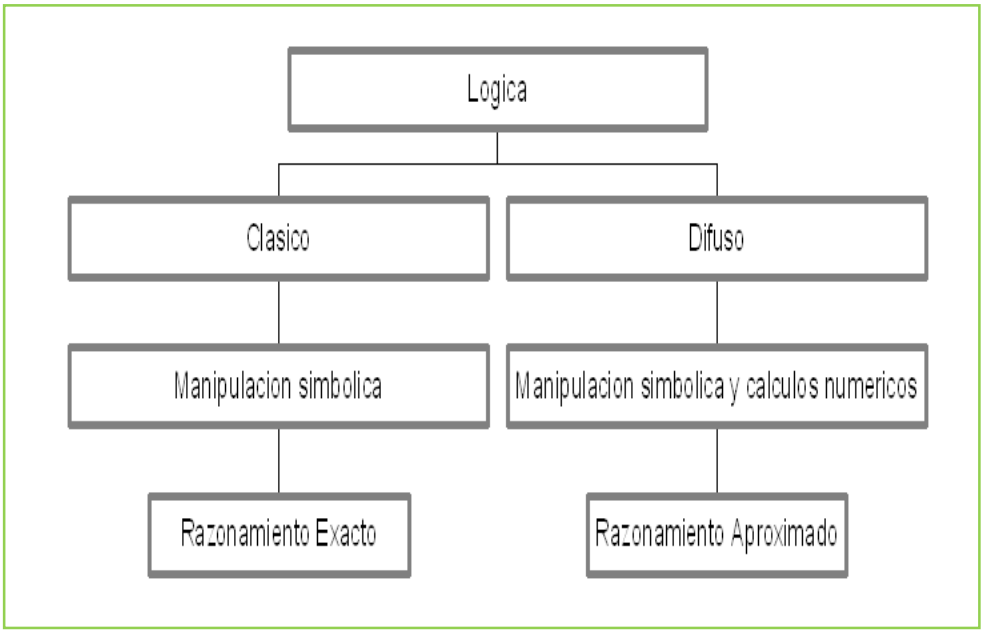
Lógica Difusa en contraposición con la lógica Clásica.

Mientras que la teoría de conjuntos tradicional (pertenece o no pertenece a un determinado conjunto) define ser miembro de un conjunto como un predicado booleano, la teoría de conjunto difusa permite representar el ser miembro de un conjunto como una distribución de posibilidades.



La Lógica Difusa, es una lógica matemática basada en la teoría de conjuntos que posibilita imitar el comportamiento de la lógica humana.

La lógica difusa se utiliza para representar la información imprecisa, ambigua, o vaga. Se utiliza para realizar operaciones en los conceptos que están fuera de las definiciones de la lógica boleana. Un tipo de lógica que reconoce valores verdaderos y falsos más que simples. Con lógica difusa, los subconjuntos se pueden representar con grados de la verdad y de la falsedad. Por ejemplo, la declaración, es hoy soleado, pudo ser el 100% verdad si no hay nubes, 80% verdad si hay algunas nubes, 50% verdad si esta nublado y 0% verdad si llueve todo el día.



Difusividad

- Es una incertidumbre determinativa.
- Esta relacionada al grado con el cual los eventos ocurren sin importar la probabilidad de su ocurrencia.
- Por ejemplo, el grado de juventud de una persona es un evento difuso sin importar que sea un elemento aleatorio.

❖ **Diferencias entre difusividad y probabilidad**

- La difusividad es una incertidumbre determinativa, la probabilidad no es determinativa.
- La incertidumbre probabilística se disipa con el número de ocurrencias y la difusividad no.
- La difusividad describe eventos ambiguos, la probabilidad describe eventos que ocurren. Si un evento ocurre es aleatorio. El grado con el que ocurren es difuso.

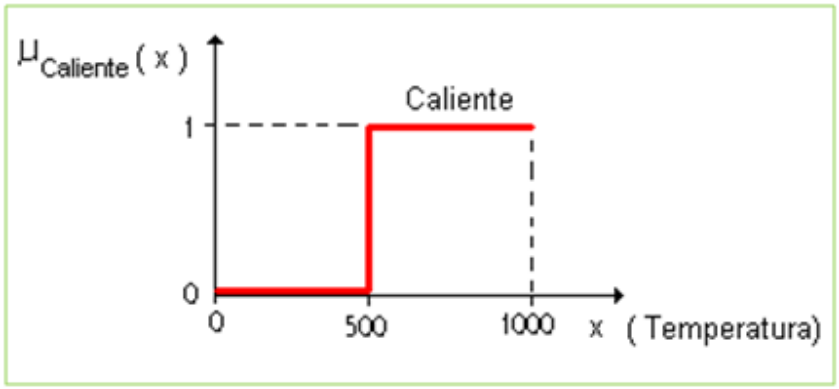
Conjuntos Difusos:

- **Conjuntos Difusos:** Es un conjunto que puede contener elementos con grados parciales de pertenencia, a diferencia de los conjuntos clásicos en los que los elementos pueden “pertenecer” o “no pertenecer” a dichos conjuntos.

  Desde el punto de vista de que se aplican palabras a la definición de cualquier propiedad por ejemplo: mujeres altas, edificios viejos, hombres bajos, elevada inteligencia, baja velocidad, viscosidad moderada… Desde este punto de vista estos valores no podrían ser definidos solo con 2 valores, 0 y 1, se ha de establecer un peso para la característica estableciendo valores intermedios (ejemplo entre 0 y 1 tomando todos los valores intermedios, o bien estableciendo una escala de 0 a 100).

- **Función de Pertenencia:** Es una curva que determina el grado de pertenencia de los elementos de un conjunto. Se denota generalmente por μ y puede adoptar valores entre 0 y 1, inclusive.

  Por ejemplo, para un conjunto clásico tendríamos lo siguiente:

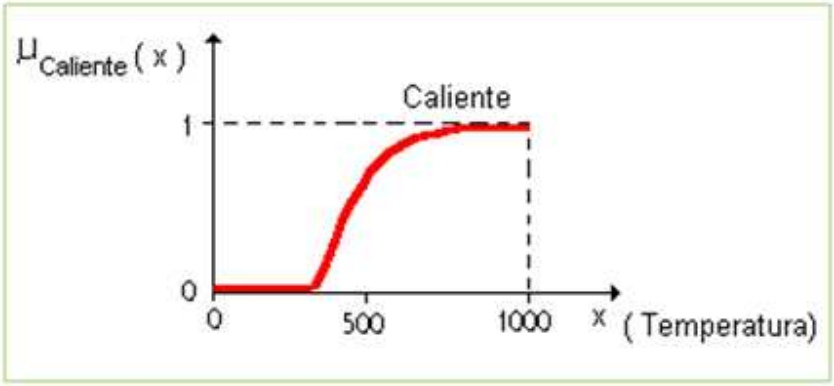


El conjunto se definiría así: **Caliente = { x | x > 500 }** , Temp [°C]

$\mu_{Caliente}(x) = 1,\ x \in$ Caliente **$\mu_{Caliente}(x) = 0,\ x \notin$ Caliente**

$\mu_{Caliente}(x) = \{0, 1\}$ ➔ Solo toma estos valores.

Para un conjunto difuso tendríamos lo siguiente:



El conjunto se definiría así: **Caliente = { (x, $\mu_{Caliente}$(x)) | x ∈ U }**

U = [0 , 1000] ➔ Universo de Discurso.

$\mu_{Caliente}$**(x) = [0 , 1]** ➔ Intervalo de valores Temp [°C]

- **Universo de Discurso:** Conjunto de valores que puede tomar una variable. Este es el conjunto de elementos que vamos a tener en consideración, por ejemplo si se considera que las personas de una comunidad, este universo estará formado por las personas bajas, las personas altas, los hombres con gafas, etc.

    - **Variable difusa**: Es cualquier valor que esta basado en la percepción humana más que en valores precisos de medición (i.e. un color, que esta compuesto en realidad por varias tintas, si la presión de la caldera es excesiva, si la temperatura del agua es la adecuada, si la cantidad de sal que lleva la tortilla es excesiva, si la velocidad de un tren es elevada,…) todas estas dependen de la percepción y están vinculadas con el uso del lenguaje y pueden ser usadas en estructuras del tipo if-then, como por ejemplo: *if*(velocidad es excesiva)*then*(reducir la presión sobre el acelerador).

Operaciones entre Conjuntos Difusos:

Las operaciones básicas entre conjuntos difusos son los siguientes:

- El conjunto complementario Ā de un conjunto difuso A es aquel cuya función característica viene definida por:

$$\mu_A(x) = 1 - \mu_A(x)$$

- La unión de conjuntos difusos A y B es un conjunto difuso A∪B en ∪ cuya función de pertenencia es:

$$\mu_{AUB}(x) = max[\mu_A(x), \mu_B(x)]$$

- La intersección de dos conjuntos difusos A y B es un conjunto difuso A∩B en con función característica:

$$\mu_{A\cap B}(x) = min[\mu_A(x), \mu_B(x)]$$

En estas tres operaciones definidas para conjuntos difusos cumplen, al igual que en la teoría clásica de conjuntos, asociatividad, conmutatividad y distribuida así como las leyes de Morgan.

Sin embargo, también hay que destacar que existen dos leyes fundamentales de la teoría clásica de conjuntos como son el Principio de Contradicción: A∪Ā=U, y el Principio de Exclusión: A∩Ā=∅, que no se cumplen en la teoría de conjuntos difusos.

Las funciones que definen la unión y la intersección de conjuntos difusos pueden generalizarse, a condición de cumplir ciertas restricciones. Las funciones que cumplen con estas condiciones se conocen como Conorma Triangular (T-Conorma) y Norma Triangular (T-Norma). En la mayoría de las aplicaciones de la ingeniería de la lógica se usan como T-Conorma el operador máximo y como T-Norma los operadores mínimos o productos.

Relaciones Difusas:

Una relación difusa representa el grado de presencia o ausencia de interacción o interconexión entre elementos de dos o mas conjuntos difusos, por ejemplo: “x es mayor que y”. Supongamos U y V dos universos de discurso, la relación difusa R(U, V) es un conjunto difuso en el espacio producto Ux V que se caracteriza por la función de pertenencia $\mu_R(x,y)$ donde x pertenece a U e y pertenece a V, es decir:

$$R(U,V) = \{[(x,y), \mu_R(x,y)] | (x,y) \epsilon U \times V\}$$

En el caso de las relaciones difusas $\mu_R(x,y) \in [0,1]$ y en el caso de las relaciones clásicas $\mu_R(x,y) = 0 \; o \; 1$.

Como las relaciones difusas son en si mismas un conjunto difuso en el espacio producto, las operaciones entre conjuntos y las operaciones definidas anteriormente también pueden ser aplicadas a ellas.

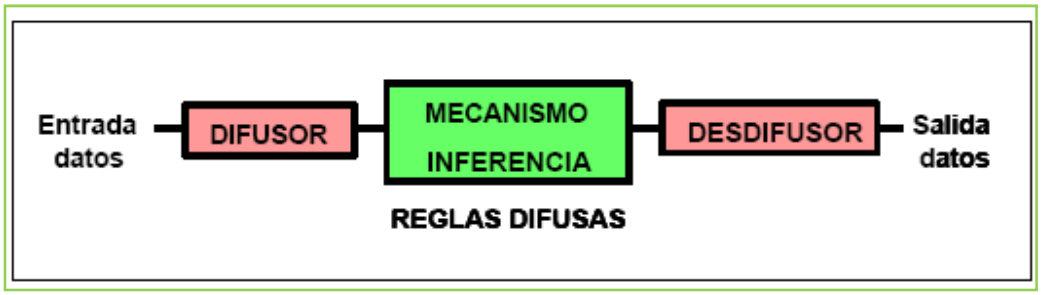
Inferencia Difusa

Se llama reglas difusas al conjunto de proposiciones IF-THEN que modelan al problema que se quiere resolver. Una regla difusa simple tiene la forma: **“Si u es A entonces v es B”** donde A y B son conjuntos difusos definidos en los intervalos “u” y “v” respectivamente. Una regla expresa un tipo de relación entre los conjuntos A y B cuya función característica seria $\mu_{A \to B}(x,y)$ y representa lo que conocemos como implicación lógica. La elección apropiada de esta función característica esta sujeta a las reglas de la lógica proposicional.

Se puede establecer un isomorfismo entre la teoría de conjuntos, la lógica proposicional y el algebra booleana que garantiza que cada teorema enunciado en una de ellas tiene un homólogo en las otras dos. La existencia de estos isomorfismos nos permitirá traducir las reglas difusas a relaciones entre conjuntos difusos y estos a términos de operadores algebraicos con los que podremos trabajar.

Sistema basado en técnicas de Lógica Difusa

El esquema de un sistema basado en técnicas de lógica difusa se presenta en la siguiente figura:



Está compuesto por los siguientes bloques:

- **Bloque Difusor:** Bloque en el que a cada variable de entrada se le asigna un grado de pertenencia a cada uno de los conjuntos difusos que se ha considerado, mediante las funciones características asociadas a estos conjuntos difusos. La entrada a este bloque son valores concretos de las variables de entrada y las salidas son grados de pertenencia a los conjuntos difusos considerados.
- **Bloque de Inferencia:** Bloque que, mediante los mecanismos de inferencia, relaciona conjuntos difusos de entrada y de salida y que representa a las reglas que definen el sistema. Las entrada a este bloque son conjuntos difusos (grados de pertenencia) y las salidas son también conjuntos difusos, asociados a la variable de salida.
- **Difusificador:** Bloque en el cual a partir del conjunto difuso obtenido en el mecanismo de inferencia y mediante los métodos matemáticos de desdifusión, se obtiene un valor concreto de la variable de respuesta, es decir, el resultado.

Mecanismo de Inferencia:

Los mecanismos de inferencia son aquellos en los que se usan los principios de la lógica difusa para realizar un mapeo de los conjuntos difusos de entrada a los conjuntos difusos de salida. Cada regla es interpretada como una implicación difusa. Es decir, el bloque de inferencia es aquel en el cual se realiza la traducción matemática de las reglas difusas; estas reglas modelan el sistema pero para poder trabajar con ellas y extraer un resultado se debe evaluar matemáticamente la información que reflejan. Como ya se ha mencionado, las reglas utilizadas para diseñar un sistema basado en la lógica difusa toman la forma:

"Si u_1 es A_1 y u_2 es A_2 y u_3 es A_3 ENTONCES v es B"

DS1820 SENSOR DIGITAL DE TEMPERATURA

Reporta cambios de temperatura con una precisión de 9 a 12 bits, desde -55C hasta 125C (+/-0.5C). El intervalo de voltajes de alimentación es de 3V a 5.5V. No requiere componentes externos.

Convierte la temperatura a una palabra digital de 12 bits en 750ms (valor máximo). Realiza mediciones desde –55℃ hasta +125℃ (–67℉ hasta +257℉) ±0.5℃ de exactitud desde–10℃ hasta +85℃ .

HIH 4030. SENSOR DE HUMEDAD RELATIVA.

Mide la humedad relativa (% HR) y lo entrega como una tensión de salida analógica. Se puede conectar la salida del sensor directamente a un ADC (Conversor análogo-digital) en un microcontrolador, y, gracias a la salida de tensión lineal del sensor, los datos son muy fáciles de procesar.

Lineal, salida analógica
4 5.8VDC tensión de alimentación
Diseño de bajo poder, corriente típica de sólo 200μA
Mayor precisión
Rápido tiempo de respuesta
Estable

OPT101. Sensor de luz

Voltaje de alimentación: +2.7 to +36V
BAJA Corriente en reposo: 120µA

APLICACIONES
Instrumentación Médica
Laboratorio de instrumentación
Posición y sensores de proximidad
Fotografía analizadores
Detectores de humo

SERVICIO DE MENSAJES CORTOS EN FORMATO PDU

Hay 2 formas de tratar los servicios de mensajes cortos (SMS):

-modo texto
-modo PDU (Protocol Description Unit)

El modo PDU trata el SMS como una cadena de caracteres en octetos hexadecimales o semioctetos decimales, de cuya codificación resulta el SMS en modo texto. La ventaja de modo PDU respecto al modo texto es que en modo texto la aplicación queda limitada a la opción de codificación que se haya preestablecido, en modo PDU se puede implementar cualquier codificación.

La cadena PDU no solo contiene el mensaje, sino que lleva información del centro de servicio SMS (AT+CSCA?), hora de llegada, tipo de mensaje, información sobre el que envía el SMS, vigencia del SMS, nº de caracteres del SMS, tipo de llamada (nacional ó internacional), tipo de alfabeto usado, etc.

La cadena PDU consiste en lo siguiente:

00 Longitud de la información SMSC. Aquí, la longitud es 0, lo que significa que el SMSC almacenado en el teléfono debe ser utilizado.

01 Nota: Este octeto es opcional. En algunos teléfonos este octeto debe omitirse.

11 Primer octeto de la SMS-el mensaje.

00 TP-Message-referencia. El valor "00" le permite establecer el número de referencia del mensaje en el teléfono.

0A Dirección de longitud. Longitud del número de teléfono en hexadecimal (10 decimal)

81 Tipo de Dirección. (91 indica el formato internacional del número de teléfono, 81 formato desconocido).

6911971106 El número de teléfono en semioctetos (9611791160). Debe tomarse en cuenta que este tiene la longitud 10 (A), que es par.

00 TP-PID. Identificador de Protocolo

00 TP-DCS. Codificación de datos scheme. Este mensaje se codifica de acuerdo con el alfabeto por defecto a 7 bits.

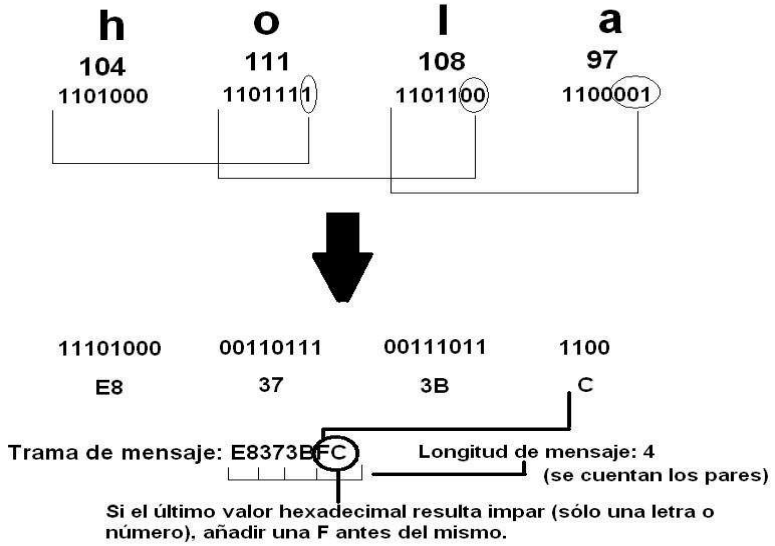
AA TP-Periodo de validez. "AA" significa 4 días.

0A TP-User-Data-Longitud. Longitud del mensaje. El TP-DCS campo indicado 7-bit de datos, por lo que la longitud de aquí es el número de septetos (10).

04E8373BFC TP-User-Data. Mensaje codificado a 7 bits, estos octetos representan la palabra “hola”

Ejemplo de trama enviando la palabra “hola”:

0011000A8169119711060000AA04E8373BFC



Para explicar la codificación A 7 BITS usaremos un ejemplo. Codificación de la palabra “HOLA”:

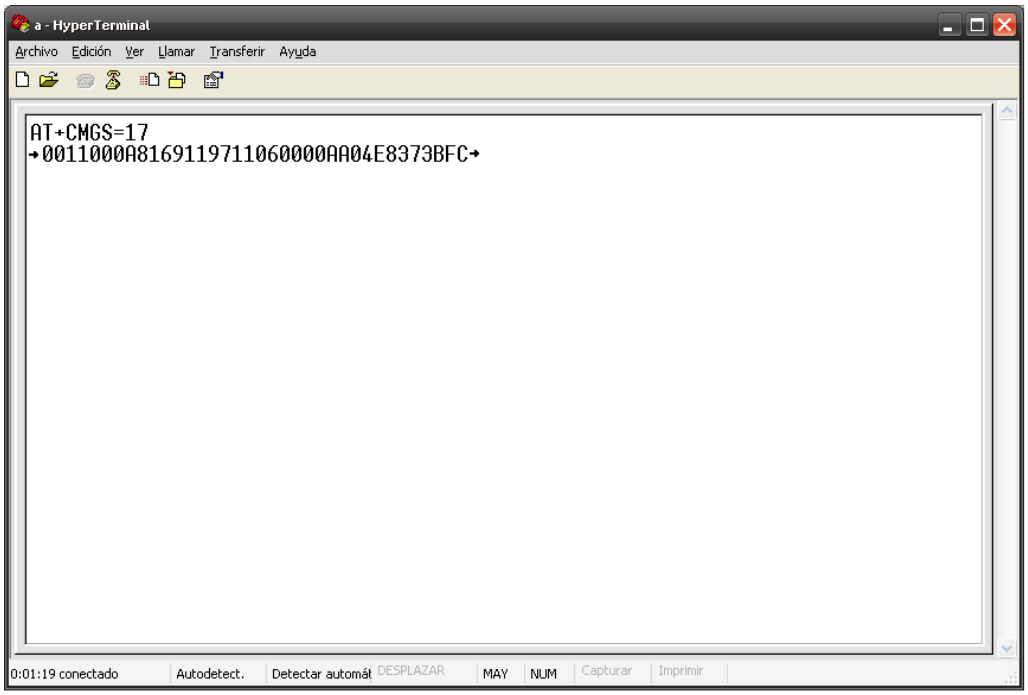
	H	O	L	A
Hex	48	4F	4C	41
Bin	1001000	1001111	1001100	1000001

Para transformarlo a octetos se toma el número de caracteres de la siguiente letra que nos falten para llegar a 8, cuando se hayan tomado caracteres de una letra para la anterior, esta se queda sin esos caracteres y los debe tomar de la siguiente letra

1 1001000	00 100111 ~~1~~	001 10011 ~~00~~	1000 ~~001~~
C8	27	33	08

El último carácter generalmente queda incompleto. Al terminar de completar todos los caracteres para que queden con 8 bits (menos el último), se pasan a formato hexadecimal y el resultado en conjunto es el mensaje codificado a 7 bits de la palabra “HOLA”.

Ejemplo de envío con hyperterminal:



PROCEDIMIENTO Y ACTIVIDADES REALIZADAS.

1.-**Se investigó y recopiló información del control automático de los sistemas de invernadero:** Se recabó información de diferentes sistemas de invernadero verificando trabajos similares.

2.-**Se Caracterizaron los principales sensores de tipo industrial:** Se organizó la información recabada para establecer los parámetros y condiciones que se utilizaron para seleccionar los sensores de temperatura, humedad relativa y luminosidad.

3.-**Se diseñó e implementó una tarjeta electrónica para evaluar las variables características en el control de un invernadero.** Se seleccionó el diseño más adecuado que permitió la implementación en un circuito electrónico.

4**.-Se diseñó y programó la interfaz gráfica:** Se programó la interfaz gráfica para la pc, que registra los datos, los muestra en pantalla y permite la modificación por el usuario en tiempos de muestreo.

5.-**Se realizaron pruebas de funcionamiento con la tarjeta electrónica y el sistema de adquisición de datos utilizado.**

6.-**Se implementaron dispositivos móviles:** Se hizo uso de comandos at para envíos de mensajes a celulares; esto con la finalidad de hacerle más fácil y cómoda al usuario la lectura de las variables sensadas.

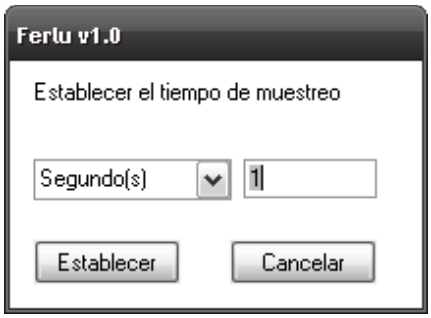
7.-**Se analizaron los datos obtenidos del prototipo:** Se efectuaron las pruebas finales al sistema con el objetivo de analizar los datos.

PROGRAMAS

CODIGO C #

Módulo: Configuración.cs

Este módulo convierte un número entero en el tiempo correspondiente dado (hora, minuto y segundo). Cuenta con una función que almacena el tipo de formato establecido para el tiempo; así como el valor entero. También se cuenta con una función para retornar dichos valores. Este valor nos sirve para establecer el tiempo de muestreo.



Código:

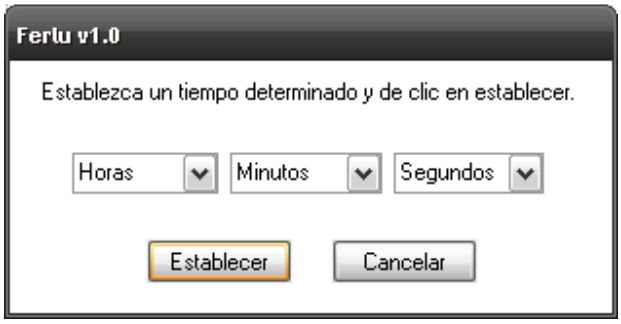
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Control_III
{
   public partial class Configuracion : Form
   {
      String Tipo;
      String t;
      int Tiempo = 0;
      public Configuracion()
      {
         InitializeComponent();
      }

      public String Regresa_Tipo()
      {
```

```
            return Tipo;
        }

        public String Regresa_Tiempo_String()
        {
            return t;
        }

        public int Regresa_Tiempo()
        {
            return Tiempo;
        }

        private void button1_Click(object sender, EventArgs e)
{
            int tiempo;
            tiempo = Convert.ToInt32(txtTiempo.Text);
            t = txtTiempo.Text;
            Tipo = cmbTipo.Text;

if (cmbTipo.Text.Equals("Segundo(s)"))
{
                Tiempo = tiempo;
            }
            else if (cmbTipo.Text.Equals("Minuto(s)"))
            {
                Tiempo = tiempo * 60;
}
            else if (cmbTipo.Text.Equals("Hora(s)"))
            {
                Tiempo = tiempo * 60 * 60;
            }
            this.Close();
        }

        private void Configuracion_FormClosed(object sender, FormClosedEventArgs e)
        {
            if (txtTiempo.Text.Equals(""))
                Tiempo = 0;
        }

        private void button2_Click(object sender, EventArgs e)
{
            Tiempo = 0;
            this.Close();
        }
```

```
    }
}
```

Módulo: ConfigurarDetener.cs

Este módulo almacena tres variables de tiempo (hora, minuto y segundo) para posteriormente retornarlas a través de una función. Estos valores nos permiten establecer un tiempo para detener el muestreo.



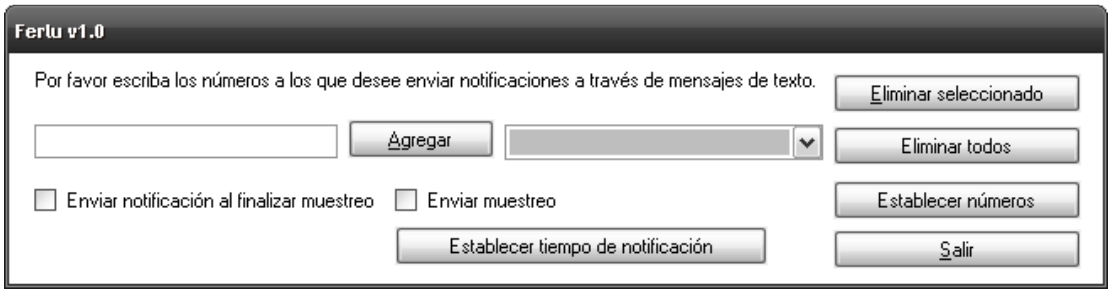
Código:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Control_III
{
    public partial class ConfigurarDetener : Form
    {
public ConfigurarDetener()
        {
            InitializeComponent();
        }

        int horas = 0, minutos = 0, segundos = 0;

        public int regresa_horas()
        {
            return horas;
```

```
        }

        public int regresa_minutos()
        {
            return minutos;
        }

        public int regesa_segundos()
        {
            return segundos;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (comboBox1.Text.Equals("Horas") || comboBox2.Text.Equals("Minutos") ||
comboBox3.Text.Equals("Segundos"))
            {
                MessageBox.Show("Especifique un tiempo en horas, minutos y segundos por
favor", "Control III", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
            else
            {
                horas = Convert.ToInt32(comboBox1.Text);
                minutos = Convert.ToInt32(comboBox2.Text);
                segundos = Convert.ToInt32(comboBox3.Text);
                this.Close();
            }
        }

        private void button2_Click(object sender, EventArgs e)
{
            this.Close();
        }
    }
}
```

Módulo: EstablecerNumeros.cs

Este módulo nos sirve para almacenar números de celulares (previa validación).Así como también un par de variables que nos indican dos tipos de notificaciones, mismas que le indicarán al programa si envía un mensaje de texto a los celulares establecidos al finalizar el muestreo (Notificación 1) ó por cada n-tiempo (si esta notificación se activa, es necesario establecer un tiempo) .



Código:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Control_III
{
   public partial class EstablecerNumeros : Form
   {
      int contador_Numeros = 0;
      String[] Numeros_Cel = new String[100];
bool dialogResultado = false;
      bool notificacionMuestras = false;
      bool notificacionFinalizar = false;
      int hora = 0, minuto = 0, segundo = 0;
      bool tiempo_timer = false;

      public int regresaHora()
      {
         return hora;
      }

      public int regresaTotalNumeros()
```

```
        {
            return contador_Numeros;
        }

        public int regresaMinuto()
        {
            return minuto;
        }

        public int regresaSegundo()
        {
            return segundo;
        }

        public EstablecerNumeros()
        {
            InitializeComponent();
        }

        public bool Notificar_Muestras()
        {
            return notificacionMuestras;
        }

        public bool Notificar_Finalizacion_De_Muestras()
{
            return notificacionFinalizar;
        }

        private void btnAgregar_Click(object sender, EventArgs e)
        {
            if (txtAgregar.Text.Equals(""))
MessageBox.Show("Por favor, rellene el campo con un número de celular válido",
"Control III", MessageBoxButtons.OK, MessageBoxIcon.Information);
else if(txtAgregar.Text.Length != 10)
                MessageBox.Show("Por favor, rellene el campo con un número de celular
válido", "Control III", MessageBoxButtons.OK, MessageBoxIcon.Information);
            else
            {
                contador_Numeros++;
                cmbLista.Items.Add(txtAgregar.Text);
                txtAgregar.Text = "";
                txtAgregar.Focus();
}
        }
```

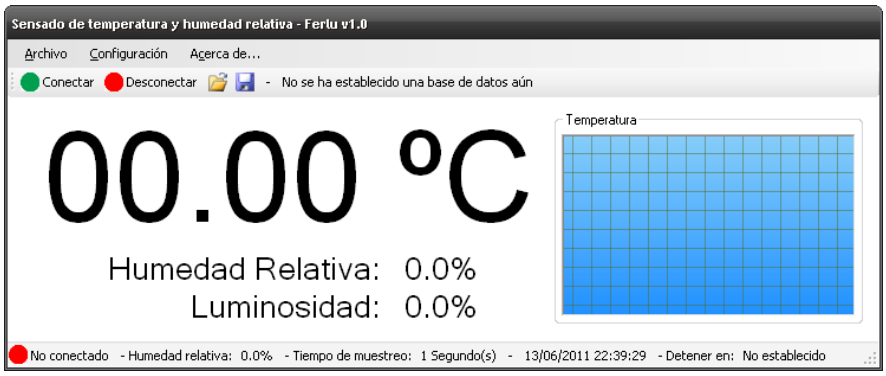
```
        private void btnEliminar_Click(object sender, EventArgs e)
{
            if (cmbLista.SelectedIndex == -1)
                MessageBox.Show("Por favor, seleccione un número para eliminarlo de la
lista", "Control III", MessageBoxButtons.OK, MessageBoxIcon.Information);
else
            {
                cmbLista.Items.RemoveAt(cmbLista.SelectedIndex);
                contador_Numeros--;
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            cmbLista.Items.Clear();
            contador_Numeros = 0;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button3_Click(object sender, EventArgs e)
{
            if (contador_Numeros != 0)
            {
                if (Notificar_Finalizacion_De_Muestras() == false && Notificar_Muestras() ==
false)
                    MessageBox.Show("Por favor, debe seleccionar que tipo de notificación
desea para poder establecer los números celulares", "Ferlu v1.0",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                else
                {
                    if (Notificar_Muestras() == true && Tiempo_Timer_Establecido() == false)
                    {
                        MessageBox.Show("Debe establecer un tiempo para enviar la
notificación de las muestras", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Information);
return;
                    }
                    for (int i = 0; i < contador_Numeros; i++)
                        Numeros_Cel[i] = cmbLista.Items[i].ToString();
                    dialogResultado = true;
                    this.Close();
                }
```

```
        }
        else
            MessageBox.Show("No se han agregado números a la lista", "Ferlu v1.0",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    public bool dialogResult()
    {
        return dialogResultado;
    }

    public String[] Numeros_Celulares()
    {
        return Numeros_Cel;
    }

    private void button4_Click(object sender, EventArgs e)
{
        ConfigurarDetener detener = new ConfigurarDetener();
        detener.ShowDialog();
        hora = detener.regresa_horas();
        minuto = detener.regresa_minutos();
        segundo = detener.regesa_segundos();

        if (hora == 0 && minuto == 0 && segundo == 0)
        {
        }
        else
            tiempo_timer = true;
    }

    public bool Tiempo_Timer_Establecido()
{
        return tiempo_timer;
    }

    private void checkBox1_CheckedChanged(object sender, EventArgs e)
    {
        notificacionFinalizar = checkBox1.Checked;
    }

    private void checkBox2_CheckedChanged(object sender, EventArgs e)
{
        notificacionMuestras = checkBox2.Checked;
    }
  }
```

}

Módulo: Form1.cs

Es la interfaz principal mediante la cual se visualizan las lecturas proporcionadas por los sensores. Las funciones que se encuentran en este módulo son: Almacenamiento de los registros en un archivo de base de datos, apertura de un archivo de base de datos, apertura de conexión con la interfaz, cierre de la conexión con la interfaz, establecimiento del puerto usado para la interfaz, establecimiento del tiempo de muestreo, establecimiento de un tiempo de finalización del muestreo, apertura de conexión con el celular, cierre de conexión con el celular, establecimiento del puerto del celular, establecimientos de números y notificaciones para el celular, llamada de prueba.



Código:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.IO.Ports;
using System.Data.OleDb;
using System.Reflection;
//using Microsoft.Office.Interop.Excel;

namespace Control_III
{

    public partial class Form1 : Form
```

```
{
        int hora2 = 0, minuto2 = 0, segundo2 = 0, hora22 = 0, minuto22 = 0, segundo22 =
0;
        int TotalNumeros = 0;
        bool notificarMuestras = false;
        bool notificarFinalizacion = false;
        String[] Numeros_Celulares = new String[100];
        String Recibidos;
        String Ruta_De_Base;
        bool base_creada = false;
        String strConexion;
OleDbConnection cnnConexion;
        OleDbCommand cmdTema;
        String temperatura_recibida = "00.00";
String humedad_relativa_recibida = "00.00";
        int bandera = 0;
        float humedad_relativa;
        int luminosidad = 0;
        /* Se incrementa cada segundo */
        int tiempo_timer = 0;
        int hora = 0, minuto = 0, segundo = 0;
        bool base_abierta = false;

public Form1()
        {
            InitializeComponent();
        }

        public bool Base_isOpen()
        {
            return base_abierta;
        }

        public void Conectar_Base(string ruta)
        {
            String auxiliar = "";
            int longitud;

            longitud = ruta.Length;

            for (int i = 0; i < longitud; i++)
            {
                if (ruta.Substring(i, 1).Equals("\\"))
{
                    auxiliar = auxiliar + "\\\\";
                }
```

```
            else
                auxiliar = auxiliar + ruta.Substring(i, 1);
        }

        ruta = auxiliar;

strConexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + ruta + ";";
        cnnConexion = new OleDbConnection(strConexion);
        cmdTema = new OleDbCommand();
        cnnConexion.Open();

        cmdTema.Connection = cnnConexion;
        cmdTema.CommandType = CommandType.Text;
        base_abierta = true;
}

    public void Cerrar_Base()
    {
        cnnConexion.Close();
        base_abierta = false;
    }

    public void Insertar_Registro(String temperatura, String fechahora, String
humedad_relativa, String luminosidad)
    {
        cmdTema.CommandText = "insert into
temperatura(Temperatura,FechaHora,HumedadRelativa, Luminosidad) values('" +
temperatura + "','" + fechahora + "','" + humedad_relativa + "','" + luminosidad + "')";
cmdTema.ExecuteNonQuery();
    }

    public void Crear_Tabla()
{
        cmdTema.CommandText = "CREATE TABLE temperatura(Temperatura
CHAR(50),FechaHora DATETIME, HumedadRelativa CHAR(50), Luminosidad
CHAR(50))";
cmdTema.ExecuteNonQuery();
    }

    private void Recepcion(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
    {
        if (serialPort1.IsOpen)
        {
            Recibidos = serialPort1.ReadLine();
            this.Invoke(new EventHandler(Actualizar));
```

```
        }
    }

    public String Humedad_String(float humedad)
    {
        String auxiliar = "";
        String humedad_relativa_string = "";

        auxiliar = Convert.ToString(humedad);

        humedad_relativa_string = "" + auxiliar.Substring(0, 1) + auxiliar.Substring(1, 1)
+ "." + auxiliar.Substring(3, 1) + auxiliar.Substring(4, 1);

        return humedad_relativa_string;
    }

    public Decimal Temperatura_Decimal(String Temperatura)
    {
        String auxiliar = "";
        Decimal t = 0;

        auxiliar = "" + Temperatura.Substring(0, 1) + Temperatura.Substring(1, 1) + "," +
Temperatura.Substring(3, 1) + Temperatura.Substring(4, 1);
t = Convert.ToDecimal(auxiliar);

        return t;
    }

    private void Actualizar(object s, EventArgs e)
    {
        if (bandera == 0)
        {
            bandera = 1;
            temperatura_recibida = Recibidos;
            /* Bandera 0 es para temperatura*/
            perfChart.AddValue(Temperatura_Decimal(temperatura_recibida));
            lbTemperatura.Text = "" + temperatura_recibida + " ºC";
        }
        else if (bandera == 1)
        {
            bandera = 2;
            humedad_relativa = Convert.ToSingle((((Convert.ToInt32(Recibidos) * 5.3) /
1023) - 0.958) / 0.0307);
            humedad_relativa_recibida = Humedad_String(humedad_relativa);
            /* Bandera 1 es para humedad relativa */
mlbRelativa.Text = "" + Humedad_String(humedad_relativa) + "%";
```

```
            lbRelativa.Text = "" + Humedad_String(humedad_relativa) + "%";
        }
        else if (bandera == 2)
{
            bandera = 0;
            luminosidad = Convert.ToInt32((Convert.ToInt32(Recibidos)*100)/1023);
            lbLuminosidad.Text = "" + luminosidad + "%";
            Insertar_Registro(temperatura_recibida, System.DateTime.Now.ToString(),
humedad_relativa_recibida, Convert.ToString(luminosidad));
}
    }

    private void EjecutarMacro(string archivoExcel, string nombreMacro)
    {
        /*
        ApplicationClass excel;
        Workbooks books;
        _Workbook book;
        object missing;

        missing = System.Reflection.Missing.Value;

        excel = new ApplicationClass();
        excel.Visible = true;

        books = excel.Workbooks;
        book = books.Open(archivoExcel, missing, missing, missing, missing, missing,
missing, missing, missing, missing,
                    missing, missing, missing, missing, missing, missing);

        excel.GetType().InvokeMember("Run", System.Reflection.BindingFlags.Default |
System.Reflection.BindingFlags.InvokeMethod, null, excel, new object[] { "Borrar" });
        excel.GetType().InvokeMember("Run", System.Reflection.BindingFlags.Default |
System.Reflection.BindingFlags.InvokeMethod, null, excel, new object[] { nombreMacro
});
        */

    }


    private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
    {
    }

    private void button1_Click(object sender, EventArgs e)
```

```
        {
        }

        private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
        }

        private void conectarToolStripMenuItem_Click(object sender, EventArgs e)
{
            if (cbPuerto.Text.Equals(""))
                MessageBox.Show("Debe especificar un puerto de comunicaciones antes de
abrir la conexión.", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Information);
else
            {
                if (!serialPort1.IsOpen)
                {
                    if (base_creada == true)
{
                        if (Base_isOpen() == false)
                            Conectar_Base(Ruta_De_Base);
try
                        {
                            if (tiempo_timer == 1)
                                timer2.Enabled = true;
                            if (notificarMuestras == true)
                                timer3.Enabled = true;
                            serialPort1.Open();
                            lbConexion.Visible = false;
                            lbConexion2.Visible = true;
                            lbConexion3.Text = "Conectado";
                        }
                        catch (System.Exception ex)
                        {
                            MessageBox.Show("No se puede abrir el puerto \n " + ex.ToString(),
"Ferlu v1.0", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                        serialPort1.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(Recepcion);
}
                    else
                    {
                        MessageBox.Show("Debe establecer una ubicación para guardar la
base de datos de muestras.", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Error);
```

```
                }
            }
            else
                MessageBox.Show("El puerto de comunicaciones se encuentra abierto",
"Ferlu v1.0", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
        }

        private void desconectarToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                if (Base_isOpen() == true)
Cerrar_Base();
                serialPort1.Close();
                lbConexion.Visible = true;
                lbConexion2.Visible = false;
                lbConexion3.Text = "No conectado";
                Cerrar_Base();
                timer2.Enabled = false;
                timer3.Enabled = false;
                tiempo_timer = 0;
                mlbTiempoDetener.Text = "No establecido";
            }
            else
                MessageBox.Show("El puerto de comunicaciones se encuentra cerrado",
"Ferlu v1.0", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

        private void salirToolStripMenuItem_Click(object sender, EventArgs e)
        {

        }

        private void btnProporcional_Click(object sender, EventArgs e)
        {
        }

        private void btnIntegral_Click(object sender, EventArgs e)
        {
        }

        private void btnDerivada_Click(object sender, EventArgs e)
        {
        }
```

```
        private void btnReferencia_Click(object sender, EventArgs e)
        {
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            int numero;
            string[] ports = SerialPort.GetPortNames();

//Conectar_Base();

            numero = cbPuerto.Items.Count;

if (numero != 0)
                for (int i = 0; i < numero; i++)
                    cbPuerto.Items.RemoveAt(0);

            foreach (string port in ports)
            {
                cbPuerto.Items.Add(port);
            }

            //cbPuerto.ComboBox.SelectedIndex = 0;
            //serialPort1.PortName = Convert.ToString(cbPuerto.SelectedItem);
        }

        private void cbPuerto_SelectedIndexChanged(object sender, EventArgs e)
        {
            if(serialPort1.IsOpen == false)
                serialPort1.PortName = cbPuerto.Text;
else
                MessageBox.Show("Debe cerrar el puerto de comunicaciones antes de
cambiar esta propiedad", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Information);
}

        private void acercaDeToolStripMenuItem_Click(object sender, EventArgs e)
{
            Presentación f = new Presentación();
            f.Show();
}

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (serialPort1.IsOpen)
            {
```

```
            serialPort1.Close();
lbConexion.Visible = true;
            lbConexion2.Visible = false;
            lbConexion3.Text = "No conectado";
}
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            int numero;
            string[] ports = SerialPort.GetPortNames();

            numero = cbPuerto.Items.Count;

            if(numero != 0)
                for (int i = 0; i < numero; i++)
                    cbPuerto.Items.RemoveAt(0);

            foreach (string port in ports)
            {
                cbPuerto.Items.Add(port);
            }
        }

        private void actualizarListaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            int numero;
            string[] ports = SerialPort.GetPortNames();

            numero = cbPuerto.Items.Count;

            if (numero != 0)
                for (int i = 0; i < numero; i++)
                    cbPuerto.Items.RemoveAt(0);

            foreach (string port in ports)
            {
                cbPuerto.Items.Add(port);
            }

            //cbPuerto.ComboBox.SelectedIndex = 0;
            //serialPort1.PortName = Convert.ToString(cbPuerto.SelectedItem);
        }

        private void serialPort1_PinChanged(object sender, SerialPinChangedEventArgs
e)
```

```
        {
        }

        private void timer1_Tick(object sender, EventArgs e)
        {

        }

        private void timer2_Tick(object sender, EventArgs e)
        {

        }

        private void timer_integral_Tick(object sender, EventArgs e)
        {

        }

        private void timer_derivativa_Tick(object sender, EventArgs e)
        {

        }

        private void timer_referencia_Tick(object sender, EventArgs e)
        {

        }

        private void graficarToolStripMenuItem_Click(object sender, EventArgs e)
{
            Configuracion configura = new Configuracion();
            configura.ShowDialog();
if (configura.Regresa_Tiempo() != 0)
            {
                if (serialPort1.IsOpen == true)
{
                    serialPort1.Write(Convert.ToString(configura.Regresa_Tiempo()) + "f");
mlbTiempo.Text = "" + configura.Regresa_Tiempo_String() + " " +
configura.Regresa_Tipo();
}
                else
                {
                    serialPort1.Open();
                    serialPort1.Write(Convert.ToString(configura.Regresa_Tiempo()) + "f");
mlbTiempo.Text = "" + configura.Regresa_Tiempo_String() + " " +
configura.Regresa_Tipo();
```

```
System.Threading.Thread.Sleep(1000);
                serialPort1.Close();
//MessageBox.Show("El puerto de comunicaciones se encuentra cerrado", "Control III",
MessageBoxButtons.OK, MessageBoxIcon.Information);
}
            }
        }

        private void salirToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                serialPort1.Close();
lbConexion.Visible = true;
                lbConexion2.Visible = false;
                lbConexion3.Text = "No conectado";
Cerrar_Base();
            }
            this.Close();
        }

        private void timer1_Tick_1(object sender, EventArgs e)
        {
        }

        private void timer1_Tick_2(object sender, EventArgs e)
        {
            mlbReloj.Text = "" + System.DateTime.Now.ToString();
        }

        private void toolStripButton2_Click(object sender, EventArgs e)
{
            if (cbPuerto.Text.Equals(""))
                MessageBox.Show("Debe especificar un puerto de comunicaciones antes de
abrir la conexión.", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Information);
else
            {
                if (!serialPort1.IsOpen)
                {
                    if (base_creada == true)
{
                        if (Base_isOpen() == false)
                            Conectar_Base(Ruta_De_Base);
try
                        {
```

```
                    if (tiempo_timer == 1)
                        timer2.Enabled = true;
                    if (notificarMuestras == true)
                        timer3.Enabled = true;
                    serialPort1.Open();
                    lbConexion.Visible = false;
                    lbConexion2.Visible = true;
                    lbConexion3.Text = "Conectado";
                }
                catch (System.Exception ex)
                {
                    MessageBox.Show("No se puede abrir el puerto \n " + ex.ToString(),
"Ferlu v1.0", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                serialPort1.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(Recepcion);
}
            else
            {
                MessageBox.Show("Debe establecer una ubicación para guardar la
base de datos de muestras.", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        else
            MessageBox.Show("El puerto de comunicaciones se encuentra abierto",
"Ferlu v1.0", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
    }

    private void toolStripButton3_Click(object sender, EventArgs e)
    {
        if (serialPort1.IsOpen)
        {
            if (Base_isOpen() == true)
Cerrar_Base();
            serialPort1.Close();
            lbConexion.Visible = true;
            lbConexion2.Visible = false;
            lbConexion3.Text = "No conectado";
            Cerrar_Base();
            timer2.Enabled = false;
            timer3.Enabled = false;
            tiempo_timer = 0;
            mlbTiempoDetener.Text = "No establecido";
        }
```

```
        else
            MessageBox.Show("El puerto de comunicaciones se encuentra cerrado",
"Ferlu v1.0", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

      private void toolStripButton1_Click(object sender, EventArgs e)
{
        if (serialPort1.IsOpen)
        {
            MessageBox.Show("La aplicación se encuentra en ejecución, debe detenerla
para establecer una nueva ubicación.", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Error, MessageBoxDefaultButton.Button1);
}
        else
        {
            if (base_creada == true)
{
                if (MessageBox.Show("Ya existe una base de datos creada, ¿Desea elegir
una ubicación y nombre diferente para la base de datos?", "Ferlu v1.0",
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2) == DialogResult.Yes)
                {
                    Cerrar_Base();
                    saveFileDialog1.Filter = "Base de datos Access (*.mdb)|*.mdb";

if (saveFileDialog1.ShowDialog() == DialogResult.OK)
                    {
                        ADOX.CatalogClass cat = new ADOX.CatalogClass();
                        cat.Create("Provider = Microsoft.Jet.OLEDB.4.0;" + "Data Source = " +
saveFileDialog1.FileName);
Ruta_De_Base = saveFileDialog1.FileName;
                        Conectar_Base(saveFileDialog1.FileName);
                        Crear_Tabla();
                        base_creada = true;
                        mlbMensaje.Text = "Base de datos establecida en: " +
saveFileDialog1.FileName;
                    }
                }
            }
            else
            {
                saveFileDialog1.Filter = "Base de datos Access (*.mdb)|*.mdb";

                if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
```

```
                ADOX.CatalogClass cat = new ADOX.CatalogClass();
                cat.Create("Provider = Microsoft.Jet.OLEDB.4.0;" + "Data Source = " +
saveFileDialog1.FileName);
Ruta_De_Base = saveFileDialog1.FileName;
                Conectar_Base(saveFileDialog1.FileName);
                Crear_Tabla();
                base_creada = true;
                mlbMensaje.Text = "Base de datos establecida en: " +
saveFileDialog1.FileName;
}
            }
        }
    }

    private void guardarDatosToolStripMenuItem_Click(object sender, EventArgs e)
{
        if (serialPort1.IsOpen)
        {
            MessageBox.Show("La aplicación se encuentra en ejecución, debe detenerla
para establecer una nueva ubicación.", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Error, MessageBoxDefaultButton.Button1);
}
        else
        {
            if (base_creada == true)
{
                if (MessageBox.Show("Ya existe una base de datos creada, ¿Desea elegir
una ubicación y nombre diferente para la base de datos?", "Ferlu v1.0",
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2) == DialogResult.Yes)
                {
                    Cerrar_Base();
                    saveFileDialog1.Filter = "Base de datos Access (*.mdb)|*.mdb";

if (saveFileDialog1.ShowDialog() == DialogResult.OK)
                    {
                        ADOX.CatalogClass cat = new ADOX.CatalogClass();
                        cat.Create("Provider = Microsoft.Jet.OLEDB.4.0;" + "Data Source = " +
saveFileDialog1.FileName);
Ruta_De_Base = saveFileDialog1.FileName;
                        Conectar_Base(saveFileDialog1.FileName);
                        Crear_Tabla();
                        base_creada = true;
                        mlbMensaje.Text = "Base de datos establecida en: " +
saveFileDialog1.FileName;
                    }
```

```
            }
        }
        else
        {
            saveFileDialog1.Filter = "Base de datos Access (*.mdb)|*.mdb";

            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{

                ADOX.CatalogClass cat = new ADOX.CatalogClass();
                cat.Create("Provider = Microsoft.Jet.OLEDB.4.0;" + "Data Source = " +
saveFileDialog1.FileName);
Ruta_De_Base = saveFileDialog1.FileName;
                Conectar_Base(saveFileDialog1.FileName);
                Crear_Tabla();
                base_creada = true;
                mlbMensaje.Text = "Base de datos establecida en: " +
saveFileDialog1.FileName;
}
        }
      }
    }

    private void toolStripButton4_Click(object sender, EventArgs e)
{
        if (serialPort1.IsOpen)
        {
            MessageBox.Show("La aplicación se encuentra en ejecución, debe detenerla
para establecer una nueva ubicación.", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Error, MessageBoxDefaultButton.Button1);
}
        else
        {
            if (base_creada == true)
{
                if (MessageBox.Show("Ya existe una base de datos creada, ¿Desea elegir
una ubicación y nombre diferente para la base de datos?", "Ferlu v1.0",
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2) == DialogResult.Yes)
                {
                    Cerrar_Base();
                    openFileDialog1.Filter = "Base de datos Access (*.mdb)|*.mdb";

                    if (openFileDialog1.ShowDialog() == DialogResult.OK)
                    {
                        Ruta_De_Base = openFileDialog1.FileName;
```

```
                    Conectar_Base(openFileDialog1.FileName);
                    base_creada = true;
                    mlbMensaje.Text = "Base de datos establecida en: " +
openFileDialog1.FileName;
                }
            }
        }
        else
        {
            openFileDialog1.Filter = "Base de datos Access (*.mdb)|*.mdb";

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                Ruta_De_Base = openFileDialog1.FileName;
                Conectar_Base(openFileDialog1.FileName);
                base_creada = true;
                mlbMensaje.Text = "Base de datos establecida en: " +
openFileDialog1.FileName;
}
            }
        }
    }

    private void abrirBaseDeDatosExistenteToolStripMenuItem_Click(object sender,
EventArgs e)
{
        if (serialPort1.IsOpen)
        {
            MessageBox.Show("La aplicación se encuentra en ejecución, debe detenerla
para establecer una nueva ubicación.", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Error, MessageBoxDefaultButton.Button1);
}
        else
        {
            if (base_creada == true)
{
                if (MessageBox.Show("Ya existe una base de datos creada, ¿Desea elegir
una ubicación y nombre diferente para la base de datos?", "Ferlu v1.0",
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2) == DialogResult.Yes)
                {
                    Cerrar_Base();
                    openFileDialog1.Filter = "Base de datos Access (*.mdb)|*.mdb";

                    if (openFileDialog1.ShowDialog() == DialogResult.OK)
                    {
```

```
                    Ruta_De_Base = openFileDialog1.FileName;
                    Conectar_Base(openFileDialog1.FileName);
                    base_creada = true;
                    mlbMensaje.Text = "Base de datos establecida en: " +
openFileDialog1.FileName;
                }
            }
        }
        else
        {
            openFileDialog1.Filter = "Base de datos Access (*.mdb)|*.mdb";

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                Ruta_De_Base = openFileDialog1.FileName;
                Conectar_Base(openFileDialog1.FileName);
                base_creada = true;
                mlbMensaje.Text = "Base de datos establecida en: " +
openFileDialog1.FileName;
}
            }
        }
    }

    private void timer2_Tick_1(object sender, EventArgs e)
{
        if (hora == 0 && minuto == 0 && segundo == 59)
        {
            timer2.Enabled = false;
            ParaDetener detener = new ParaDetener();
            detener.ShowDialog();

            if (detener.detener() == 0)
            {
                serialPort1.Close();
                lbConexion.Visible = true;
                lbConexion2.Visible = false;
                lbConexion3.Text = "No conectado";
                Cerrar_Base();
                //Trama_PDU(txtCelular.Text, "El tiempo de muestreo ha concluido.");
timer3.Enabled = false;
                if (notificarFinalizacion == true)
                    for (int i = 0; i < TotalNumeros; i++)
Trama_PDU(Numeros_Celulares[i], "El tiempo ha concluido. Temperatura: " +
temperatura_recibida + " - H.Rel: " + humedad_relativa_recibida + " - Luminosidad: " +
luminosidad + " - Ferlu v1.0");
```

```
                MessageBox.Show("El tiempo de muestreo ha concluido", "Ferlu v1.0",
MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1);
                tiempo_timer = 0;
                mlbTiempoDetener.Text = "No establecido";
            }
            else
            {
                timer2.Enabled = false;
                tiempo_timer = 0;
                mlbTiempoDetener.Text = "No establecido";
            }
        }
        else if (hora == 0 && minuto == 0 && segundo == 0)
{
            serialPort1.Close();
            lbConexion.Visible = true;
lbConexion2.Visible = false;
            lbConexion3.Text = "No conectado";
            Cerrar_Base();
            timer2.Enabled = false;
            timer3.Enabled = false;
            //Trama_PDU(txtCelular.Text, "El tiempo de muestreo ha concluido.");
if (notificarFinalizacion == true)
                for (int i = 0; i < TotalNumeros; i++)
Trama_PDU(Numeros_Celulares[i], "El tiempo ha concluido. Temperatura: " +
temperatura_recibida + " - H.Rel: " + humedad_relativa_recibida + " - Luminosidad: " +
luminosidad + " - Ferlu v1.0");
            MessageBox.Show("El tiempo de muestreo ha concluido", "Ferlu v1.0",
MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1);
            tiempo_timer = 0;
            mlbTiempoDetener.Text = "No establecido";
}
        else
        {
            if (segundo == 0)
            {
                if (minuto == 0)
{
                    if (hora != 0)
                    {
                        hora--;
                        minuto = 59;
                        segundo = 59;
                    }
```

```
                }
                else
                {
                    minuto--;
                    segundo = 59;
                }
            }
            else
            {
                segundo--;
            }
            mlbTiempoDetener.Text = "" + hora + ":" + minuto + ":" + segundo;
}
        }

        private void tiempoParaDetenerElMuestreoToolStripMenuItem_Click(object
sender, EventArgs e)
        {
        }

        private void establecerTiempoToolStripMenuItem_Click(object sender, EventArgs
e)
{
            tiempo_timer = 1;
            ConfigurarDetener detener = new ConfigurarDetener();
            detener.ShowDialog();
            hora = detener.regresa_horas();
            minuto = detener.regresa_minutos();
            segundo = detener.regesa_segundos();
            if (hora == 0 && minuto == 0 && segundo == 0)
            {
            }
            else
                mlbTiempoDetener.Text = "" + hora + ":" + minuto + ":" + segundo;
if (serialPort1.IsOpen == true)
            {
                timer2.Enabled = true;
            }
        }

        private void borrarTiempoToolStripMenuItem_Click(object sender, EventArgs e)
{
            tiempo_timer = 0;
            mlbTiempoDetener.Text = "No establecido";
if (serialPort1.IsOpen == true)
            {
```

```
            timer2.Enabled = false;
        }
    }

    private void Form1_FormClosed(object sender, FormClosedEventArgs e)
    {
    }

    private void conectarToolStripMenuItem1_Click(object sender, EventArgs e)
{
        if (cmbPuerto.Text.Equals(""))
            MessageBox.Show("Debe especificar un puerto de comunicaciones antes de
inicializar la comunicación.", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Information);
else
        {
            if (serialPort2.IsOpen == true)
MessageBox.Show("La conexión con el celular se encuentra actualmente abierta",
"Ferlu v1.0", MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1);
else
            {
                serialPort2.Open();
            }
        }
    }

    public String Borrar(String origen, int index, int count)
{
        String izquierda = "", derecha = "", final = "";

        /* Guardando parte izquierda */
for (int i = index + count; i < origen.Length; i++)
izquierda += origen.Substring(i, 1);

        /* Guardando parte derecha */
for (int j = 0; j < index; j++)
{
            derecha += origen.Substring(j, 1);
        }

        final = derecha + izquierda;

return final;
    }
```

```
        public String Insertar(String origen, int index, String value)
{
            String derecha = "", izquierda = "", resultado = "";

            /* Guardando parte derecha */
for (int i = index; i < origen.Length; i++)
derecha += origen.Substring(i, 1);

            /* Guardando parte izquierda */
for (int j = 0; j < index; j++)
izquierda += origen.Substring(j, 1);

            resultado = izquierda + value + derecha;

return resultado;
        }

        public String Binario_a_Decimal(String cadena)
        {
            int longitud = cadena.Length;
int numero = 0, contador = 0;
            String valor = "";

for (int i = longitud - 1; i >= 0; i--)
            {
                if (cadena.Substring(i, 1).Equals("1"))
numero += Convert.ToInt32(Math.Pow(2, contador));
                contador++;
            }

            valor = Convert.ToString(numero);

            return valor;
}

        public String Decimal_a_Hexadecimal(String cadena)
        {
            int number = int.Parse(cadena);
            string hex = number.ToString("x");

            if (hex.Length == 1)
                hex = "0" + hex;

            return hex;
        }
```

```
        public String Decimal_a_Hexadecimal2(String cadena)
        {
            int number = int.Parse(cadena);
            string hex = number.ToString("x");

            if (hex.Length == 1)
                hex = "F" + hex;

            return hex;
        }


        public String Trama_PDU(String Telefono, String Mensaje)
        {
            String[] arreglo = new String[2000];
            int longitud = 0;
int diferencia = 0, contador = 0;
            int i;
            int t;

            longitud = Mensaje.Length;

            /* Convirtiendo cada letra a binario */
for (i = 0; i < longitud; i++)
            {
                arreglo[i] = Convertir_Binario(Mensaje.Substring(i, 1));
            }


            for (i = 0; i < longitud-1; i++)
            {
                if (arreglo[i].Equals("") == false)
                {
                    if (arreglo[i + 1].Equals("") == false)
{
                        diferencia = 8 - arreglo[i].Length;
                        for (t = 0; t < diferencia; t++)
                        {
                            diferencia = 8 - arreglo[i].Length;
                            arreglo[i] = Insertar(arreglo[i], 0, arreglo[i + 1].Substring(arreglo[i +
1].Length - diferencia, diferencia));
                            arreglo[i + 1] = Borrar(arreglo[i + 1], arreglo[i + 1].Length - diferencia,
diferencia);
                        }
                        contador++;
                    }
```

```
            }
        }

        /* Convirtiendo de binario a hexadecimal */
String pdu = "";
        for (int m = 0; m < longitud; m++)
        {
            if (arreglo[m] != null && arreglo[m].Equals("") == false)
            {
                if(m == longitud-1)
pdu += Decimal_a_Hexadecimal2(Binario_a_Decimal(arreglo[m])).ToUpper();
                else
                    pdu +=
Decimal_a_Hexadecimal(Binario_a_Decimal(arreglo[m])).ToUpper();
}
        }



         int octetos = 13 + (pdu.Length/2);
         String octetosMensaje =
Convert.ToString(Mensaje.Length);//Convert.ToString(pdu.Length / 2);

octetosMensaje = Decimal_a_Hexadecimal(octetosMensaje);
         char ctrlz = (char)26;

         String auxiliar = "";

         for (int e = 0; e < Telefono.Length; e = e + 2)
            auxiliar += Telefono.Substring(e + 1, 1) + Telefono.Substring(e, 1);

Telefono = auxiliar;
        serialPort2.WriteLine("AT+CMGS=" +octetos + (char)13);
        System.Threading.Thread.Sleep(1200);
serialPort2.WriteLine("0011000A81" + Telefono + "0000AA" + octetosMensaje + pdu +
ctrlz);
         return pdu;
        /*
         char LN [] = "08";              Nº octetos que forman el campo dirección centro
                            servicio
        char TLL [] = "91";             //Nº con prefijo internacional
        char NCS [] = "254901001014F0";  //Nº centro servicio 5294100001410
        char TPDU [] = "11";             Tipo PDU (SMS para enviar. Formato Periodo
Vigencia
                            Relativo(1 octeto))
        char NR [] = "00";              //Número de referencia
```

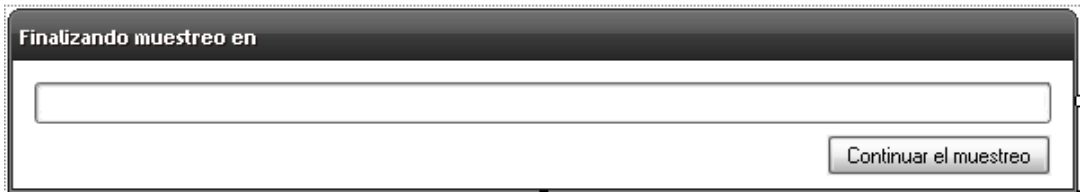
```
        char LT [] = "0A";              //Número de semioctetos (caracteres) del teléfono de
envío
        char NP [] = "A1";              //Nº con prefijo internacional
        char DD [] = "6951979648";      //Nº teléfono de envío 9615796984
        char PID [] = "00";             //Protocolo identificación
        char COD [] = "08";             //Codificación trama de datos a 16 bits.
        char PV [] = "AA";              //Periodo de vigencia del SMS
        char LD [] = "68";              /*Nº de octetos que forman los datos del mensaje
                                (no tomando en cuenta espacios en blanco)
*/
    }

    public string Convertir_Binario(string val)
    {
        string[] numerote = new string[val.Length];
        int contador = 0;
        int longitud = 0;
        int diferencia = 0;
        foreach (char c in val)
        {
            numerote[contador] = Convert.ToString(Convert.ToInt32(((int)c).ToString("X"),
16), 2);
            contador++;
        }
        string binarios = string.Join(" ", numerote);

        longitud = binarios.Length;


        if (longitud < 7)
        {
            diferencia = 7 - longitud;
            for (int i = 0; i < diferencia; i++)
            {
                binarios = "0" + binarios;
            }
        }

        return binarios;
    }

    private void enviarMensajeToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (serialPort2.IsOpen == true)
            serialPort2.Close();
        else
```

```
            MessageBox.Show("La conexión con el celular se encuentra actualmente
cerrada", "Ferlu v1.0", MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1);
        }

        private void establecerNúmerosDeEnvíoToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            if (serialPort2.IsOpen == false)
MessageBox.Show("El puerto de comunicaciones se encuentra cerrado", "Ferlu v1.0",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            else
            {
                EstablecerNumeros establecer = new EstablecerNumeros();
                establecer.ShowDialog();
                if (establecer.dialogResult() == true)
                {
                    Numeros_Celulares = establecer.Numeros_Celulares();
                    TotalNumeros = establecer.regresaTotalNumeros();
                    notificarFinalizacion = establecer.Notificar_Finalizacion_De_Muestras();
                    notificarMuestras = establecer.Notificar_Muestras();
                    hora2 = establecer.regresaHora();
                    minuto2 = establecer.regresaMinuto();
                    segundo2 = establecer.regresaSegundo();
}
            }
        }

        private void cmbPuerto_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (serialPort2.IsOpen == false)
                serialPort2.PortName = cmbPuerto.Text;
else
                MessageBox.Show("Debe cerrar el puerto de comunicaciones antes de
cambiar esta propiedad", "Ferlu v1.0", MessageBoxButtons.OK,
MessageBoxIcon.Information);
}

        private void actualizarListaToolStripMenuItem1_Click(object sender, EventArgs e)
        {
        }

        private void llamadaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //serialPort2.WriteLine("ATD 9611791160;\r\n");
//Trama_PDU("9611791160", "Hola MUNDO, esta es una prueba de mensaje.");
```

```
        //serialPort2.WriteLine(Trama_PDU("9611791160","Hola, este es un mensaje de
prueba. La muestra finalizo"));
        if (serialPort2.IsOpen == false)
            MessageBox.Show("El puerto de comunicaciones se encuentra cerrado",
"Ferlu v1.0", MessageBoxButtons.OK, MessageBoxIcon.Information);
else if (txtCelular.Text.Equals("") || txtCelular.Text.Length != 10)
            MessageBox.Show("Por favor introduzca un celular válido", "Ferlu v1.0",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        else
            serialPort2.WriteLine("ATD "+ txtCelular.Text +";\r\n");
    }

    private void timer3_Tick(object sender, EventArgs e)
{
        if (segundo22 == 59)
        {
            segundo22 = 0;
            if (minuto22 == 59)
            {
                minuto22 = 0;
                if (hora22 == 23)
                    hora22 = 0;
                else
                    hora22++;
            }
            else
                minuto22++;
        }
        else
            segundo22++;

        if (hora22 == hora2 && minuto22 == minuto2 && segundo22 == segundo2)
        {
            hora22 = 0;
            minuto22 = 0;
            segundo22 = 0;
            //MessageBox.Show("Se alcanzó el tiempo para la notificación");
for (int i = 0; i < TotalNumeros; i++)
{
                Trama_PDU(Numeros_Celulares[i], "Temperatura: " +
temperatura_recibida + " - H.Rel: " + humedad_relativa_recibida + " - Luminosidad: " +
luminosidad + " - Ferlu v1.0");
System.Threading.Thread.Sleep(5000);
            }
        }
    }
```

```
        private void cmbPuerto_Click(object sender, EventArgs e)
        {

        }

        private void graToolStripMenuItem_Click(object sender, EventArgs e)
        {

        }

        private void graficarToolStripMenuItem1_Click(object sender, EventArgs e)
        {

        }

        private void graficarToolStripMenuItem1_Click_1(object sender, EventArgs e)
        {

        }

        private void graficarToolStripMenuItem1_Click_2(object sender, EventArgs e)
{
            Graficar g = new Graficar();
            g.ShowDialog();
}
    }
}
```

Módulo: ParaDetener.cs



Este módulo permite al usuario continuar con las mediciones. Aparece 1 minuto antes de detener las mediciones cuando se ha configurado un tiempo determinado.

Código:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Control_III
{
   public partial class ParaDetener : Form
   {
      public ParaDetener()
      {
         InitializeComponent();
      }

      int detiene = 0;
      int segundos = 59;

      public int detener()
      {
         return detiene;
      }

      private void button1_Click(object sender, EventArgs e)
      {
         detiene = 1;
         timer1.Enabled = false;
         this.Close();
      }

      private void timer1_Tick(object sender, EventArgs e)
{
         if (segundos != 0)
         {
            segundos--;
            this.Text = "Finalizando muestreo en " + segundos + " segundos";
progressBar1.Value = Convert.ToInt32((segundos * 100) / 59);
         }
         else
         {
            timer1.Enabled = false;
this.Close();
         }
      }
   }
}
```

Módulo: Presentación.cs

Este módulo es informativo. En él se presentan los datos del desarrollador del programa.

Código:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Control_III
{
   public partial class Presentación : Form
   {
      public Presentación()
      {
         InitializeComponent();
      }

      private void Presentación_KeyPress(object sender, KeyPressEventArgs e)
      {
         if (e.KeyChar == 13 || e.KeyChar == 27)
            this.Close();
      }

      private void Presentación_Load(object sender, EventArgs e)
      {
```

```
        }

        private void pictureBox1_Click(object sender, EventArgs e)
{

        }
    }
}
```

Módulo: Program.cs (Principal)

Esta es una clase que manda a llamar el formulario principal (Form1.cs) para su ejecución.

Código:

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace Control_III
{
    static class Program
{
        /// <summary>
        /// Punto de entrada principal para la aplicación.
/// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
Application.Run(new Form1());
        }
    }
}
```

Función implementada en C# para generar la trama PDU:

```
public String Trama_PDU(String Telefono, String Mensaje)
        {
            String[] arreglo = new String[2000];
            int longitud = 0;
int diferencia = 0, contador = 0;
            int i;
            int t;
```

```
        longitud = Mensaje.Length;

        /* Convirtiendo cada letra a binario */
for (i = 0; i < longitud; i++)
        {
            arreglo[i] = Convertir_Binario(Mensaje.Substring(i, 1));
        }


        for (i = 0; i < longitud-1; i++)
        {
            if (arreglo[i].Equals("") == false)
            {
                if (arreglo[i + 1].Equals("") == false)
{
                    diferencia = 8 - arreglo[i].Length;
                    for (t = 0; t < diferencia; t++)
                    {
                        diferencia = 8 - arreglo[i].Length;
                        arreglo[i] = Insertar(arreglo[i], 0, arreglo[i + 1].Substring(arreglo[i +
1].Length - diferencia, diferencia));
                        arreglo[i + 1] = Borrar(arreglo[i + 1], arreglo[i + 1].Length - diferencia,
diferencia);
                    }
                    contador++;
                }
            }
        }

        /* Convirtiendo de binario a hexadecimal */
String pdu = "";
        for (int m = 0; m < longitud; m++)
        {
            if (arreglo[m] != null && arreglo[m].Equals("") == false)
            {
                if(m == longitud-1)
pdu += Decimal_a_Hexadecimal2(Binario_a_Decimal(arreglo[m])).ToUpper();
                else
                    pdu +=
Decimal_a_Hexadecimal(Binario_a_Decimal(arreglo[m])).ToUpper();
            }
        }


         int octetos = 13 + (pdu.Length/2);
```

```
            String octetosMensaje = Convert.ToString(Mensaje.Length);

            octetosMensaje = Decimal_a_Hexadecimal(octetosMensaje);
            char ctrlz = (char)26;

            String auxiliar = "";

            for (int e = 0; e < Telefono.Length; e = e + 2)
               auxiliar += Telefono.Substring(e + 1, 1) + Telefono.Substring(e, 1);

Telefono = auxiliar;
            serialPort2.WriteLine("AT+CMGS=" +octetos + (char)13);
            System.Threading.Thread.Sleep(1200);
serialPort2.WriteLine("0011000A81" + Telefono + "0000AA" + octetosMensaje + pdu +
ctrlz);
return pdu;
        }
```

DIAGRAMAS

OPT101

Este Ruteado corresponde al conexionado del sensor de luz OPT101.

Pistas

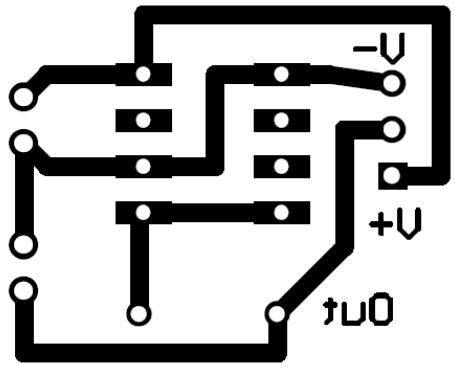
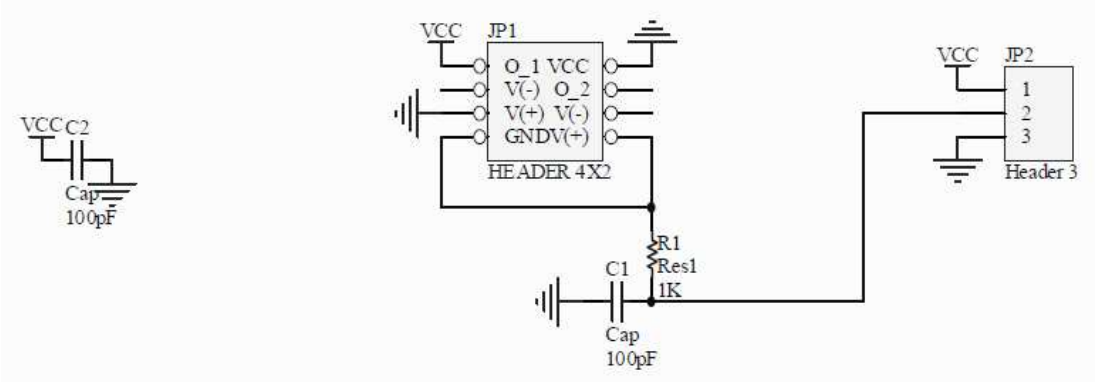


Diagrama esquemático



Circuito principal.

El ruteado siguiente corresponde a la interfaz principal de adquisición de datos.

Pistas

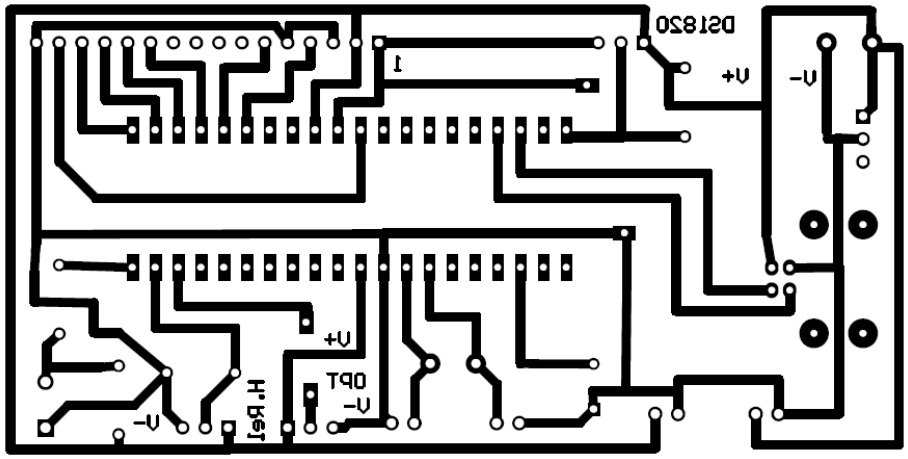
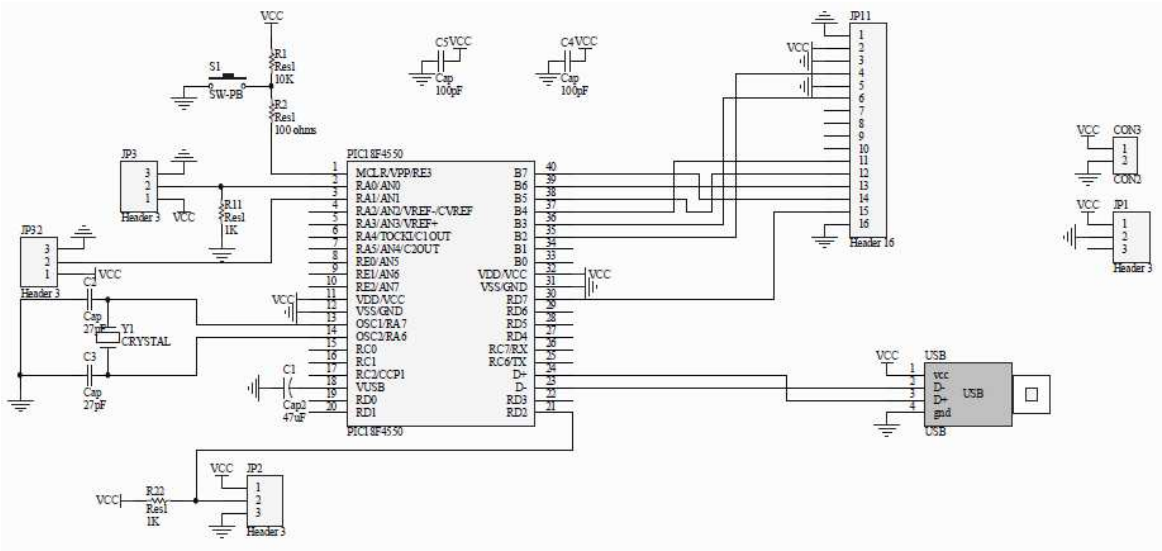
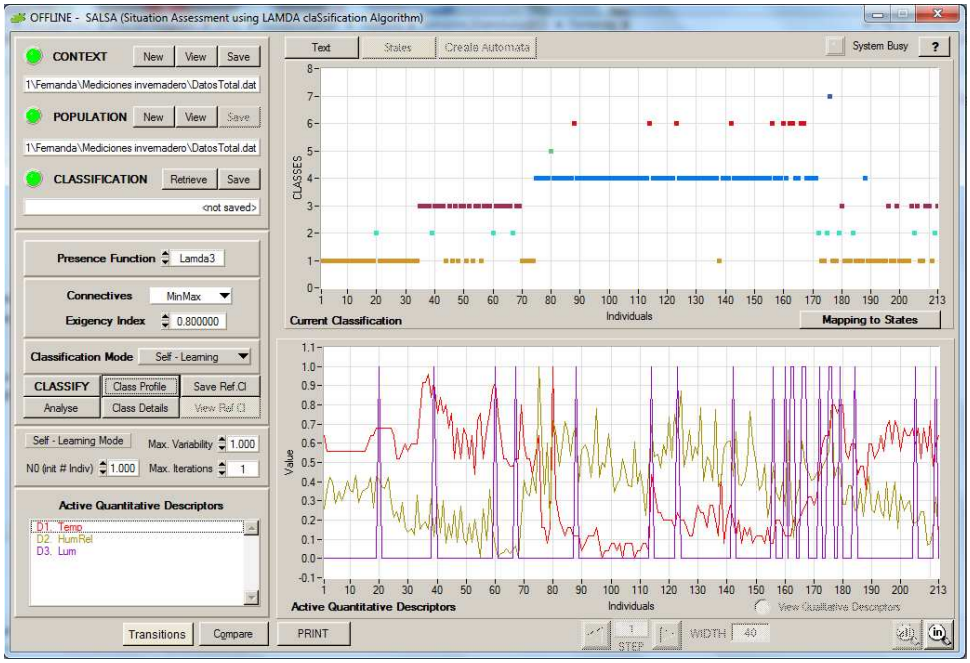


Diagrama esquemático

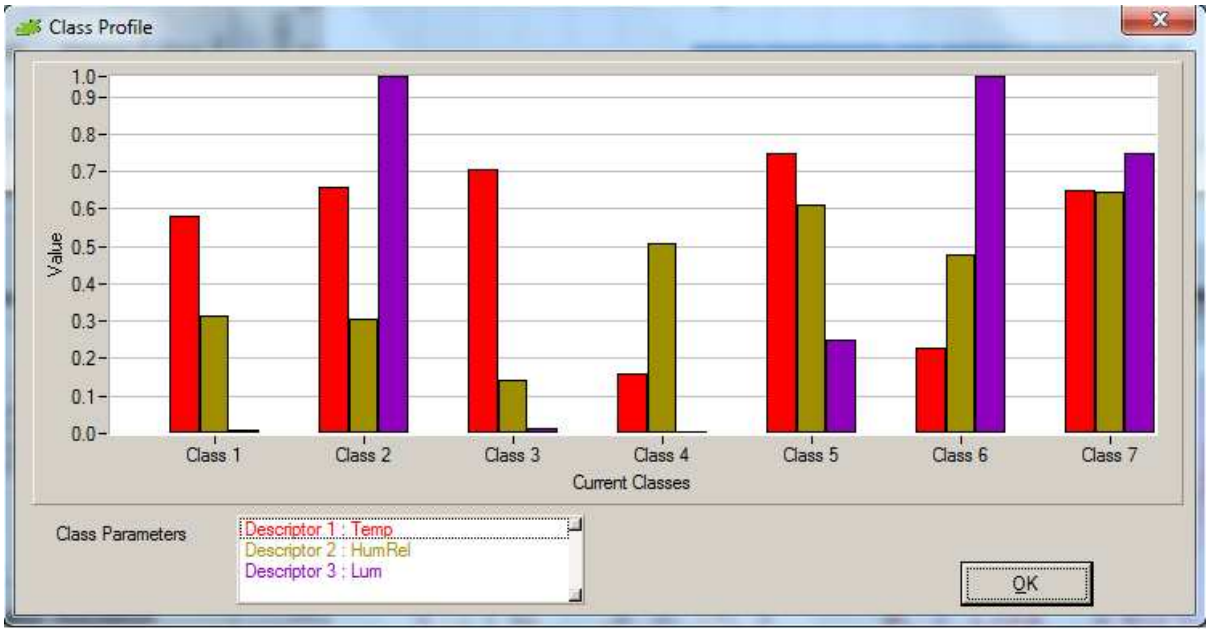


Análisis de Resultados

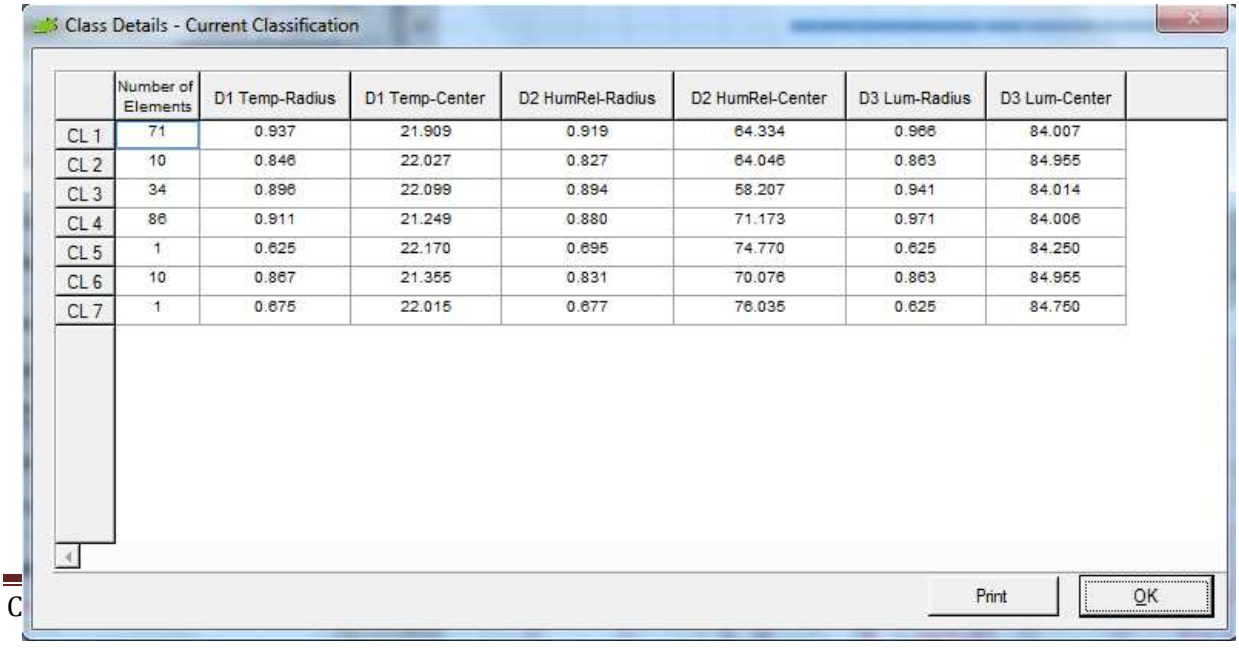
.



Eligiendo la función de presencia lambda 3 junto con una conectividad mínimo máximo y un índice de exigencia de .8 el software nos clasifica los 213 datos en siete clases de acuerdo a su funcionamiento.

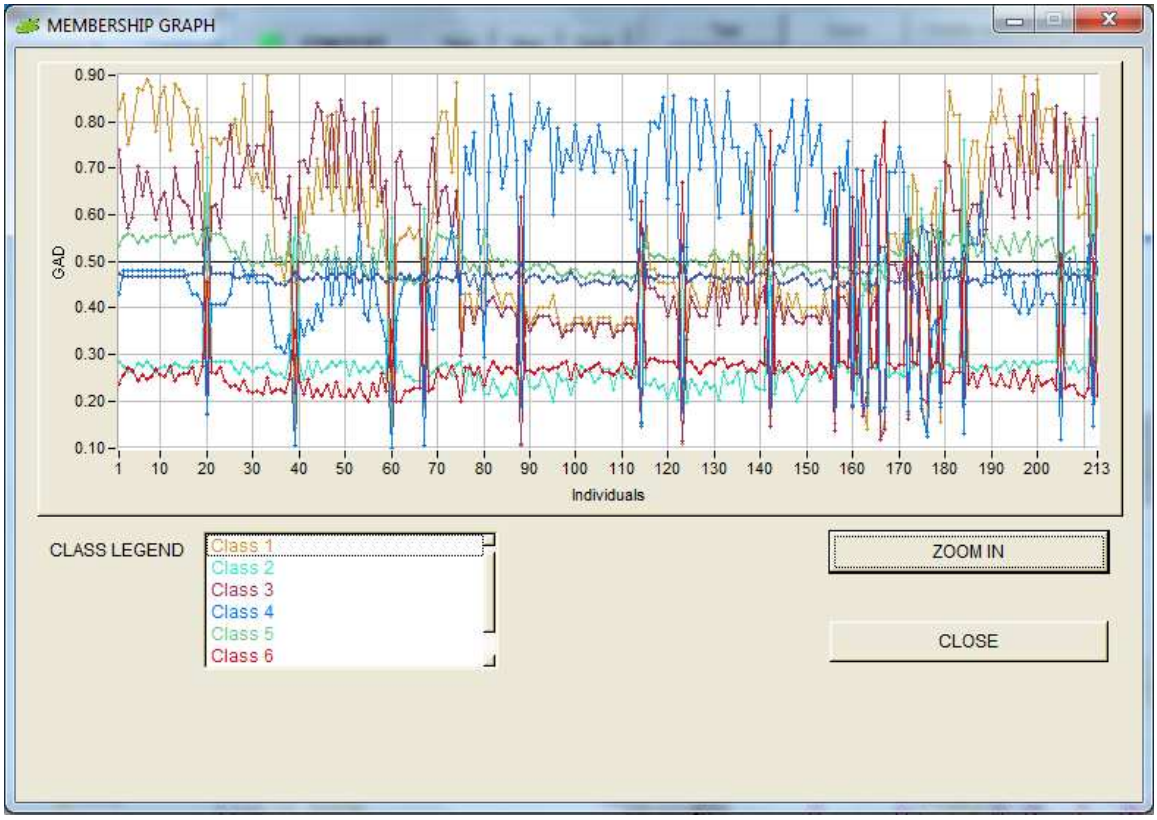


La imagen anterior nos muestra las siete clases junto al valor de las tres variables. Eso nos ilustra los valores de cada uno de estos y su pertenencia. La siguiente imagen nos muestra una tabla con el valor numérico que es representado en la gráfica anterior.



	Number of Elements	D1 Temp-Radius	D1 Temp-Center	D2 HumRel-Radius	D2 HumRel-Center	D3 Lum-Radius	D3 Lum-Center
CL 1	71	0.937	21.909	0.919	64.334	0.966	84.007
CL 2	10	0.846	22.027	0.827	64.046	0.863	84.955
CL 3	34	0.896	22.099	0.894	58.207	0.941	84.014
CL 4	86	0.911	21.249	0.880	71.173	0.971	84.006
CL 5	1	0.625	22.170	0.695	74.770	0.625	84.250
CL 6	10	0.867	21.355	0.831	70.076	0.863	84.955
CL 7	1	0.675	22.015	0.677	76.035	0.625	84.750

C



Analizando los datos con el software obtenemos la gráfica anterior donde muestra el grado de pertenencia de cada dato con respecto a las clases con motivo de implementar un algoritmo de carácter difuso (sistema diagnóstico o control difuso).

El desarrollo del modelo experimental permite determinar un punto de control bajo las condiciones reales de operación y como resultado el modelo encontrado tiene un desempeño cercano a los valores teóricos. La confrontación a la realidad permite un seguimiento de las señales de referencia dadas por el usuario en este caso las condiciones son determinadas por las característica de cultivo de frijol y agave; además muestra la robustez a las perturbaciones del ambiente externo, como los cambios de temperatura.

CONCLUSIÓN.

En base al resultado del análisis experimental se encontró un punto de operación para el desarrollo del control difuso y se puede observar que el comportamiento es no lineal. Para la operación con el modelo a escala se puede decir que funciona de acuerdo a lo predicho en el modelo matemático. En conclusión se puede decir que se logró el objetivo; se pudo sensar las tres variables y registrarlas en base de datos para su posterior tratamiento. El sistema de control es de bajo costo y sencillo de manejar, por lo que se recomienda su empleo en otros invernaderos.

Gracias a que se implementó un software se pudo hacer el registro de los datos y se pudo apreciar las lecturas en la computadora para que se le facilitara al usuario.

Referencia Bibliográfica y virtual

[1] FLOYD, Thomas (1999). Dispositivos Electrónicos. México, D.F.: Noriega Editores.

[2] MALONEY, Timthy J. (1983). Electrónica Industrial Dispositivos y Sistemas. Madrid, España.: Madrid Editores.

[3] BOYLESTAD Robert, NASHELSKY Louis (1997). Electrónica: Teoría de Circuitos. México, D.F.: Pearson Educación.

[4]RUIZ VASALLO, Francisco (1997). Componentes Electrónicos. Barcelona: CEAC.

[5]MAXIM, Dallas semiconductor (1997). HIH4030. Consultado en 05/11/2011 en http://www.sparkfun.com/datasheets/Sensors/Weather/SEN-09569-HIH-4030-datasheet.pdf.

[6]MAXIM, Dallas semiconductor (2001). DS1820. Consultado en 05/19/2011 en http://www.datasheetcatalog.com/datasheets_pdf/D/S/1/8/DS1820.shtml.

[7] MAXIM, Dallas semiconductor (1995). OPT101. Consultado en 05/26/2011 en http://www.datasheetcatalog.com/datasheets_pdf/O/P/T/1/OPT101.shtml.

[8]MAXIM, Dallas semiconductor (2003). Envío de Mensajes SMS en Formato PDU. Consultado en 06/01/2011 en http://www.zonabot.com/electronica/2-comunicaciones/35-envio-de-mensajes-sms-en-formato-pdu-desde-pc.html.