ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE ESTACIÓN DE MEDICIÓN DE PARÁMETROS METEOROLÓGICOS PARA SU INCORPORACIÓN EN UNA RED EXTENDIDA DE MONITORIZACIÓN AMBIENTAL

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE FÍSICO

VÁSCONEZ ALBÁN FREDDY GUSTAVO
freddy.vasconez@epn.edu.ec

DIRECTOR: CÉSAR COSTA VERA, Ph.D.
cesar.costa@epn.edu.ec

Quito, FEBRERO 2009

DECLARACIÓN

Yo, Freddy Gustavo Vásconez Albán, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido presentado previamente para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Freddy Gustavo Vásconez Albán

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Freddy Gustavo Vásconez Albán, bajo mi supervisión.

_____

César Costa Vera, Ph.D.
DIRECTOR DEL PROYECTO

CONTENIDO

RESUMEN

En este trabajo se describe un sistema de recolección y distribución de información multiparamétrica, dirigido al estudio climático, por cuanto se basa en parámetros meteorológicos.

El sistema se compone de una parte tangible (hardware) y una parte informática (software). El hardware corresponde a un nodo de red recolector de datos, y un nodo de recepción. El software constituye un componente de distribución.

El nodo recolector está compuesto por seis sensores agregados a un módulo con capacidad de comunicación inalámbrica. Este nodo recoge información de temperatura, humedad relativa, luminosidad, y cantidad de lluvia.

El segundo nodo, de recepción, sirve como puente para que los datos recolectados puedan distribuirse a través de la Internet. El componente de distribución tiene como tarea poner la información obtenida del nodo recolector a disposición de quien tuviese interés en utilizarla, en cualquier lugar que tenga acceso a la Internet.

Se ha dividido este documento en cuatro capítulos. El primer capítulo contiene una introducción a la medición remota de parámetros físicos. Inicia con la conceptualización de los parámetros físicos en general y luego hace hincapié en aquellos que intervienen en el tema de estudio y su papel dentro del clima. Además se describen las técnicas utilizadas para el manejo de la información que se ha de obtener acerca de los parámetros de interés.

En el segundo capítulo se encuentra la descripción del experimento en sí. En este capítulo se detalla el montaje de la parte física (hardware) así como el de la parte informática (software).

El tercer capítulo contiene los resultados obtenidos y una discusión sobre éstos. Empieza con una muestra de los datos que se han recogido y un breve análisis

sobre esta muestra; y luego se encuentra una comparación entre un sistema electrónico básico y el sistema construido para este trabajo. Por último, se presentan las conclusiones del trabajo en el cuarto capítulo.

En este trabajo se demuestra que los objetivos planteados se pueden cumplir, obteniéndose una red de medición basada en un nodo de recolección de información, un nodo de almacenamiento y una plataforma de distribución de datos a través de Internet.

PRESENTACIÓN

La medición de parámetros meteorológicos es de fundamental importancia para el estudio del medio ambiente y los procesos que ahí ocurren. La ciudad de Quito presenta un estado climático que cambia notablemente con la zona (microclimas), debido a la extensión de la ciudad así como al relieve de la misma; convirtiéndose en un laboratorio extendido para dicha actividad. Sin embargo, esta misma situación presenta retos importantes para la medición de los parámetros mencionados. Es necesario contar con una herramienta que sea robusta y extensible, de manera que pueda darnos una buena idea de la variación espacial y temporal de los parámetros meteorológicos.

Actualmente, la disponibilidad de dispositivos de medición con capacidad de comunicación inalámbrica y de bajo consumo de energía (por ejemplo, dispositivos TELOS, del fabricante *moteiv*) provee posibilidades muy interesantes para desarrollar tecnologías de medición. Estas características proporcionan una ventaja en cuanto al desarrollo de redes de monitoreo multiparamétrico. Además, la Internet presenta la facilidad de intercambiar información a distancias enormes en tiempos muy cortos, siendo así uno de los medios de comunicación más utilizado actualmente.

En este proyecto nos hemos propuesto crear una red, a la que se llamará Red Climatológica Quito, que permita la medición multiparámetro en una zona relativamente extendida, que explote las características de comunicación inalámbrica de ciertos dispositivos, y que facilite la disponibilidad de información en tiempo real en cualquier lugar del Mundo que tenga una conexión a Internet.

Los recursos descritos pretenden ser utilizados para el desarrollo de una herramienta que cubra las necesidades detalladas previamente. Para completar la idea, diremos que el objetivo de este trabajo es el desarrollo de un sistema de medición que facilite la disponibilidad de resultados a través de la Internet.

Para llegar al objetivo es necesario analizarlo de manera minuciosa, pudiéndose desglosar los siguientes objetivos específicos:

a) Estudiar las características de un dispositivo TELOS en cuanto al acoplamiento con sensores, comunicación y su lenguaje de programación. Este dispositivo servirá con referente para el desarrollo futuro de la red.
b) Desarrollar un sistema robusto y confiable que funcione como nodo inalámbrico en una red de medición.
c) Desarrollar hardware, software y los procesos adecuados para la recolección, almacenamiento, distribución y análisis de datos.
d) Crear una página Web en la que se publiquen los datos obtenidos de modo que estén disponibles en cualquier lugar del Mundo que tenga acceso a la Internet.
e) Comparar las ventajas de mantener una red inalámbrica de medición de parámetros meteorológicos en base a dispositivos TELOS en relación con una red tradicional.

Este trabajo comprende la creación de una Red de Medición de parámetros meteorológicos, que consta de tres partes: un nodo recolector, un nodo receptor, y un sistema distribuidor. La construcción del nodo recolector significó el acople de dos sensores extras a un dispositivo TELOS y la programación de éste para que pueda recoger datos y comunicarse con otro nodo. El nodo receptor es un dispositivo TELOS programado con una aplicación llamada *TOSBase*, que viene como ejemplo en la distribución de TinyOS. Además, el nodo receptor comprende un programa, que ingresa la información de el/los nodos recolectores en una base de datos; ambos, programa y base de datos, creados como parte de este trabajo.

El sistema de distribución es un portal Web que consta de ocho páginas. Para la programación de estas páginas se ha utilizado lenguaje Java conjuntamente con HTML. Además se ha utilizado un programa creado por el ingeniero en sistemas Pablo Marcillo Lara <pmarcillo@igepn.edu.ec>. Este programa se llama *Componente.java*, y fue desarrollado para realizar gráficas en aplicaciones sísmicas del Instituto Geofísico.

1. INTRODUCCIÓN A LA MEDICIÓN REMOTA DE PARÁMETROS FÍSICOS

1.1. PARÁMETROS FÍSICOS

El principal objetivo de la ciencia es darnos una comprensión del mundo físico y biológico que nos rodea mediante el descubrimiento de principios generales que muestran las relaciones entre hechos observados. Aunque los descubrimientos pueden resultar de la sola deducción, la experimentación actúa como último juez de la validez de las teorías [1].

La medición puede ser definida como la determinación de números que representan propiedades. Estas propiedades están ligadas a entidades determinadas, a las que llamamos *magnitudes*. Dicho de una manera más sencilla, las magnitudes físicas son todas aquellas entidades en el universo a las que podemos asignar un valor numérico; a diferencia, por ejemplo, de ciertas características humanas como la amistad o el patriotismo, que no son susceptibles de medición cuantitativa y por lo tanto no están sometidas a la investigación científica, aunque tengan gran importancia para todos [1].

Una cualidad sobresaliente de la física es la exitosa armonía entre la naturaleza de sus conceptos y sus definiciones. Al eliminar las ambigüedades en las definiciones de los conceptos se evitan discusiones y se promueve el desarrollo científico [1]. Un buen ejemplo de esto se muestra claramente con el Sistema Internacional de Unidades[♣] (SI), construido en base a sistemas métricos más antiguos y de tal manera que se use una sola unidad para cada magnitud [2].

De acuerdo con el SI existen solamente siete magnitudes fundamentales con sus respectivas unidades de medida, en base a las que se pueden expresar las demás. Las magnitudes fundamentales son: longitud, tiempo, masa, intensidad de corriente eléctrica, intensidad luminosa, temperatura y cantidad de sustancia; con

---

♣ Adoptado en el Ecuador en Enero de 1974 [2].

sus respectivas unidades: metro (m), segundo (s), kilogramo (kg), amperio (A), candela (cd), kelvin (K) y mol (mol).

Básicamente con estas siete magnitudes se caracteriza el mundo que nos rodea para, de esta manera, entender mejor los fenómenos que se suscitan en nuestro diario vivir.

1.2. PARÁMETROS FÍSICOS METEOROLÓGICOS

Una gran cantidad de fenómenos que suceden a nuestro alrededor están relacionados entre sí formando lo que llamamos clima. El objetivo de muchos estudios, desde tiempos inmemorables, ha sido la predicción de cambios climáticos a corto, mediano y largo plazo; llevando al hombre a fortalecer y afinar ciertas técnicas que le permitan obtener información de los parámetros climatológicos y conjeturar al respecto de las causas de su variación.

Para facilitar su comprensión se estudia por separado cada parámetro que describa el estado del clima en una condición determinada para luego encontrar las relaciones existentes entre ellos y poder plantear modelos que describan el comportamiento del sistema como tal.

Son muchos los factores que intervienen en los cambios climáticos, entre ellos podemos citar la topografía de la zona en estudio, la época del año, la influencia de volcanes, la influencia de corrientes marinas… Todos estos factores son responsables de la variación de parámetros que describen el clima. Entre estos parámetros, llamados meteorológicos, se puede citar a la temperatura del aire, la temperatura del suelo, la humedad relativa del aire, la presión atmosférica, la cantidad de lluvia, la nubosidad, la cantidad de luz.

En el presente trabajo se considerarán, y en adelante debe entenderse como parámetros meteorológicos o climatológicos a la temperatura del aire, la humedad relativa, la cantidad de luz y la pluviosidad (cantidad de lluvia).

Uno de los parámetros en el que se ha puesto más énfasis es la temperatura. En general se define la temperatura como el promedio de energía cinética de las partículas que conforman un sistema. Algunos autores prefieren definirla como la condición que determina la dirección del flujo resultante de calor entre dos cuerpos; proponiendo esta definición como un paso previo al equilibrio termodinámico que constituye la base del funcionamiento de los termómetros [3].

Quizás el hecho de su "popularidad" se deba a la facilidad en la medición de este parámetro y a lo familiar que nos parecen sus efectos. Nuestro sentido del tacto constituye, en primera instancia, un medidor de temperatura, brindándole a éste parámetro una importancia significativa sobre otras variables menos perceptibles a nuestros sentidos. Además, cualquier propiedad física de una sustancia que sea función de la temperatura puede ser utilizada como base de un termómetro, lo que se ve reflejado en la amplia variedad de tipos de termómetros, cada uno con su correspondiente propósito.

Los procesos fisiológicos en los organismos vegetales, tales como; respiración, fotosíntesis, asimilación y transpiración, transcurren solamente a determinadas temperaturas; los valores óptimos y los valores extremos de las temperaturas son diferentes para las plantas de distintas especies e incluso para diversos periodos de su vida, por lo que la temperatura del aire tiene una gran importancia en la vida de las plantas [3].

Es necesario, además, tomar en cuenta que todas las muestras de aire tomadas cerca de la superficie de la Tierra contienen vapor de agua. En algunos casos, la cantidad de vapor en la muestra es muy pequeña y se debe aplicar técnicas complejas para determinarla. La relación entre la masa de vapor de agua contenido en una unidad de volumen del aire y la masa de vapor de agua que sería necesario para saturar este volumen a la misma temperatura, es lo que llamamos humedad relativa [3].

La cantidad máxima de vapor que puede contener una masa de aire depende de las condiciones de temperatura y presión. Si la temperatura desciende, el punto

de saturación del aire también desciende; de este modo, suponiendo que la presión se mantenga constante, llegará un punto en el que la masa de aire se sature con la misma cantidad de vapor con la que antes no estaba saturada y el vapor de agua empieza a condensarse. La temperatura a la que esto sucede se denomina *punto de rocío* [3].

Para la medición de la humedad relativa existen varios métodos. Uno de ellos es medir la temperatura con dos termómetros, uno de ellos envuelto en una mecha empapada: el agua se evapora de la mecha tomando calor del termómetro por lo que marcará una temperatura menor que la del termómetro seco. La diferencia de temperaturas será mayor mientras la humedad relativa sea menor [4].

Los flujos de radiación hacia y desde la superficie terrestre son de vital importancia en varios procesos tanto biológicos como industriales y por tanto, en la ciencia. El gráfico de la Fig 1.1 da una idea de la distribución espectral (línea 1) de la radiación solar difusa. En la misma figura se compara la intensidad de la radiación solar directa al nivel del mar y a 35° de altura del Sol sobre el horizonte (línea 2). La línea 3 muestra la intensidad de la radiación de la superficie terrestre de onda larga, suponiendo que la temperatura de la superficie de la Tierra es de 20 °C.

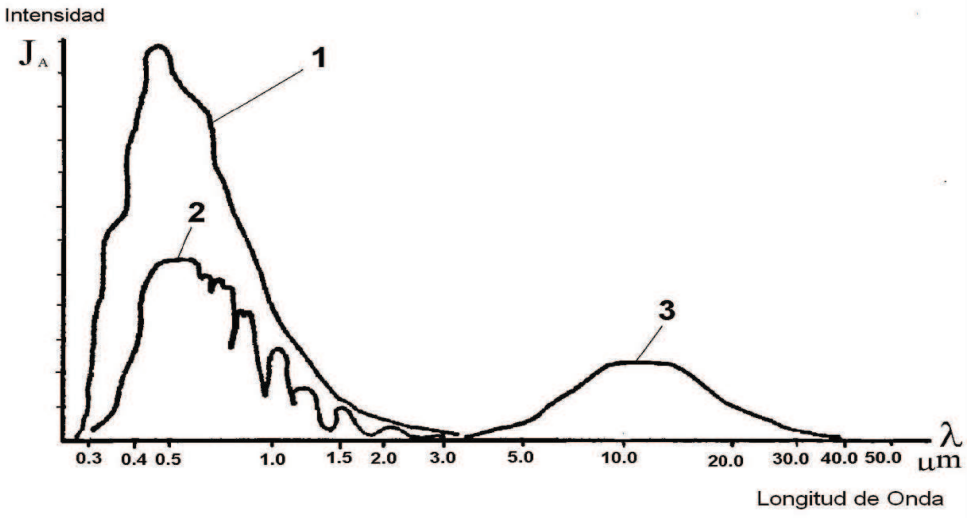


Fig. 1.1 Distribución espectral de la radiación extraterrestre. 1- Radiación solar difusa. 2- Radiación solar directa al nivel del mar, con el sol a 35° de elevación sobre el horizonte. 3- Radiación de la superficie terrestre, suponiendo una temperatura de 20°C. Referencia [3].

Alrededor del 99 por ciento de la radiación emitida por el Sol, (con un temperatura superficial de 5800 °K), se ubica en la gama que va de 0.150 a 4000 nm. Casi el 45% de la emisión solar total se produce en la banda visible y otro tanto, en la infrarroja (700 – $3.5*10^5$ nm [4]). Alrededor del 9% se produce en la banda ultravioleta (5 - 400 nm [4]) [3].

La luz visible ocupa el espectro entre 400 y 700 nm [4]. La radiación correspondiente a este rango es la única fotosintéticamente activa, lo que quiere decir que es la que absorben los cloroplastos de las plantas para el proceso de fotosíntesis. Los encargados de darle color a las plantas son los pigmentos, que absorben la radiación solar para que la fotosíntesis se pueda producir. Existen varios tipos de pigmentos, entre ellos está la clorofila que absorbe la luz en el rango visible y refleja la correspondiente al color verde (490 - 560 nm) dándole esa coloración a las hojas de las plantas. En la Fig. 1.2 se muestra la radiación absorbida por diferentes clases de pigmentos [6].

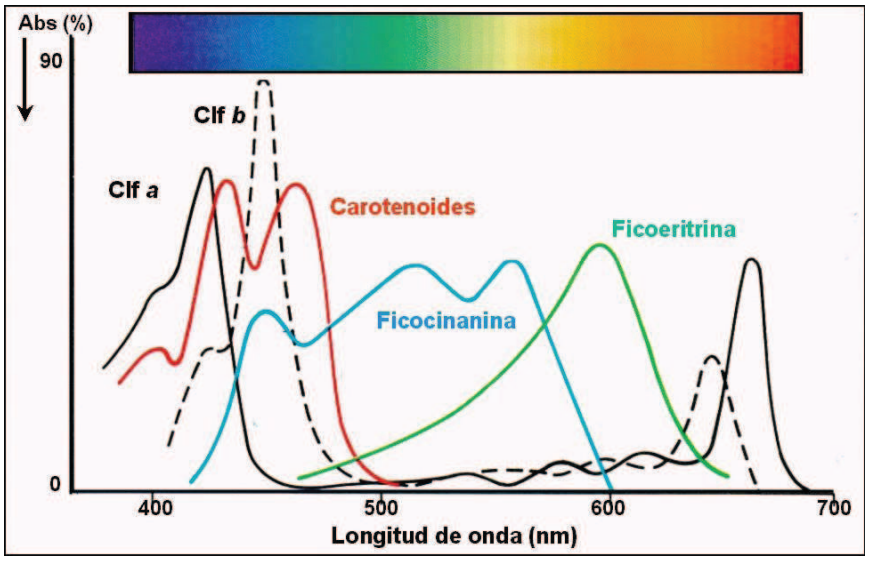


Fig. 1.2 Espectro de absorción de algunos pigmentos fotosintéticos. Tomado de [6].

Otro proceso natural imprescindible para la vida sobre la Tierra corresponde a las precipitaciones. Para que las gotas de agua que forman las nubes lleguen a precipitarse se requiere que éstas aumenten de tamaño y que el aire situado

debajo de las nubes no sea demasiado cálido, ni muy seco, ya que de no ser así, las gotas de agua que inician la caída pueden volver a evaporarse [3].

Se denomina Precipitación o Lluvia al agua que cae en forma líquida, sólida, o líquida y sólida desde las nubes hasta la superficie de la Tierra. El objetivo que se tiene al medir la precipitación es el de obtener tanta información de ésta como sea posible, acerca de la cantidad y distribución, en el tiempo y el espacio. La forma más simple y usual de realizar la medición es usando un medidor de abertura horizontal, circular, de diámetro conocido; se colecta y mide, a intervalos regulares, la cantidad que cae por unidad de área de la abertura del medidor y es igual a la cantidad de precipitación por unidad de área que cae en los alrededores [3].

1.3. TÉCNICAS DE MEDICIÓN

1.3.1. RECOLECCIÓN DE DATOS

Para enfatizar un poco en la medición de parámetros físicos meteorológicos, mencionaremos que el estudio instrumental de un fenómeno natural pasa necesariamente por disponer de una serie de instrumentos que nos aporten datos. La utilización correcta de estos datos requiere conocer cómo funcionan los instrumentos y hasta qué punto la información que podemos obtener con ellos refleja el fenómeno que estamos estudiando. Es muy frecuente que se consideren los instrumentos como máquinas perfectas, que nos dicen aquello que en nuestra imaginación queremos oír. En general debemos considerar el instrumento, por bueno que éste sea, como una caja que nos da una pobre información del fenómeno que está midiendo; y que en cualquier momento ésta información puede ser falsa [7].

Sin embargo, los instrumentos resultan una parte importantísima al momento de estudiar fenómenos naturales, por el simple hecho de que constituyen una ventana que nos conecta con el mundo físico y nos permiten obtener información de éste. Al considerar un instrumento cualquiera siempre se debe tener presente

cómo se establece la relación entre el fenómeno físico y la información que el instrumento nos proporciona. En un instrumento existen siempre dos partes: el sensor o captador y la salida de información [7].

Al decir que vamos a obtener información viene inmediatamente la interrogante sobre el tipo de información: ¿será analógica o digital? Para entender la diferencia entre estos dos términos es conveniente echar un vistazo a los conceptos mismos de sistemas analógicos y digitales. Los sistemas analógicos consisten sistemas continuos; su nombre hace referencia a tiempo pasado en el que para resolver problemas era necesario crear un sistema análogo que responda a las mismas ecuaciones y que se pudiera reproducir en un laboratorio. Los sistemas digitales, en cambio, consisten en sistemas discretos que permanecen íntimamente relacionados con los números, y los números surgieron del hecho que tenemos dedos; de este modo los sistemas digitales están en la base misma de nuestra cultura [7].

Es común que se piense que todos los sistemas digitales son mejores que los analógicos, cuando en muchas ocasiones la solución analógica a un problema resulta mucho más sencilla y barata [7]. Por otro lado, el hecho de contar con información digital nos brinda una ventaja enorme al poder contar con herramientas muy poderosas como son los sistemas computarizados. Es por esto que se tiende a digitalizar las señales analógicas.

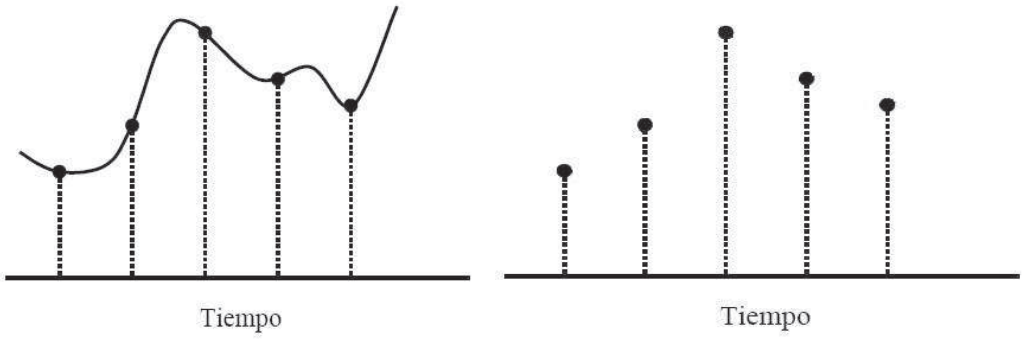


Fig. 1.3 Muestreo. A la izquierda la señal analógica completa y a la derecha la información que se obtiene. Adaptado de la referencia [7].

Normalmente se recoge la información de un parámetro cada cierto tiempo, a esto se llama *muestreo*. Pero esa información es parcial, debido a que no conocemos la variación del parámetro entre medida y medida (Fig. 1.3) [7].

Es importante entonces el concepto de *frecuencia de muestreo*, que resulta el inverso del tiempo transcurrido entre una medida y su consecutiva; se expresa en Hz. Es preciso utilizar una frecuencia de muestreo que asegure la completa reconstrucción de la señal original. Para fijar la frecuencia de muestreo (F_m) debemos recurrir al teorema de Nyquist, que establece:

$$F_m \geq 2 \cdot F_{máx} \tag{1}$$

Donde $F_{máx}$ la frecuencia de la componente espectral de mayor frecuencia de la señal a muestrear [8]. La Fig. 1.4 ilustra un ejemplo de una señal sinusoidal muestreada con una frecuencia que no cumple el teorema de Nyquist.

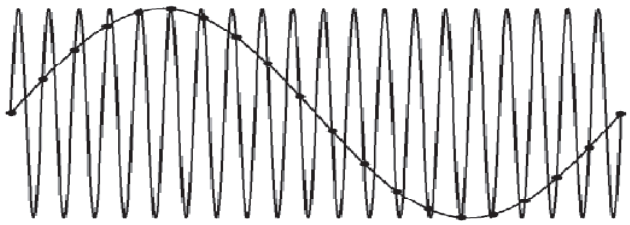


Fig. 1.4 Ejemplo de una señal sinusoidal que no cumple con el teorema de Nyquist. Los puntos negros representan las muestras tomadas de la señal analógica. Referencia [8].

Los conversores AD (Análogo a Digital) son los encargados de tomar las muestras y entregar un dato numérico correspondiente. Es necesario, sin embargo, indicarle al dispositivo la frecuencia de muestreo a la que debe trabajar. Una vez que se obtiene un número que representa el valor de una muestra, se querrá utilizarlo de alguna manera. Un conversor generalmente guarda esta información por un tiempo muy pequeño, de modo que para no perder la información se la envía hacia algún procesador de información. Es común que sea un microcontrolador quien, además de haber fijado la frecuencia de muestreo, reciba esa información y haga algo útil con ella.

1.3.2. TRANSMISIÓN

Tomando en cuenta que nuestro instrumento está (en la mayoría de casos) físicamente en contacto con el fenómeno que se estudia, es necesario trasladar la información de un lugar (espacial) a otro. Los sistemas actuales nos proveen de varias alternativas para este efecto, que de cualquier manera se pueden reducir a dos posibilidades: mediante una conexión física o de maner a inalámbrica.

Utilizar una conexión física se refiere al uso de algún tipo de cable para la transmisión de información. En principio se puede variar la diferencia de potencial entre dos hilos de un cable en un extremo y medir esa diferencia en el otro extremo. Indiscutiblemente esto se traduce a la utilización de la corriente eléctrica como portadora de información.

La forma inalámbrica de transportar información se resume en la utilización de ondas electromagnéticas como medio. Es necesario en este medio un transmisor que genere las ondas y un receptor que sea capaz de reproducir los mensajes que le fueron enviados.

Existen varios aspectos que nos llevan a escoger entre estas dos alternativas, uno de ellos es el alcance (en distancia) que nos ofrecen: la comunicación inalámbrica puede resolver el problema de intentar extender un cable de varios kilómetros; mientras un cable brinda mayor confiabilidad y reduce enormemente los costos en distancias pequeñas.

Existen algunas alternativas para obtener la comodidad de una transmisión inalámbrica a bajo precio, a costa de su alcance. Básicamente cuando se prefiere esta “comodidad” no es sino dentro de una red de información, donde no existen únicamente un punto de partida y uno de llegada, sino una cantidad mayor de nodos donde cada uno puede entregar y requerir información de los otros.

Para citar un ejemplo de lo que se acaba de menci onar, se puede traer a la mente a la Internet. Aunque se refiere a una enorme red de redes de computadores,

responde al hecho de que todos los nodos demandan y entregan información a otros nodos. La cantidad de nodos que esta mega red contiene es tan grande que su complejidad ha crecido inimaginablemente, y ha puesto a dar vueltas al mundo a la palabra *globalización*.

Afortunadamente, una sola palabra no es lo único que ha dado vueltas por el mundo entero. Es posible usar la Internet como medio de transmisión de información relativa a una infinidad de temas. La parte medular en ésta, como en cualquier otra red es enviar el mensaje en un formato definido, de modo que todos los computadores conectados en red puedan “hablar el mismo idioma” [9].

1.3.3. ALMACENAMIENTO

Al estar inmersos en un ambiente computarizado se puede hablar de varias formas de manejar información. Una de las más útiles es mantenerla en un espacio de memoria para tener acceso a ella cuando sea necesario. Sin embargo, aún con las facilidades que ofrecen los computadores puede resultar un tanto difícil el acceder a la información, sobre todo si ésta es abundante y se la almacena de una manera no sistemática.

Una base de datos (BDD) constituye un conjunto de objetos (espacio de memoria) en el que se organiza la información para su fácil manejo. Normalmente una los objetos mencionados son tablas cuyas columnas se denominan *campos* y las filas constituyen *registros* [10]. Dicho de otra manera, un registro contiene los datos de una serie de campos; en nuestro caso, un registro contiene la información de la temperatura del aire, la humedad relativa, la cantidad de luz y la pluviosidad para un instante dado, además de la identificación de la estación de la que proviene la información.

En una BDD las tablas se relacionan entre sí. Por ejemplo, puede existir una tabla que contenga la información de un empleado de una empresa, y otra tabla con los dependientes de los empleados; en ambas tablas habrá un campo ‘número de empleado’ que, si su valor coincide, indicará que los dependientes (en la tabla 2)

corresponden al empleado cuyos datos están en el registro de la tabla 1 [10]. Para manejar las relaciones entre tablas de una BDD se recurre a un recurso de software administrador de bases de datos, como MySQL, MSAccess, Or acle, etc.

2. MONTAJE EXPERIMENTAL

2.1. DESCRIPCIÓN DEL HARDWARE

El sistema se ha diseñado en la forma que muestra el diagrama de la Fig. 2.1. Se desarrollaron dos esquemas diferentes para el dispositivo de adquisición de datos. El primer esquema se desarrolló a base de un microcontrolador sencillo, al cual llamaremos *'Sistema Tradicional'*, y el segundo está basado en un dispositivo TELOS RevB de la familia MOTEIV, que en adelante se referirá como *'Sistema TELOS'*. A continuación se detallan las características de ambos equipos. La comparación de estos dos sistemas se realiza en el capítulo 3.

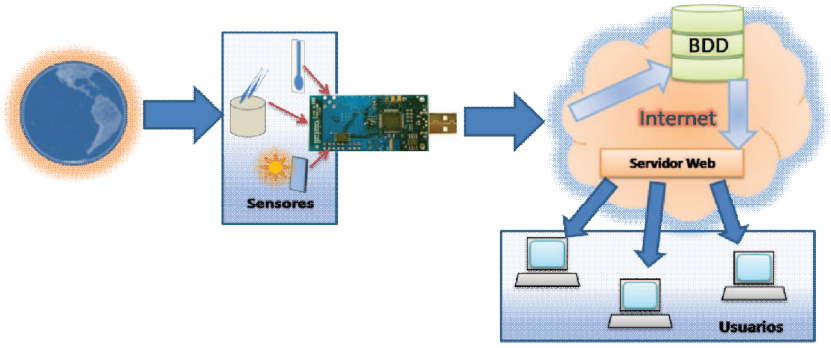


Fig 2.1 Diagrama que describe el funcionamiento del sistema de recolección y distribución de información.

2.1.1. SISTEMA TRADICIONAL

Básicamente se compone de un microprocesador al que se anexan sensores de temperatura y pluviosidad. El microcontrolador se comunica por medio de un puerto serial con un radio transmisor (externo) o directamente con una PC. El detalle gráfico se presenta en el Anexo C.

El diseño se basa en el microcontrolador PIC16f870 del fabricante Microchip (Arizona, EEUU), escogido por la versatilidad que ofrece además de la sencillez en cuanto a su programación. Algunas de sus características que se han aprovechado en el desarrollo de este hardware son [11]:

- Cinco canales de conversión analógica-digital de 10 bits
- Dos puertos adicionales (cada uno de 8 entradas/salidas) para periféricos
- Comunicación serial
- Tres temporizadores
- Sistema WDT (Watch Dog Timer) de autoreinicio

La conversión analógica-digital puede verse como una función que toma una muestra de voltaje y le asigna un número entre 0 y 1024 (esto equivale a una resolución de 2^{10} valores posibles). En la práctica, se divide al rango analógico en 2^{10} segmentos o valores digitales correspondientes. Por ejemplo, si el valor del voltaje (analógico) está dentro del primer segmento, el conversor devuelve el número 0000000001 en binario, que en decimal es 1; si el valor del voltaje estuviese dentro del décimo quinto intervalo, el conversor le asigna el número 0000001111, que es equivalente a 15, y así hasta el valor 1024.

Los puertos digitales, en cambio, realizan lecturas de los 8 pines y forman un número de 8 bits asignando el valor de uno o cero de acuerdo al pin que tenga un nivel alto (encendido) o bajo (apagado) respectivamente. De esa manera toman la información directamente en binario.

En cuanto a la comunicación serial, el microcontrolador utiliza comunicación serial con niveles TTL, que define al uno (1) lógico a un valor entre 4.5V y 5V, y al cero (0) lógico de 0V a 0,8V. Desafortunadamente, los radios transmisores y las PC manejan las señales de diferente manera que el microcontrolador. A diferencia de los valores indicados para el micro, el radio y la PC utilizan el protocolo RS-232, que usa como uno (1) lógico un valor entre -15V y -5V, y como cero (0) lógico un valor entre 5V y 15V. Haciendo una breve analogía, se podría decir que ambos se comunican en el mismo idioma pero con diferentes dialectos. Por esta razón se utiliza el circuito integrado MAX232, del fabricante Maxim Integrated Products Inc. (Sunnyvale, EEUU), que precisamente sirve como un traductor entre el microcontrolador y el radio (o la PC directamente) [12].

Un temporizador es una serie de instrucciones que le indican al procesador que debe incrementar una variable hasta que alcance un cierto valor, fijado por el usuario, de manera que al alcanzar ese valor ha transcurrido el tiempo requerido. Una vez transcurrido el tiempo fijado, se produce una interrupción en el resto de tareas del microprocesador y se levanta una bandera de modo que éste pueda saber el porqué de la interrupción.

En algunas aplicaciones, el microcontrolador se bloquea y deja de realizar las tareas pendientes (el microcontrolador se “cuelga”). En estos casos es necesario realizar un reinicio para que continúe con las tareas programadas. El microcontrolador PIC16f870 tiene implementado un sistema de vigía (Watch Dog Timer) que lo reinicia automáticamente cuando existe algún problema en la realización de tareas [13].

Al microcontrolador se anexó como periférico un sensor de temperatura LM35 del fabricante National Semiconductor (California, EEUU). La respuesta de voltaje del sensor varía linealmente con la temperatura medida en grados centígrados, con una precisión de ±¼°C a temperatura ambiente y de ±¾°C al trabajar en el rango completo desde -55°C a 150°C. El factor de escala (o constante de proporcionalidad) es de 10mV/°C, trabaja en un rango de alimentación de 4V a 40V y consume menos de 60μA, lo que permite que el autocalentamiento sea de máximo 0,08°C en el aire [14].

El Pluviómetro (modelo 260-2501M, Novalinx Corporation, Auburn, EEUU) es un recolector de área circular con dos recipientes formando una palanca, unidos a un eje (similares a los juegos “sube-y-baja” de los niños). Cada vez que uno de los recipientes se llena hace que cambie de lado la palanca y conecta un interruptor magnético [15]. De esta forma, podría decirse que el sensor es un ente que oprime un pulsador, produciendo una interrupción al microcontrolador, cada vez que llueve una cantidad determinada. El trabajo del microcontrolador consiste en contar el número de pulsos que recibe del pluviómetro.

El microprocesador es el encargado de recolectar la información que proporciona el sensor de temperatura a través del conversor AD interno; de contar las interrupciones del pluviómetro; de leer el estado de la batería a través de otro canal del conversor AD; de ordenar esa información en un arreglo (trama) donde conste la identificación de la estación, la cantidad de datos que contiene el arreglo, y la información misma de los sensores; y enviar esa información hacia el transmisor externo a través del puerto serial.

El arreglo que contiene la información proporcionada por los sensores se rige por el estándar PPP (Point-to-point Protocol). Es decir, se forma un arreglo (trama) que contiene una cabecera con información de quién lo envía, a quién va dirigido, entre otros detalles; un cuerpo, que es donde se ubica la información que queremos enviar; y un final de trama, que contiene un parámetro con el que es posible asegurarse que la información no haya sido cambiada de alguna forma durante su transporte.

2.1.2. SISTEMA TELOS

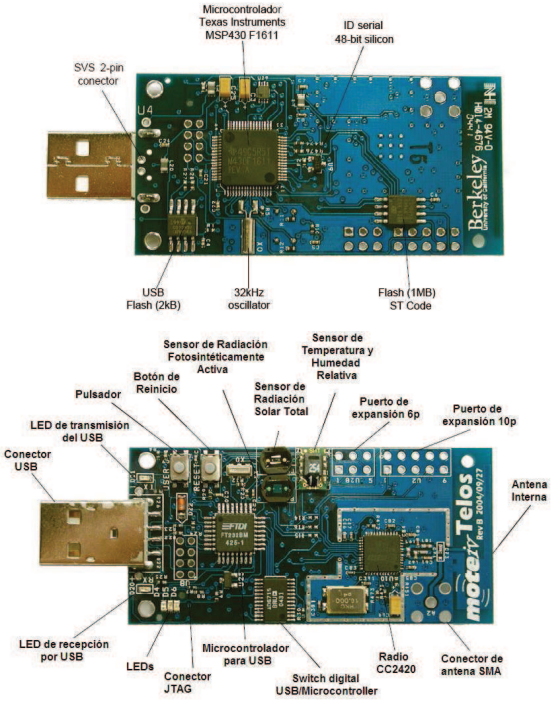


Fig. 2.2 Ubicación de componentes en un dispositivo TELOS RevB, arriba el anverso y abajo el reverso. Adaptado de la referencia [16].

Un dispositivo TELOS es un módulo inalámbrico de ultra bajo consumo energético, diseñado para usarse en redes de sensores (modelo RevB, 2004, San Francisco, EEUU). Está conformado por un microcontrolador, un radio digital, una memoria flash, un microcontrolador para comunicación USB, una antena, un sensor de temperatura y humedad relativa, un fotodiodo sensor de espectro completo, un fotodiodo sensor de radiación fotosintéticamente activa (revisar sección 2.2), y dos puertos de expansión (de 10 pines y de 6 pines) [16]. En la Fig. 2.2 encontramos con detalle la ubicación de estos elementos.

Los puertos de expansión con los que cuenta este dispositivo le brindan una gran versatilidad. A través de éstos se tiene comunicación con el microcontrolador a través de entradas/salidas analógicas, entradas/salidas digitales, puerto serial, puerto de comunicación tipo I2C; además de la posibilidad de un interruptor multipropósito programado por el usuario y un interruptor de reinicio (reset) [16]. El detalle de los puertos de expansión de 10 y 6 pines se encuentra detallado en la Fig. 2.3.

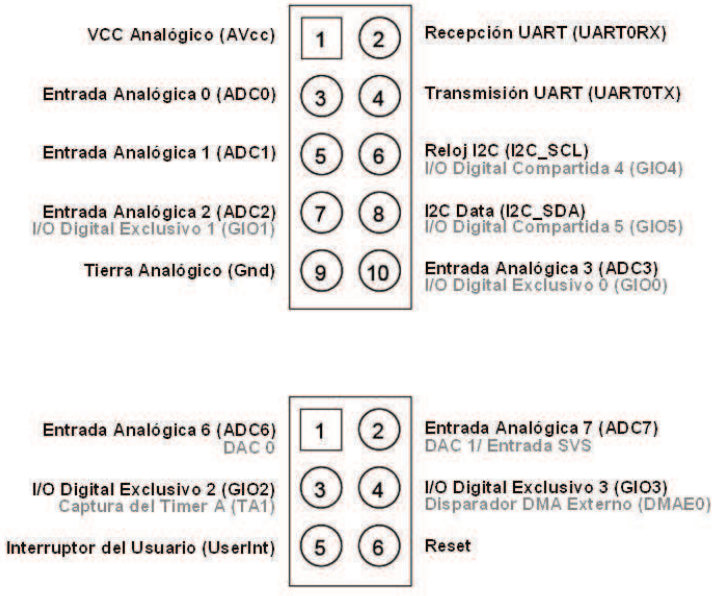


Fig. 2.3 Detalle de los puertos de expansión de un dispositivo TELOS RevB. En gris se especifica las funciones alternativas. Adaptado de [16].

Con estos puertos es posible una comunicación directa al microcontrolador, y es aquí donde los sensores externos se conectan físicamente. Los sensores externos conectados en este puerto de expansión son el sensor de temperatura LM35 y el pluviómetro. El sensor de temperatura provee una señal analógica, por lo cual se conecta a una entrada analógica del TELOS; el pluviómetro actúa como un interruptor, por lo que se conecta directamente al interruptor programable en el pin # 5 del conector de expansión de 6 pines.

De esta manera, el nodo construido a base de TELOS recolecta información de 7 parámetros. Cada uno de estos parámetros está ligado a un sensor diferente y son:

- Temperatura (sensor interno)
- Humedad relativa (sensor interno)
- Intensidad lumínica (espectro visible completo, sensor interno)
- Intensidad lumínica (espectro fotosintético, sensor interno)
- Pluviosidad (sensor externo)
- Temperatura externa (sensor externo)
- Nivel de batería (sensor interno)

A continuación se detalla el funcionamiento de estos sensores.

2.1.2.1. Sensor de Temperatura

El dispositivo TELOS contiene empotrado un sensor digital SHT11 de temperatura/humedad del fabricante Sensirion AG (Staefa, Suiza). Este sensor contiene anexa una memoria EEPROM (Electrically Erasable Programmable Read-Only Memory) donde se guardan los coeficientes de calibración, de modo que su utilización se facilite al máximo [17].

En la Fig. 2.4 se describe la precisión del sensor de temperatura en función de la temperatura.

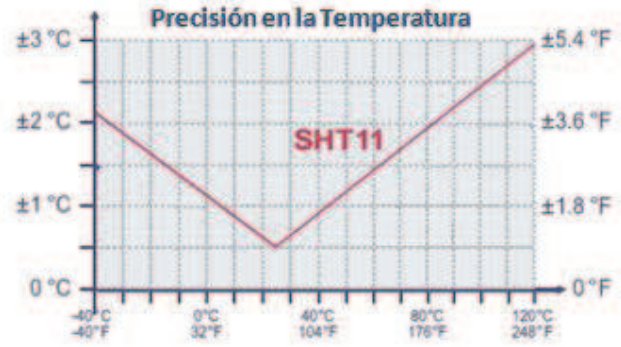


Fig. 2.4 Precisión del sensor SHT11 en función de la temperatura. Adaptado de la referencia [17].

Tomando en cuenta que la temperatura del aire en Quito difícilmente llega a cero grados centígrados y de igual manera será poco probable que sobrepase los cuarenta grados centígrados, la precisión del sensor será menor a ±1.1 °C y mayor a ±0.5 °C. Si decimos que en un día normal en Quito la temperatura varía entre 12 y 20 grados centígrados, la precisión con la que el sensor puede tomar medidas está entre ±0.6 y ±0.9 °C.

2.1.2.2. Sensor de Humedad relativa

Como se menciona arriba, el dispositivo TELOS viene con el sensor SHT11. El elemento sensible a la humedad de este sensor se basa en el cambio de la capacidad de un condensador [18]. Constituye un condensador cuyo dieléctrico es un polímero que absorbe o elimina agua proporcionalmente a la humedad relativa del medio, cambiando la capacidad del dispositivo [4]. El sensor SHT11 incluye además un circuito (conversor AD) que recoge la información de la variación del condensador y la envía a una interfaz digital. La arquitectura del sensor SHT11 se encuentra detallada en la Fig. 2.5.

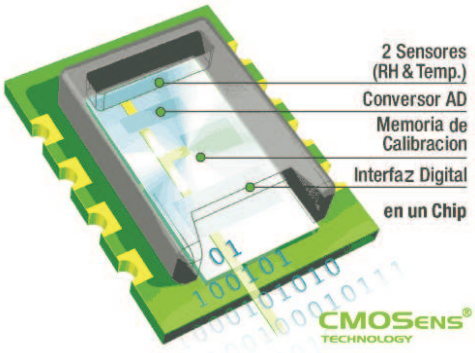


Fig. 2.5 Arquitectura del sensor SHT11 de humedad relativa y temperatura. Adaptado de la referencia [18].

2.1.2.3. Intensidad Lumínica (Espectro Completo y Espectro Fotosintético)

Para la medición de intensidad lumínica se han incluido fotodiodos (Hamamatsu Photonics K.K., Hamamatsu, Japón) en el dispositivo TELOS, uno de ellos opera en el rango visible completo incluyendo parte del infrarrojo (en adelante se referirá a éste como *rango visible completo*), de 320 a 1100 nm (modelo S1087-01), mientras el otro lo hace solamente en el rango fotosintético, de 320 a 730 nm (modelo S1087) [19].

Los fotodiodos son componentes electrónicos que entregan una corriente en sus electrodos que varía en función de la intensidad de luz que reciben [19]. Esta corriente pasa a través de una resistencia apareciendo en los extremos de ésta un voltaje V_I que obedece la ley de Ohm. Una entrada analógica del TELOS recoge el valor de V_I y entrega esta información para ser procesada.

2.1.2.4. Sensor de Pluviosidad (Pluviómetro)

El Pluviómetro implementado es fabricado por Novalynx Corporation, el modelo es 260-2501M. Es el mismo que se usó en el primer diseño (Fig. 2.6); por lo tanto, la función del microprocesador que contiene el TELOS es, así como en el caso del PIC16F870, contar la cantidad de interrupciones que da el pluviómetro en un intervalo determinado de tiempo.

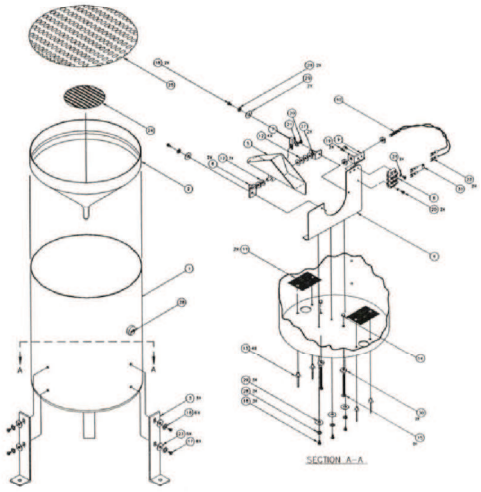


Fig. 2.6 Esquema de construcción del Pluviómetro Novalynx. Este pluviómetro es un recolector de área circular con dos recipientes formando una palanca, unidos a un eje (similares a los juegos "sube-y-baja" de los niños). Cada vez que uno de los recipientes se llena hace que cambie de lado la palanca y conecta un interruptor magnético. Tomado de la referencia [15].

Cada vez que el recipiente se llena con la lluvia el peso hace que gire la palanca, un conector magnético produce un pulso de 100 ms y queda listo el recipiente en el otro lado para llenarse si sigue lloviendo. Los dos recipientes son del mismo volumen y vienen calibrados de fábrica para dar un pulso por cada milímetro de lluvia recogida [15].

2.1.2.5. Sensor de Temperatura Externa

Debido a que el sensor de temperatura y humedad relativa incluido en el TELOS se encuentra localizado a un costado de los fotodiodos, necesariamente permanecerá expuesto a la luz del sol. El impacto directo de la luz solar causa variaciones en la temperatura, produciendo errores [3].

Por este motivo se ha implementado un sensor externo de temperatura, que se ubicará en un lugar cercano al dispositivo pero a buen recaudo del impacto directo de la luz solar. La diferencia entre ambas lecturas de temperatura y su relación con los datos de intensidad lumínica puede resultar un tema de investigación bastante interesante.

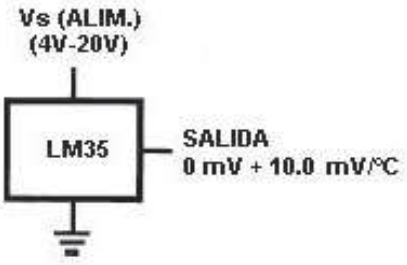


Fig. 2.7 Configuración usada para el sensor de Temperatura LM35. Adaptado de la referencia [14]

El sensor externo de temperatura es un LM35 (National Semiconductors, Santa Clara-CA, EEUU). Este sensor tiene una respuesta en voltaje que varía linealmente con la temperatura y viene calibrado de fábrica para proveer 10 mV por cada grado centígrado [14]. Como se había mencionado con anterioridad, el TELOS recoge la información de este sensor a través de uno de los canales del

conversor AD. La configuración usada para el sensor LM35 es la más sencilla propuesta por el fabricante, se detalla en la Fig. 2.7.

2.1.2.6. Nivel de batería

La fuente de alimentación de la estación no contiene un sustento como paneles solares o alguna otra alternativa que le permita recargarse; y a pesar de ser un diseño de ultrabajo consumo de energía, la batería tiene un tiempo de utilidad limitado. El medidor de batería nos indica qué tan lejos está la culminación de ese tiempo, y si es necesario realizar un cambio de baterías.

El medidor de batería consiste en un divisor de tensión como se muestra en la Fig. 2.8. Este divisor se conecta a una entrada del conversor AD [16], y de ese modo obtenemos un valor numérico que nos indica si la batería está en un buen estado o es necesario reemplazarla.

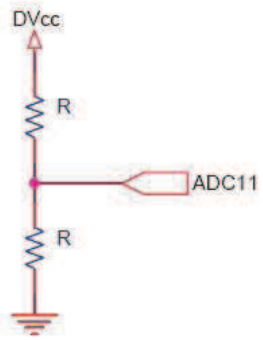


Fig. 2.8 Divisor de tensión que actúa como sensor de nivel de batería para el dispositivo TELOS. Tomado de la referencia [16].

El sistema TELOS contiene un radio transmisor/receptor digital CC2420 (Chipcon, Dallas, EEUU) y una antena empotrada con la que puede comunicarse con otro TELOS separado hasta 150 m [16]. Existe la posibilidad de usar una antena externa con la que la distancia de comunicación inalámbrica puede aumentar hasta 400 m. Usando ésta última configuración, el sistema TELOS envía la información recogida de los sensores a un TELOS Base, que recibe la información vía radio y la ingresa en una base de datos dentro de una PC utilizando el puerto USB.

2.2. DESCRIPCIÓN DEL SOFTWARE

El software desarrollado en el presente trabajo está constituido por dos partes principalmente: Una aplicación en lenguaje NesC, que se ha de ejecutar en el dispositivo TELOS (ver código fuente en el Anexo A); y el software para presentación de información desarrollado en lenguaje Java y JSP (Java Server Pages), destinado a ejecutarse en una PC (ver código fuente en el Anexo B).

2.2.1. PROGRAMA EN NesC

El TELOS trabaja con un sistema operativo llamado TinyOS, bajo el cual se ejecuta la aplicación en lenguaje NesC. El lenguaje NesC se puede describir como una variación del lenguaje C (o C++) tradicional, con la diferencia de que NesC fue diseñado para computadores pequeños (los llamados *embedded systems*) y está orientado al manejo de componentes. [20]

Un componente es un pedazo de código que controla una parte específica del dispositivo. En el caso del TELOS tenemos un componente para cada sensor, uno para el ADC, otro para el radio, otro para el temporizado. Estos componentes se enlazan y comunican entre sí utilizando interfaces, y la forma en la que se enlazan (denominada *wiring*) se especifica en un archivo con extensión *.nc*, que constituye un módulo de configuración. [21]

Debe existir, además, otro archivo con extensión *.nc* que contenga las instrucciones específicas para que el TELOS trabaje como el usuario desea. Este archivo es un módulo de implementación. [21]

En nuestro caso, a la aplicación se dio el nombre de *Sensores*, que incluye un módulo de configuración *Sensores.nc* y un módulo de implementación *SensoresM.nc*. La aplicación contiene además un archivo (*ExtSensores.h*) con especificaciones y definiciones para el manejo del ADC. Básicamente en esta aplicación se define un estado para cada sensor, y hace correr un temporizador para producir una interrupción. En cada interrupción el sistema verifica el estado

en el que se encuentra y recoge la información del sensor que corresponda; si no se ha producido ningún error en el proceso entonces se cambia el estado para que en la siguiente interrupción se recoja información de un sensor diferente, y se inicia el temporizador. En la interrupción correspondiente al último sensor se adiciona una instrucción para enviar por el radio un mensaje con la información de todos los sensores; se cambia el estado y se inicia el temporizador para que la siguiente interrupción se produzca después de 30 minutos (o el tiempo que se decida es óptimo. Esto debe programarse).

Por su parte, el receptor o Nodo Base TELOS (Fig. 2.9) contiene una aplicación en lenguaje NesC llamada *TOSBase*. Esta aplicación viene como ejemplo en la distribución de TinyOS. La función de esta aplicación es recibir paquetes de datos por el radio y enviarlos a través del puerto USB; o en su defecto, empaquetar la información proveniente del puerto USB y enviarla utilizando el radio.

2.2.2. PROGRAMA EN JAVA

La segunda parte del software, que se ha de ejecutar en una PC, está constituido a la vez por dos programas: *RecibeTelos*, desarrollado en lenguaje Java; y *RedClimatologicaQuito*, que es una aplicación JSP (html con código Java).

El ingreso de datos al computador es posible mediante un *nodo base TELOS* (receptor TELOS, Fig. 2.9), conectado a un puerto USB de la PC.

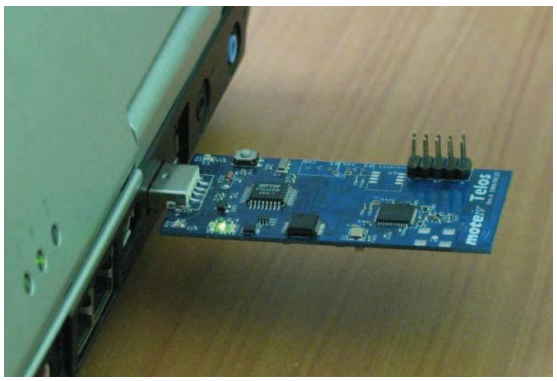


Fig. 2.9 Nodo Base TELOS. La información ingresa de manera inalámbrica por este receptor hacia el puerto USB de un computador, para guardarse en una base de datos.

El programa RecibeTelos tiene por objetivo el ingreso de los datos provenientes del nodo TELOS hacia una base de datos (BDD) MySQL. Es así que las principales tareas de este programa comprenden la apertura del puerto USB de la PC al que está conectado el receptor TELOS; la conexión entre el programa y la BDD; el reconocimiento de los paquetes de datos que ingresen por el puerto USB y el almacenamiento de éstos en la BDD.

Esta aplicación receptora de datos se muestra como una ventana muy sencilla que le agrega un ambiente gráfico, amigable con el usuario (Fig. 2.10).

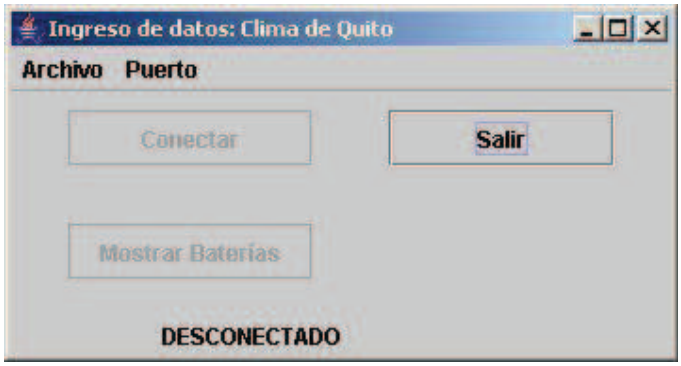


Fig. 2.10 Ventana generada por el programa RecibeTelos.java. Su carácter gráfico le hace amigable con el usuario.

2.2.3. PROGRAMA EN JSP

En cuanto a la aplicación *RedClimatologicaQuito*, se puede describir como una página web que presenta los datos recolectados por una red de estaciones meteorológicas en la ciudad de Quito. En este caso existe un nodo de recolección y uno de base (mencionado anteriormente). En la página se ha insertado código en lenguaje Java, con el fin de presentar gráficamente los datos obtenidos del nodo TELOS.

Gracias a la creciente popularidad de la Internet, se desarrollan a diario nuevas técnicas y métodos de presentación de información a través de páginas web. De hecho, durante la realización del presente trabajo ha existido un cambio tremendo en la fusión entre lenguaje de programación Java y el HTML, siendo reemplazado el JSP por JSF (Java Server Faces). Básicamente, la diferencia radica en la

posibilidad de hacer reutilizable el código desarrollado. Para esto, la aplicación se desglosa en varios archivos, a manera de piezas de un rompecabezas.

2.2.3.1. Portal de Internet

Podemos llamar *Portal de Internet* a un conjunto de páginas electrónicas (a las que en adelante nos referiremos simplemente como 'página') que presentarán información tanto técnica como general, de una manera fácil y atractiva al usuario. Este portal se compone de cinco páginas individuales con un esquema que empieza con (1) una introducción, lleva a (2) la presentación de los datos recolectados, (3) muestra la información general del proyecto, (4) propone enlaces que pueden resultar de interés y (5) brinda la posibilidad de enviar comentarios y sugerencias. Existe también la posibilidad de suscribirse a un servicio de información y mensajería que permite recibir los datos obtenidos de manera periódica en el correo electrónico del suscriptor.

Fig. 2.11 Página de Inicio del portal 'Red Climatológica de Quito'. Presenta información de las últimas 24 horas además de invitar al usuario a revisar información en periodos con fechas de inicio y final personalizadas.

La primera de estas páginas (Fig. 2.11), como se ha mencionado, es la encargada de introducir el trabajo invitando al usuario a interactuar con el sistema. Como

resultado de la visualización de esta página, y como requisito para la siguiente, se define el parámetro o parámetros específicos de los que se requiere la información y el intervalo de tiempo en el que ésta se produjo. Los parámetros entre los que el usuario puede escoger son la temperatura, la humedad relativa, la luminosidad y la cantidad de lluvia (desde luego un número adicional de sensores podrán integrarse en el futuro). Mientras que el período de tiempo se determina por una fecha inicial y una fecha final.

Una vez que se han determinado los parámetros meteorológicos y el intervalo de tiempo que le interesan al usuario del sistema, se ha de presentar la información en manera gráfica (Fig. 2.12). El portal utiliza un pequeño programa o *applet*♦ para graficar los datos recolectados durante el intervalo escogido; este applet lleva el nombre de *Componente,* fue diseñado para graficar señales sísmicas pero se puede adaptar para ser usado en otras aplicaciones.

Fig. 2.12 Página de Reportes del portal 'Red Climatológica de Quito'. Presenta gráficos de los datos del clima y permite obtener un reporte tabulado en un archivo de formato Microsoft Excel.

♦ Programa en lenguaje Java que se ejecuta en la PC en la que el usuario está visualizando la página de Internet, independientemente del sistema operativo y del programa de internet. Se dice que se ejecuta en *'el lado del cliente'*.

La información científica que pueda ser obtenida del clima no sería de utilidad en absoluto si no puede ser compartida. Expresando esta idea de otra manera diríamos que el contenido de este portal no sería científico si no le permitiese al usuario contar con una tabla que contenga los datos que le interesan en un formato útil. De manera que la opción de guardar una tabla de los datos requeridos es viable, y el formato del archivo en el que se guarda es de Microsoft Excel, por ser uno de los más populares.

Es importante además, recurrir a otras fuentes de datos y de información relacionada con nuestro tema de investigación. Para facilitar en cierto grado la divulgación, existe una sección en el portal en la que se presentan unos pocos enlaces que podrían resultar de interés para el investigador.

Como información adicional se presenta una breve descripción del proyecto como parte del portal. Además existe la posibilidad de recibir información de los usuarios a través de una página que les permite enviar correos electrónicos con comentarios y/o sugerencias (Fig. 2.13).

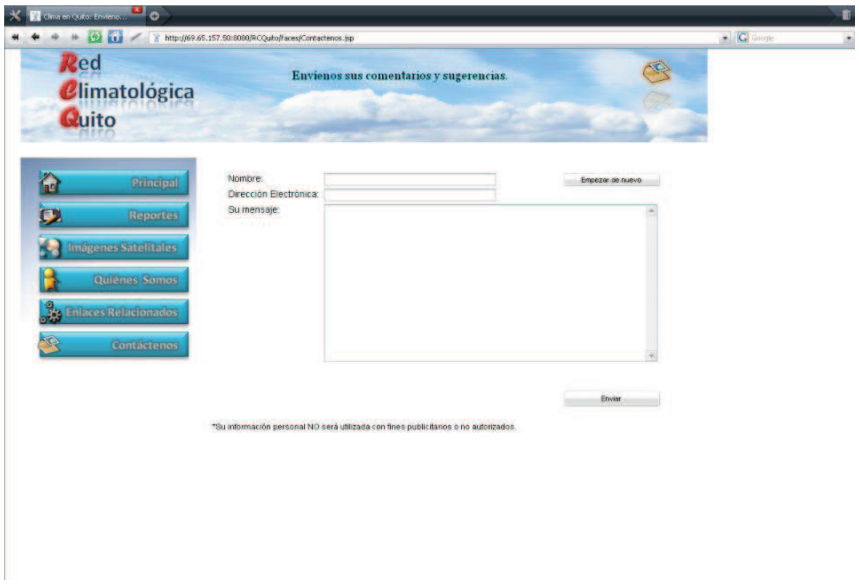


Fig. 2.13 Página de Comentarios y sugerencias del portal 'Red Climatológica de Quito'. Compone un mensaje de correo electrónico con los comentarios y/o sugerencias que escriban los usuarios del portal 'Red Climatológica Quito'.

3. RESULTADOS Y DISCUSIÓN

3.1. DATOS OBTENIDOS

En el presente trabajo se ha descrito un sistema inalámbrico de recolección y distribución de información climática, directamente accesible en tiempo real por Internet e integrable en una red de adquisición de datos. En principio, el concepto y el diseño son aplicables a diferentes tipos de redes de adquisición de datos para una amplia variedad de sensores y tipos de datos (por ejemplo, sísmicos). El primer componente del sistema es el nodo recolector, éste consiste en el conjunto de los sensores que adquieren la información en contacto directo con los fenómenos físicos en estudio. Esta información se transforma en señales electromagnéticas y luego en bits: un formato adecuado para que puedan ser manejados por un computador[♠]. Los bits son procesados a continuación (se arreglan en tramas) y transmitidos a la red. Un segundo componente, el nodo receptor, recoge estas tramas de información, las almacena y las distribuye a la red para que puedan ser usadas con posterioridad. Por último, un tercer componente del sistema maneja la distribución de la información sobre protocolos de Internet para que pueda ser utilizada por quien tuviere interés en ello, básicamente en cualquier lugar del mundo que cuente con conexión de Internet.

En principio, el sistema puede operar ininterrumpidamente con el mantenimiento mínimo básico, sin embargo existen situaciones externas que pueden provocar pérdida de datos. Entre las causas de estas pérdidas se incluyen los cortes de energía en el nodo recolector, que provocan que el computador a través del que ingresa la información a la base de datos se apague; fallas en la conexión a internet debido a por problemas con el proveedor (enlace WAN) o incluso dentro de la red de área local (LAN); falta de respuesta en el programa de recepción, posiblemente por falta de memoria RAM en el computador durante el ingreso de alguna trama; deficiencias en la administración del sistema: el administrador del nodo colector o del nodo distribuidor no toma las medidas necesarias para que el

[♠] Puede ser un computador 'de escritorio' así como cualquier otro dispositivo que contenga un procesador (por ejemplo, palm, embedded system, etc.).

sistema reanude su trabajo luego de algún desperfecto; y finalmente, otros agentes externos.

La cantidad de información que se genera depende del fenómeno que se estudie. Por ejemplo, el caso de información sísmica demanda frecuencias de muestreo alrededor de los 100 Hz (es decir 100 muestras por segundo). En nuestro caso, además de innecesaria, una frecuencia de muestreo alta nos inundaría de datos provocando problemas de espacio en la base de datos y dificultad para su manejo. Entonces se ha configurado el sistema para tomar realizar una medida completa (de todos los sensores) cada 30 minutos; este tiempo es ajustable el momento de la programación del nodo recolector.

La red es accesible a través de la dirección http://157.100.42.162:8080/RCQuito/ ; en este caso ha sido llamada 'Red Climatológica Quito' (RCQ). Para demostración de las capacidades del sistema, en este trabajo se presentan y discuten los datos tomados durante el mes de noviembre de 2008. Los sensores actualmente instalados y en operación son uno de temperatura interna, uno de humedad relativa, uno de luminosidad a espectro completo, y otro de luminosidad en el espectro fotosintético; estos sensores son parte de la dotación original del dispositivo TELOS usados en este nodo; y finalmente un sensor de temperatura externa, y un pluviómetro.

Con esta configuración, la RCQ recoge 48 registros diarios, cada uno de aproximadamente 144 bytes, lo que resulta en 6,9 Kbytes diarios y 2,5 Mbytes al año. Es una cantidad pequeña de información para un computador, de manera que las búsquedas en base de datos no toman más de unos pocos milisegundos en completarse.

De la misma manera lo referente a la transmisión de los datos. Cada trama generada en el nodo recolector contiene únicamente 28 bytes, siendo transmitidos a una velocidad de 250 kbps, lo que significa menos de un milisegundo para llegar al nodo receptor. Es evidente que el tiempo de transmisión de la información no interfiere en absoluto en el concepto de 'tiempo real'.

Para revisar los datos se ingresa a la página web de la Red Climatológica de Quito con la dirección indicada arriba. Una vez dentro de la página de bienvenida, existe la opción de escoger los parámetros meteorológicos que se desea analizar, y el periodo de tiempo en el que se verán los datos de estos parámetros elegidos. Esta información puede desplegarse gráficamente en la página, pero además existe la posibilidad de obtener un reporte en formato Excel, en forma de una tabla que contiene los datos con sus fechas y horas específicas. Se puede decir que en esta tabla se encuentran los datos 'en crudo' (raw data). Ejemplos de gráficos generados a partir de estos datos en crudo se muestran en la Fig. 3.1.

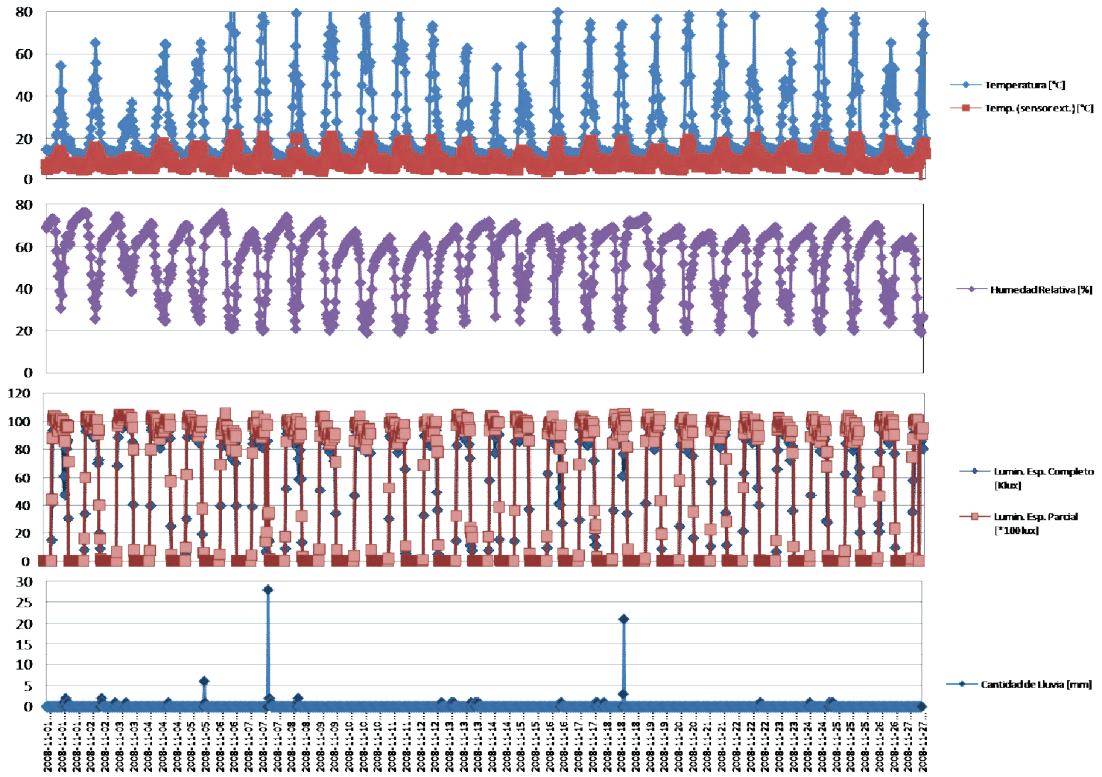


Fig 3.1 Información de los parámetros meteorológicos en Quito obtenidos durante el mes de Noviembre de 2008. Se grafican temperatura externa, temperatura interna, luminosidad a espectro completo, luminosidad en el espectro fotosintético, humedad relativa.

Para comparación, la Fig. 3.2 muestra la página de Reportes de la Red Climatológica Quito donde se puede ver la presentación gráfica de los datos en línea.

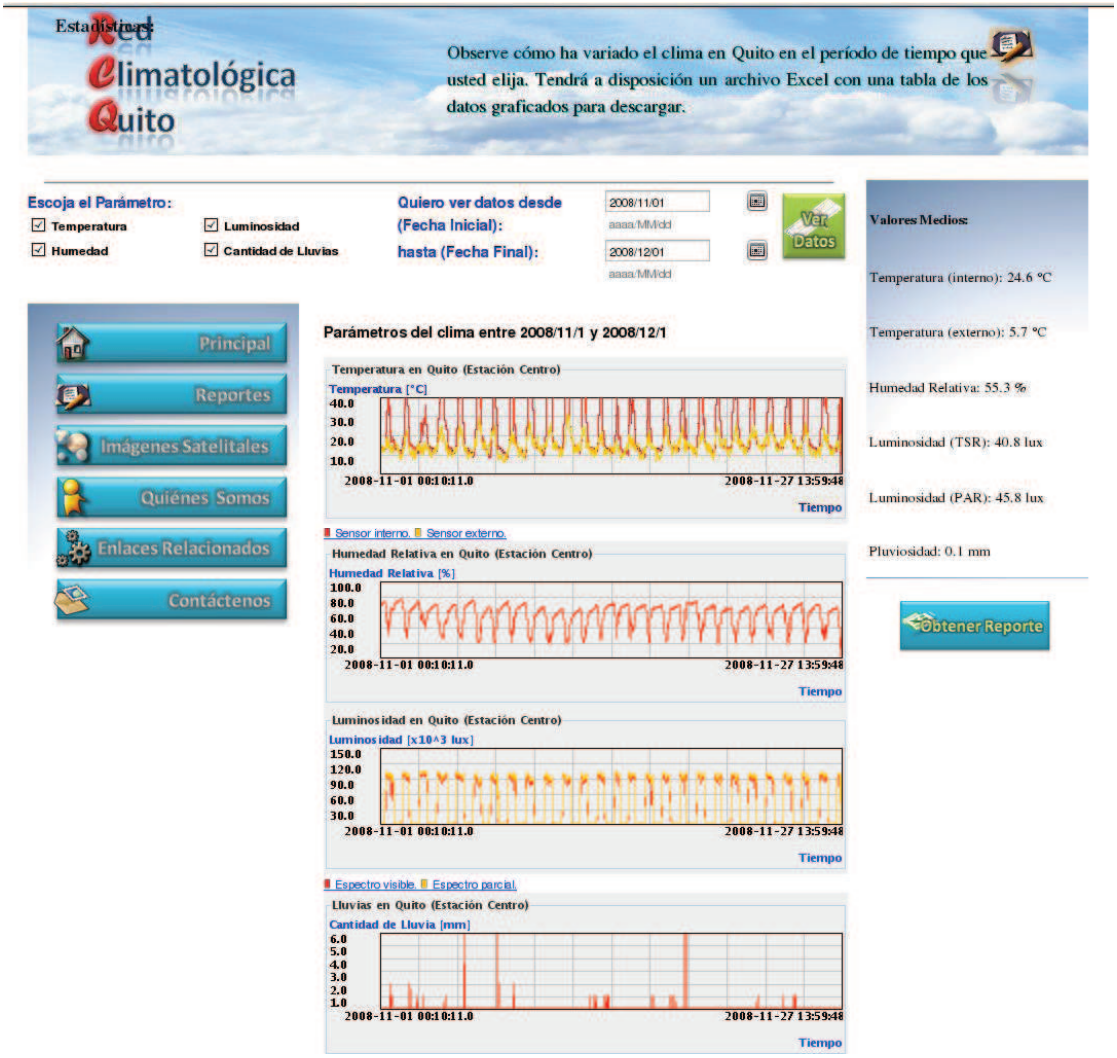


Fig 3.2 Página de Reportes de la Red Climatológica de Quito. Se observan los gráficos generados en línea, los promedios en la parte derecha y el botón para obtener los datos en un archivo Excel. Desde el panel superior se puede indicar las fechas y los parámetros que se quieren revisar.

Para entender los valores presentados, es importante indicar la forma en que se ha instalado el nodo recolector. El dispositivo TELOS de colección/transmisión de datos se encuentra dentro de una caja plástica con cubierta transparente (Fig. 3.3) para evitar que el agua de lluvia y el polvo puedan dañar el dispositivo, y al mismo tiempo permitiendo que la luz llegue a los fotodiodos. La caja tiene dos agujeros para que el aire pueda circular y así se puede medir la humedad relativa. Los sensores externos (termómetro y pluviómetro), se conectan con el dispositivo a través de cables, usando conectores impermeables en la caja y en el terminal del cable (Fig. 3.3).

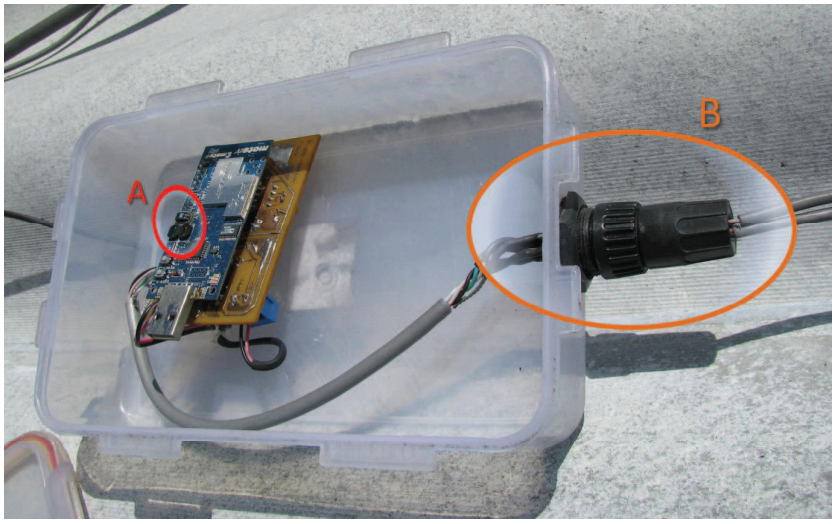


Fig. 3.3. Nodo recolector del sistema TELOS dentro de una caja transparente para que la lluvia y el polvo no dañen el dispositivo. En la parte A se muestran los fotodiodos junto al sensor de temperatura/humedad relativa. En la parte B se aprecian los conectores en la caja y el terminal del cable de sensores externos.

Es claro, que bajo esta configuración, el termómetro encerrado en la caja muestre temperaturas mucho más altas, hasta de 80°C, que la del aire fuera de la caja. El sensor externo por su parte está solamente protegido del impacto directo de la luz solar y del viento, pero en el ambiente externo directo, lo que hace más confiable su lectura de la temperatura del aire.

Como en todo sistema de colección de datos, los datos reunidos de manera cruda deben ser necesariamente procesados matemática y estadísticamente para su evaluación. Esto es muy fácil de realizar con el sistema de la Red, por ejemplo es fácil hacer un gráfico de los promedios diarios o verificar la variabilidad de los valores, o determinar tendencias estacionales, etc.

A manera de ejemplo, la Fig. 3.4 muestra detalle de los datos en crudo de la primera semana de Noviembre del 2008. De este gráfico se puede sacar conclusiones interesantes.

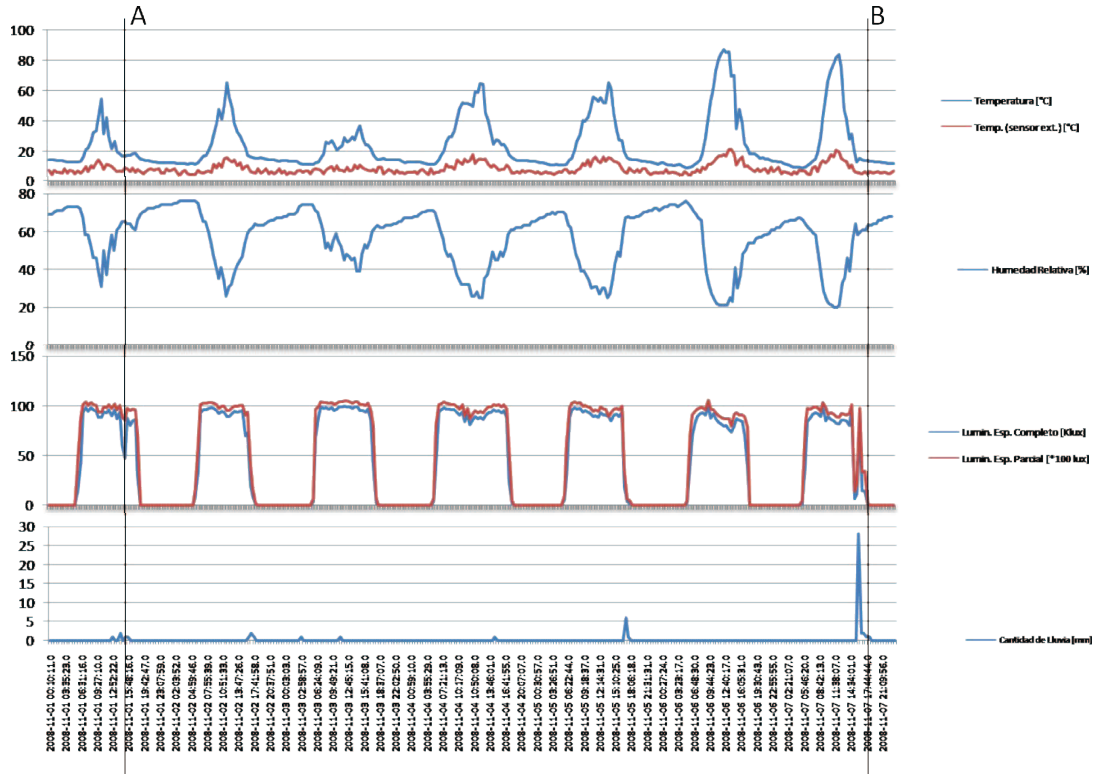


Fig. 3.4 Datos de los parámetros meteorológicos en la ciudad de Quito obtenidos durante la primera semana del mes de Noviembre de 2008.

Para empezar, resalta la relación inversa que existe entre la temperatura y la humedad relativa. Esto es coherente simplemente revisando algunos conceptos termodinámicos. Tomemos una masa de aire m_a que contiene una masa de vapor de agua m_v. La masa de aire necesita una masa m_s de vapor de agua para saturarse; esta cantidad depende de la temperatura a la que se encuentra el aire. La humedad relativa es la relación entre la masa presente de vapor de agua (m_v) en la masa de aire y m_s. En el caso que no se agregue ni se retire vapor de agua a la masa de aire, con un incremento en la temperatura será necesaria una masa mayor de vapor de agua para la saturación, por lo tanto existe un decremento en la humedad relativa [22].

El gráfico de luminosidad indica, por su lado la obvia diferencia entre el día y la noche. Dentro del pico correspondiente al periodo de luz diurna, es posible inferir por la cantidad relativa de señal la nubosidad promedio sobre la zona de medición. Como ejemplo, si se traza una línea en el primer día, un poco antes de

las 15:48 horas (Fig. 3.4, línea A), se observa a derecha un decremento muy marcado en la luminosidad lo cual es una clara indicación de que está muy nublado. Como comprobación de esta situación puede tomarse la señal en el en el pluviómetro como un pulso, lo que indica la presencia de lluvia.

Lo mismo ocurre con la línea B en el día 7 de esa semana. Los valores registrados indican un panorama bastante oscuro, casi como si hubiese llegado el ocaso. En este caso, se puede adscribir la observación al efecto adicional de la intensa lluvia (ver pico en la señal del pluviómetro) que dispersa adicionalmente la luz disponible. Para comparación, la línea indica el tiempo del verdadero ocaso. Nótese que la temperatura del aire también sufre un descenso hasta valores parecidos a los registrados durante la noche. La lluvia que ocasionó esto es un caso particular, el que se describe a continuación.

Para apreciar la validez de las inferencias indicadas y las posibilidades de análisis sobre los datos colectados por la RCQ, consideremos los datos del séptimo día del mes de noviembre. Este día se registró una lluvia torrencial por la tarde (Fig. 3.5), según lo reporta el diario El Hoy en su edición del día 8 de noviembre de 2008, [23]: "*A las 15:30 de ayer, el cielo de Quito se cubrió de gris y una fuerte tormenta que duró 40 minutos provocó inundaciones en los sectores aledaños a la Casa de la Cultura Ecuatoriana […]. Según el Instituto Nacional de Meteorología e Hidrología (Inamhi), la precipitación caída ayer en Quito sobrepasó los 20 ml (20 litros de agua por metro cúbico)*." Esta situación está reportada fielmente en el registro de datos de la Red como se puede ver en la Fig. 3.1, la Fig. 3.4, y la Fig. 3.5. Lo interesante del caso es que esta información está disponible en tiempo real y en cualquier lugar del mundo, para cualquier persona interesada (por ejemplo, los periodistas de El Hoy).

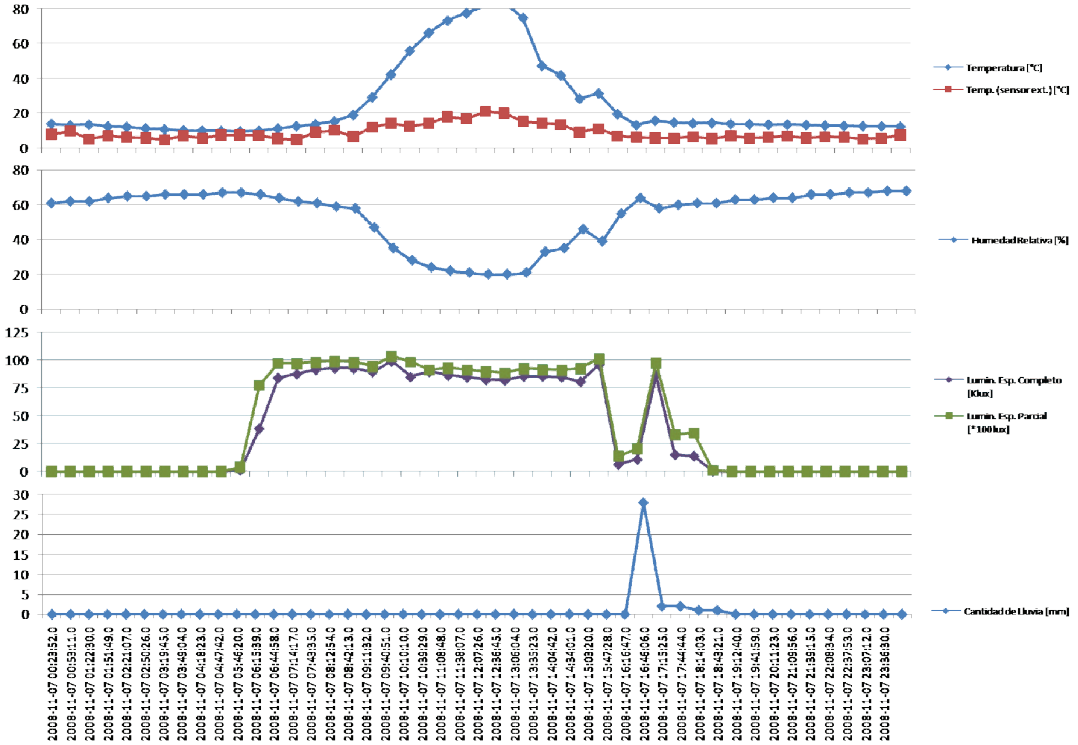


Fig 3.5 Datos meteorológicos de la ciudad de Quito obtenidos el 07 de Noviembre de 2008. Se registra una lluvia torrencial en horas de la tarde.

3.2. DESCRIPCIÓN FÍSICA DEL NODO RECOLECTOR

Dado que el nodo de recolección de datos se encuentra encerrado en una caja de paredes negras con una tapa transparente, y tomando en cuenta que en una de las paredes de la caja existen dos pequeños agujeros, se puede describir como un sistema en contacto con un reservorio termodinámico que además recibe calor de origen externo al reservorio (Fig 3.6).

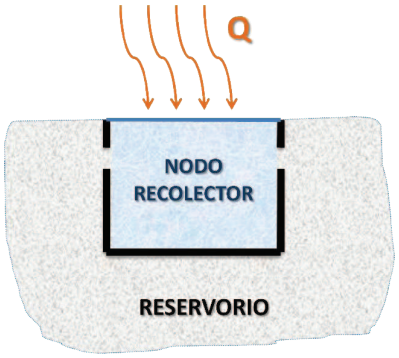


Fig 3.6 Descripción gráfica del nodo recolector como un sistema termodinámico. El nodo recolector se puede describir como un sistema en contacto con un reservorio termodinámico que además recibe calor de origen externo al reservorio mencionado.

El nodo recolector es nuestro sistema en estudio, el cual tendrá una energía media definida por la energía interna del sistema, la energía intercambiada con el reservorio, y el calor recibido por la radiación solar. Se puede expresar de la siguiente manera:

$$\bar{E} = E_i - E_e + Q \tag{2}$$

Donde E_i es la energía interna del sistema; E_e es la energía intercambiada con el reservorio; y Q es el calor recibido por la radiación solar.

La energía interna del sistema se refiere a la energía que tendría el sistema si estuviese aislado; y se puede cuantificar, en primera instancia, de la teoría cinética de los gases dependiendo únicamente de la temperatura [24].

De la misma manera que en un invernadero, el calor que se queda dentro de nuestro sistema en estudio se debe a la diferencia entre la intensidad de la radiación recibida y la intensidad de la radiación emanada [25][26]. En el estudio de los invernaderos solares utilizados en la agricultura se utiliza un factor para cuantificar esta diferencia, y es la transmitancia. Se define la transmitancia como [25]

$$\tau = \frac{I}{I_0} \tag{3}$$

Donde I es la intensidad de la radiación debajo del material cobertor del invernadero, e I_0 es la intensidad de la radiación fuera del invernadero. La transmitancia es una propiedad del material del que se fabrica el invernadero [25], y es posible encontrar valores tabulados en la literatura al respecto, como en el caso de Ganem Carolina et al. [26]. Para el polietileno común, material del que está hecha la cubierta de nuestra caja en estudio, el valor es de 0,92 [26].

En este experimento se ha medido la iluminancia I en lux. Esta medida nos lleva a la intensidad de la radiación dentro de la caja. La iluminancia es un parámetro fotométrico; como las demás magnitudes fotométricas, ésta mantiene una relación con su magnitud análoga radiométrica mediante una curva de sensibilidad del ojo humano en función de la longitud de onda de la radiación [27].

Uno de los parámetros más importantes en este tipo de estudio es la energía radiante *W*, que se refiere a la cantidad total de energía emitida por la fuente [28]. Por su lado también está la potencia radiante *P* (también llamada *flujo radiante* Φ) es la energía radiada por unidad de tiempo. Considerando un elemento de área de una fuente de luz *dA* (Fig. 3.7), la potencia emitida por *dA* en el ángulo sólido $d\Omega$ es [28]

$$dP = L(\theta)\, dA\, d\Omega \tag{4}$$

Donde la *radiacia* $L(\theta)$ es la potencia emitida por unidad de área *dA* en el ángulo sólido $d\Omega$.

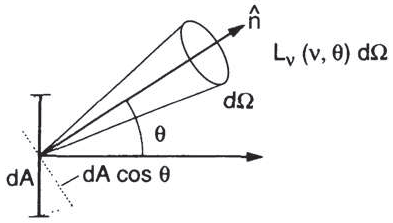


Fig 3.7 Magnitudes radiantes básicas en una fuente de luz. Tomado de [5]

La intensidad se define como la potencia irradiada en una dirección dada (flujo radiante) por unidad de área [28], es decir que

$$I = \frac{dP}{dA} \tag{5}$$

Es imperativo, en este punto, diferenciar entre los elementos de área de la fuente luminosa y los del detector. Supongamos un detector a una distancia *r* de la fuente como se ve en la Fig. 3.8, éste cubre un ángulo sólido $d\Omega = dA' \cos\theta' / r^2$ [28].

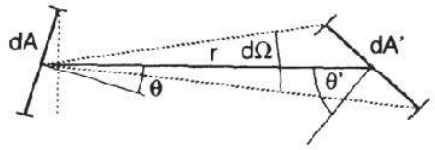


Fig 3.8 Radiancia de una fuente y un detector. dA representa un elemento de área de la fuente mientras dA' representa un elemento de área del detector. Tomado de [5].

Con $r^2 >> dA$, la potencia radiante recibida por *dA'* es [28]

$$dP = L(\theta) dA \cos\theta\, d\Omega = \frac{L(\theta)\cos\theta\, dA \cos\theta'\, dA'}{r^2} \tag{6}$$

Como se puede apreciar en la Fig. 3.8, $cos\theta\, dA$ es la proyección de dA vista por dA'. Para fuentes isotrópicas la ecuación (6) es simétrica al cambio de dA por dA' y $cos\theta$ por $cos\theta'$, por lo que se puede intercambiar entre fuente y detector sin afectar el resultado. Esta intercambiabilidad hace que en la ecuación (6) se pueda interpretar a $L(\theta)$ como la radiancia de la fuente a un ángulo θ desde la normal a la superficie, o como la radiancia incidente en el detector al ángulo θ' [28].

En todo este análisis se han tomado magnitudes referentes la radiación total integrada sobre el espectro completo. Las versiones espectrales de éstas $W_\nu(\nu)$, $P_\nu(\nu)$, $L_\nu(\nu)$, se denominan *densidades espectrales* [28].

Para poder usar las medidas realizadas en el experimento se debe integrar estas densidades entre los valores de longitud de onda en los que trabaja el detector que se utilizó. Además, se debe tomar en cuenta la respuesta del sensor; esto sería hacer la convolución entre la curva de $L\nu$ y la de respuesta del sensor.

Al final, de los datos medidos, podemos tener la potencia que recibe la caja (nuestro sistema) por la radiación en el espectro de 320 a 1100 nm (rango de medida del sensor), de esta manera

$$P = \int_{-\infty}^{+\infty} \int_{x'_0 y'_0}^{x'_f y'_f} \int_{\nu_0}^{\nu_f} \int_{-\pi/2}^{\pi/2} [L(\nu) * S(\nu)] \frac{cos\theta\, dA\, cos\theta'\, dA'\, d\nu\, d\theta}{r^2} \qquad (7)$$

En esta expresión, el área de la fuente se supone infinita por ser la luz solar, la función L es la obtenida mediante las mediciones realizadas, la función S contiene la sensibilidad del sensor frente a las longitudes de onda, el área del detector se supone rectangular con lados $x'_f - x'_0$ e $y'_f - y'_0$, y r es la distancia entre el Sol y la Tierra.

Este valor de potencia, junto con las consideraciones calorimétricas correspondientes, nos avisa cómo va incrementándose la temperatura de nuestro sistema en estudio debido a la incidencia solar. Sin embargo, esta apreciación no está lo suficientemente ajustada debido a varios factores no tomados en cuenta,

como son la incidencia de la radiación a la que el detector no reacciona, la humedad del aire, entre otros.

Ahora analicemos E_e. De la termodinámica tenemos que [29]

$$dE_e = \mu \cdot dn + dW \tag{8}$$

Siendo μ el potencial químico; y W el trabajo realizado por el sistema. El potencial químico es la cantidad de energía que gana o pierde el sistema debido al aumento o disminución de moléculas dn [29]. Además, en este caso específico, no existe una variación del volumen por lo que el trabajo es nulo [29].

Con la definición de potencial químico expresada arriba, éste sería la energía cinética que tiene cada molécula. Esta idea lleva a la conclusión de que la energía perdida (o ganada) por la disminución (o aumento) de dn moléculas en el sistema es

$$dE_e = \frac{3}{2} m \cdot v^2 \cdot dn \tag{9}$$

Donde v es el módulo de la velocidad de una molécula. El factor 3/2 se debe a que el estudio se realiza en las tres dimensiones y la velocidad de la molécula en cada componente es $\frac{1}{2} mv^2$.

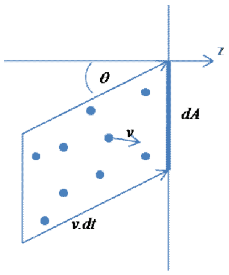


Fig 3.9 La cantidad dn de moléculas de un gas incidentes en una superficie de área dA son aquellas que en un instante dado se encuentran dentro del volumen $dA \cdot v\ dt \cdot \cos\theta$.

Por otro lado, la cantidad de moléculas que salen del recipiente son aquellas que inciden en una superficie dA de la pared, como se describe en la Fig 3.9. Las moléculas incidentes son las contenidas en el volumen $dA \cdot v\ dt \cdot \cos\theta$. En primera aproximación asumimos al aire como un gas ideal, por lo que las velocidades de

sus moléculas cumplen la ley de distribución de Boltzmann, entonces el número de moléculas por unidad de área y por unidad de tiempo que inciden en *dA* es [30]

$$dn \cdot dA \cdot dt = [\int_0^{\infty} \int_0^{\frac{\pi}{2}} \int_0^{2\pi} f(v)\, v\, cos\theta \cdot v^2 dv \cdot\ sen\theta d\theta d\emptyset\,] \cdot dA \cdot dt \qquad (10)$$

Entonces la energía intercambiada con el reservorio se expresaría como

$$dE_E = 2 \cdot \int_0^r \int_0^{2\pi} \left[\frac{3\pi m}{2} \int_0^{\infty} f(v) v^5 dv\right] \cdot r\, dr\, d\vartheta \cdot dt \qquad (11)$$

Donde *r* representa el radio de los agujeros (*r*=2mm); $f(v)$ es la función de distribución de Boltzmann; y ϑ es el ángulo que barre *r* para formar el área de los agujeros (el factor inicial se debe a que son 2 agujeros). Entonces la expresión para E_E queda

$$dE_E = \frac{3\pi}{2} mnr^2 \left(\frac{3k_B T}{m}\right)^{3/2} dt \qquad (12)$$

A modo de resumen diremos que el cambio de energía con el medio externo se produce por la fuga de gas del contenedor dada la presencia de los agujeros en la pared del recipiente, y el motor que regula esta salida de gas es el incremento en la temperatura debido a la irradiación. En otras palabras, cuando aumenta la temperatura dentro del contenedor debería aumentar la presión del gas; pero ésta se mantiene constante porque, en lugar de este cambio, una cantidad de moléculas sale del recipiente.

Entonces la energía media del sistema en un instante dado se puede expresar como

$$\bar{E} = \frac{1}{2} k_B T - \frac{3\pi}{2} mn \left(\frac{3k_B T}{m}\right)^{3/2} + Q \qquad (13)$$

3.3. COMPARACIÓN ENTRE EL SISTEMA TELOS Y EL SISTEMA TRADICIONAL

Para apreciar las bondades de la Red, se puede indicar que con el Sistema Tradicional descrito en el capítulo 3.1.1 se pueden obtener datos muy similares excepto por los de luminosidad, temperatura interna y humedad relativa (el

Sistema Tradicional no incluye fotodiodos ni el sensor empotrado SHT11 del TELOS). Es decir que obtendríamos únicamente datos de temperatura del aire y cantidad de lluvia. Esto nos limita en cuanto a la interpretación realizada en la sección anterior de éste capítulo, acerca de la nubosidad. Y aunque esta información es colectable automáticamente y en tiempo real, no podría fácilmente reunirse sobre espacios relativamente extendidos sin ayuda de largas extensiones de cable.

Cabe recalcar que estos elementos pueden incluirse en un Sistema Tradicional, significando un incremento en el costo del instrumento, así como un incremento en el tiempo de diseño.

De la misma manera, existen algunos detalles que se pueden incluir en el diseño del sistema tradicional para el ahorro de energía. Los dispositivos TELOS han sido desarrollados tomando en cuenta consideraciones de este tipo, reduciendo su consumo energético. Nuevamente, en el sistema tradicional, esto significa un incremento en el costo del instr umento así como en el tiempo de diseño.

En cuanto a costos se refiere, la siguiente tabla ilustra las diferencias entre los dos sistemas (los valores se expresan en dólares americanos).

El cuadro comparativo aquí presentado es referencial, y tiene como finalidad mostrar que el sistema TELOS no solamente tiene mejores características técnicas, sino que también cuesta menos.

	Sistema Tradicional	**Sistema TELOS**
Microprocesador	15	120
Sensores Luminosidad, Temperatura/HR	70	0
Sensores Temperatura, Cantidad de Lluvia	430	430
Circuitería adicional	75	40

Diseño electrónico	130	40
Tiempo hombre de montaje	200	40
Comunicación inalámbrica	70	0
Programación	200	200
Total (USD)	**1190**	**870**

Dentro del contexto de las redes de medición es común la necesidad de ampliar la red, ya sea para aumentar grados de libertad al sistema en estudio; o para hacer redundante al sistema de medida, de manera que descarte información falsa; o mantener un respaldo; o simplemente para cubrir un área física mayor. Esto puede representar un problema si los nodos no mantienen una buena capacidad de integración.

El sistema TELOS posee la capacidad de comunicarse de forma inalámbrica entre dispositivos de su misma clase e inclusive con otros dispositivos inalámbricos, gracias a que maneja el estándar de comunicación IEEE 802.15.4. Aún más interesante resulta la existencia de un algoritmo que permite a un TELOS comunicarse con otro utilizando como puente a un tercero (repetidor). Esta característica se denomina Multihop, y resulta muy útil porque en una red permite que un mensaje llegue desde un nodo hasta otro que está físicamente ubicado a una distancia mayor a su alcance.

El alcance de la comunicación entre dos TELOS es de 125 m con línea de vista, si se utiliza la antena interna del dispositivo. Dentro de un edificio y con la misma antena, el alcance es de 50 m. Si se conecta una antena externa, el alcance puede aumentar hasta los 400 m con línea de vista. Está por demás aclarar que este incremento depende directamente de la antena externa que se emplee.

4. CONCLUSIONES Y TRABAJO FUTURO

Las siguientes conclusiones han resultado del presente trabajo:

- Con este trabajo se ha desarrollado un sistema de medición multiparamétrica que permite la disponibilidad de información en tiempo real en cualquier lugar del Mundo que tenga acceso a Internet. El sistema se traduce como una red de medición de parámetros meteorológicos, que se ha denominado *Red Climatológica Quito*.
- Gracias a su diseño, los dispositivos TELOS pueden acoplarse con facilidad con distintos tipos de sensores. Hemos demostrado que es posible integrar sensores que entregan tanto señales analógicas de diferencia de potencial, señales de intensidad de corriente eléctrica, como información digital, y pulsos electromagnéticos. Un dispositivo TELOS puede funcionar como nodo receptor, con la capacidad de intercambiar información con nodos recolectores y por lo tanto, demuestran ser ideales para su uso en la Red Climatológica Quito.
- Se ha logrado implementar hardware, software, y protocolos para lograr la comunicación inalámbrica y transmisión de datos entre dispositivos TELOS y su integración al ambiente general de la Internet; a través de un prototipo para su utilización en la Red Climatológica Quito. Con esto se ha demostrado la posibilidad de construir una red remota multiparamétrica, inalámbrica, confiable y robusta que permite la disponibilidad de información en tiempo real.
- Para este trabajo se han tenido que desarrollar programas propios adecuados 1) para la programación de los dispositivos TELOS, que se realizó en lenguaje *NesC*, 2) para el condicionamiento y almacenamiento de los datos generados y enviados por los TELOS, lo que se realizó en lenguaje *Java*, y 3) para la generación de una página WEB amigable y eficiente para la presentación e intercambio de la información, lo que se hizo en leguaje *http* (como aplicaciones *JSF* para Java).
- Es posible establecer una red de medición acoplando un nodo receptor con uno o más nodos recolectores.

- Un aporte importante de este trabajo es que los datos obtenidos por la red de medición establecida son accesibles en tiempo real para cualquier persona desde cualquier lugar que tenga acceso a la Internet, a través de una página Web. Actualmente, se accede a esta página a través de la dirección http://157.100.42.162:8080/RCQuito/ .
- Una red de medición de parámetros meteorológicos basada en dispositivos TELOS presenta ventajas frente a una red tradicional en cuanto a la extensión física de la red, al consumo de energía, y al costo, principalmente.
- Este trabajo ha iniciado la construcción de una herramienta de investigación del medio ambiente, que se espera se enriquezca con el aporte de otras personas e instituciones que quieran adherirse a la red con sensores y/o recursos informáticos propios locales.
- El análisis de los resultados obtenidos muestra que es posible extraer conclusiones interesantes a partir de los datos “en crudo” obtenidos por los sensores; y hacer, por ejemplo, la estimación de la nubosidad sobre los puntos de la red por medio del análisis de las señales de los fotodiodos.
- Como parte del trabajo futuro está la implementación de servicios en línea que faciliten el trabajo del investigador; por ejemplo la entrega de reportes periódicos a través de correo electrónico, o la entrega de información estadística a través de la página web.
- Una parte importantísima del trabajo a realizarse es la implementación de más nodos en diferentes partes de la ciudad, del país, o inclusive del Mundo. Con esto se adquiere la capacidad de relacionar la información por regiones obteniéndose información interesante como la estimación del movimiento de las nubes sobre una zona específica de la ciudad.
- Dependiendo del tipo de aplicación que se le pueda dar a esta red, en el futuro se pueden adecuar software, hardware y procesos de modo que sea posible el control de manera remota. Por ejemplo se puede utilizar para captura de imagen, moviendo una cámara ubicada en algún lugar de interés con tan solo presionar un botón en la pantalla del computador.

REFERENCIAS BIBLIOGRÁFICAS

[1] Lagemann Robert. **Ciencia Física Experimental**. Departamento de Física y Astronomía, Universidad Vanderbilt. Editorial Norma. Cali, Colombia. 1968.

[2] Instituto Ecuatoriano de Normalización. **Información Básica sobre el SISTEMA INTERNACIONAL DE UNIDADES, SI**. Edición 2000. http://www.inen.gov.ec/web_sp/si/si.doc

[3] Área de Agrometeorología, Instituto de Recursos Naturales, Colegio de Postgraduados. **Informe sobre Instrumentación Meteorológica**. http://www.colpos.mx/IRENAT/agm/instru/instru.pdf

[4] Corrales Luis. **Sensores y Transductores**. Área de instrumentación y Automatización industriales, Escuela Politécnica Nacional. Quito. 2002.

[5] Acosta Virgilio; Cowan Clydel; Graham B. J. **Curso de Física Moderna.** Editorial TEC-CIEN, LTDA. México D.F. 1975

[6] Universidad Politécnica de Valencia. **La fotosíntesis.** http://www.euita.upv.es/VARIOS/BIOLOGIA/Temas/tema_11.htm#Las%20reacciones%20dependientes%20de%20la%20luz.%20.

[7] Ortiz Ramón; García Alicia; Artiz Mar. **Instrumentación en Volcanología**. Excmo. Cabildo Insular de Lanzarote. 2002.

[8] García Juan. **Conversión de Datos.** Dpto. de Ing. Eléctrica, Electrónica y Automática. Universidad de Castilla. La Mancha. Marzo 2003.

[9] Axelson Jan. **Embedded Ethernet and Internet Complete.** Lakeview Research LLC. Madison. 2003.

[10] Powell James. **Referencia instantánea para Access de Microsoft.** Grupo Noriega Editores. México D.F. 1994

[11] Microchip Technology Inc., **PIC16f870/871 Data Sheet**. http://ww1.microchip.com/downloads/en/DeviceDoc/30569b.pdf . Marzo 2003.

[12] Reyes Carlos. **Microcontroladores PIC Programación en BASIC.** Impreso por RISPERGRAF. Quito. 2006.

[13] Angulo José, Romero Susana, Angulo Ignacio. **Microcontroladores <<PIC>>. Diseño práctico y aplicaciones. Segunda parte: PIC 16F87X.** McGraw Hill. Segunda Edición. Madrid. 2000.

[14] National Semiconductor Co. **LM35 Precision Centigrade Temperature Sensor Data Sheet.** http://www.national.com/ds/LM/LM35.pdf. Noviembre 2000.

[15] Novalynx Corporation. **Model 260-2501M Tipping Bucket Rain Gauge Instruction Manual.** http://www.novalynx.com/manuals/260-2501-manual.pdf. Diciembre 2001.

[16] Moteiv Corporation. **Telos RevB (Low Power Wireless Sensor Module)**. http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf. Mayo 2004.

[17] Sensirion AG. **SHT1x/SHT7x Humidity & Temperature Sensor Data Sheet.**http://www.sensirion.com/en/pdf/product_information/Data_Sheet_humidity_sensor_SHT1x_SHT7x_E.pdf. Julio 2004.

[18] Sensirion AG. **Humidity sensors with CMOSens®.** http://www.sensirion.com/en/04_cmosens_technology/02_humidity.htm

[19] Hamamatsu Photonics K.K. **Si Photodiodes S1087/S1133 Series.** http://sales.hamamatsu.com/assets/pdf/parts_S/S1087_etc.pdf. Abril 2001.

[20] Gay David; Levis Philip; Culler David; Brewer Eric. **nesC 1.1 Language Reference Manual.** http://nescc.sourceforge.net/papers/nesc-ref.pdf. Mayo 2003.

[21] Levis, Philip. **TinyOS Programming.** http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf. Junio 2006.

[22] Giancoli Douglas, Campos Víctor; **Física: Principios con aplicaciones;** Pearson Educación; Sexta edición; 2006.

[23] Diario EL HOY; http://www.hoy.com.ec/noticias-ecuador/inundaciones-y-trafico-por-lluvia-en-quito-317172.html; Edición 2008/11/08.

[24] Aguilar José. **Termodinámica y Mecánica Estadística.** SABER: Entidad Española de Publicaciones Médicas y Científicas. Valencia. 1962.

[25] Giacomelli Gene, Roberts William. Greenhouse Covering Systems.

[26] Ganem Carolina, Esteves Alfredo. **Invernadero Adosado: Tecnología Solar Para Acondicionamiento Térmico de Viviendas y Obtención de Hortalizas y Forrajes en Comunidades de Bajos Recursos.**

[27] Cromer Alan. **Física en la Ciencia y en la Industria.** Reverté, edición ilustrada. Barcelona. 1995.

[28] Demtröder Wolfgang. **Laser Spectroscopy: Basic Concepts and Instrumentation.** Springer-Verlag, edición 3 ilustrada. Berlín. 2003

[29] Reif F. **Fundamentos de Física Estadística y Térmica.** McGraw-Hill Book Company.

[30] Vásconez Freddy. **Apuntes de Clase de Mecánica Estadística.** Profesor: Dr. Edy Ayala. Quito. 2004.

A. CÓDIGO DEL PROGRAMA EN NesC

A.1. COMPONENTE "Sensores.nc"

```
/**
*@autor fvasconez
*@creatOn 2007/06
**/

includes ADC01Msg;      //Contiene la definición del tipo de mensaje
includes ADC01;         //Contiene los parámetros para usar el ADC

configuration Sensores
{
}
//Se definen los components que se van a utilizar
implementation
{
    components Main, SensoresM,
                LedsC,
                TimerC,
                ADCC as SensorC,
                HamamatsuC as PhotoC,
                HumidityC as Humidity,
                MSP430InterruptC as PluvioC,
                InternalVoltageC,
                GenericComm as Gen,
                GenericComm as IDC;

//Wiring: Cómo se enlazan los componentes entre sí
      Main.StdControl               -> SensoresM;
      Main.StdControl               -> SensorC;
      Main.StdControl               -> PhotoC;
      Main.StdControl               -> InternalVoltageC;

      SensoresM.CommControl         -> Gen;
      SensoresM.Comunication        -> IDC;
      SensoresM.Leds                -> LedsC;

      SensoresM.TSR                 -> PhotoC.TSR;
      SensoresM.PAR                 -> PhotoC.PAR;

      SensoresM.Snr                 -> SensorC.ADC[TOS_ADC_SENS_PORT];
      SensoresM.SensorControl       -> SensorC;

      SensoresM.Bat                 -> InternalVoltageC;

      SensoresM.SensT               -> TimerC.Timer[unique("Timer")];
      SensoresM.Send                -> IDC.SendMsg [AM_WEATHMSG];

      SensoresM.Pluvio              -> PluvioC.Port27;
      SensoresM.HumControl          -> Humidity;
      SensoresM.Hum                 -> Humidity.Humidity;
      SensoresM.HumError            -> Humidity.HumidityError;
      SensoresM.TmpError            -> Humidity.TemperatureError;
      SensoresM.Tmp                 -> Humidity.Temperature;

}
```

A.2. COMPONENTE "SensoresM.nc"

```
/**
*
*@autor fvasconez
*@creatOn 2007/06
**/

includes ADC01Msg;      //Contiene la definición del tipo de mensaje
includes ADC01;         //Contiene los parámetros para usar el ADC

//Frecuencia de muestreo en milisegundos (1800000 = 30 min)
#define PLUV_TIME_MS 1800000
//Identificador de la Estación
#define STATION_ID      1

module SensoresM
{
      provides interface StdControl;
      uses interface Leds;
      uses interface Timer as SensT;

      uses interface ADC as Snr;
      uses interface ADC as Tmp;
      uses interface ADC as Hum;
      uses interface ADC as TSR;
      uses interface ADC as PAR;
      uses interface ADC as Bat;

      uses interface ADCControl as SensorControl;
      uses interface SplitControl as HumControl;
      uses interface ADCError as HumError;
      uses interface ADCError as TmpError;

      uses interface MSP430Interrupt as Pluvio;

      uses interface StdControl as CommControl;
      uses interface StdControl as Comunication;
      uses interface SendMsg as Send;
}

implementation

{
//Para definir un 'estado' de toma de datos
      enum{
            HUME,
            TEMP,
            LTSR,
            LPAR,
            BATR,
            PLUV,
            SENS,
      };
      norace int state;
      uint8_t pluv;
      uint8_t j=0;
      uint8_t x=0;
```

```
//Crea una instancia de Mensaje
      struct TOS_Msg  datos;
//Tipo de mensaje: weathMsg
      weathMsg *weather;
//Inicio del sistema
      command result_t StdControl.init(){
            weather=(weathMsg *) datos.data;
            weather->stid=STATION_ID;
            pluv=0;
            state=TEMP;
            call Pluvio.edge(FALSE);
            call Pluvio.clear();
            call HumControl.init();
            call Leds.init();
            call CommControl.init();
            call SensorControl.init();
            call SensorControl.bindPort(TOS_ADC_SENS_PORT,
TOSH_ACTUAL_ADC_SENS_PORT);
            return SUCCESS;   }

//Una vez que se haya iniciado el sensor de HR dispara este evento
      event result_t HumControl.initDone() {
            return SUCCESS;   }

//Iniciado el sistema debe empezar a trabajar
      command result_t StdControl.start(){
            call Leds.redOn();
            call CommControl.start();
            call Pluvio.enable();
            call HumControl.start();
            call SensT.start(TIMER_ONE_SHOT,500);
            return SUCCESS;    }

//El sensor de HR fue puesto a trabajar
      event result_t HumControl.startDone(){
            call HumError.enable();
            call TmpError.enable();
            return SUCCESS;   }

//Para detener el trabajo del sistema llamo a este comando
      command result_t StdControl.stop(){
            call CommControl.stop();
            call HumControl.stop();
            call Pluvio.disable();
            return SUCCESS;   }

//Si se ha detenido el sensor de HR se dispara este evento
      event result_t HumControl.stopDone(){
            call HumError.disable();
            call TmpError.disable();
            return SUCCESS;   }

//Al desbordarse el temporizador verifica el estado
      event result_t SensT.fired(){
            switch(state){
                  case HUME:  call Hum.getData(); break;
                  case TEMP:  call Tmp.getData(); break;
                  case LTSR:  call TSR.getData(); break;
                  case LPAR:  call PAR.getData(); break;
                  case BATR:  call Bat.getData(); break;
```

```
                  case SENS:  call Snr.getData(); break;
                  case PLUV:  call Leds.yellowOn();
                              weather->plvs=pluv;
                              call Send.send(TOS_BCAST_ADDR,
sizeof(weathMsg) , &datos);
                              state=TEMP;
                              call SensT.start(TIMER_ONE_SHOT,
PLUV_TIME_MS); break;
            }
            return SUCCESS;   }

//Cuando un dato del sensor HR está listo lo ubica en el mensaje y cambia
de estado a LTSR
      async event result_t Hum.dataReady(uint16_t data){
            weather->hmdd=data;
            state=LTSR;
            call SensT.start(TIMER_ONE_SHOT, 100);
            return SUCCESS;   }

//Cuando un dato de Temperatura (interna) está listo lo ubica en el
mensaje y cambia de estado a HUME
      async event result_t Tmp.dataReady(uint16_t data){
            weather->tmpr=data;
            state=HUME;
            call SensT.start(TIMER_ONE_SHOT, 100);
            return SUCCESS;   }

//Cuando un dato de luminosidad (completo) está listo lo ubica en el
mensaje y cambia de estado a LPAR
      async event result_t TSR.dataReady(uint16_t data){
            weather->lcmp=data;
            state=LPAR;
            call SensT.start(TIMER_ONE_SHOT, 100);
            return SUCCESS;   }

//Cuando un dato de luminosidad (parcial) está listo lo ubica en el
mensaje y cambia de estado a BATR
      async event result_t PAR.dataReady(uint16_t data){
            weather->lfts=data;
            state=BATR;
            call SensT.start(TIMER_ONE_SHOT, 100);
            return SUCCESS;   }

//Cuando un dato de batería está listo se lo ubica en el mensaje y cambia
de estado a SENS
      async event result_t Bat.dataReady(uint16_t data){
            weather->batr=data;
            state=SENS;
            call SensT.start(TIMER_ONE_SHOT, 100);
            return SUCCESS;   }

//Cuando un dato de Temperatura (ext) está listo lo ubica en el mensaje y
cambia de estado a PLUV
      async event result_t Snr.dataReady(uint16_t data){
            weather->etmp=data;
            call SensT.start(TIMER_ONE_SHOT, 100);
            state=PLUV;
            return SUCCESS;   }

//Cuando hay un pulso del pluviómetro se aumenta en el contador
```

```
      async event void Pluvio.fired(){
            pluv++;
            call Leds.greenToggle();
            call Pluvio.clear();    }

//Si hay un error al tomar la medida del sensor HR haga esto
      event result_t HumError.error(uint8_t token){
            weather->hmdd=1;
            state=LTSR;
            return SUCCESS;   }

//Si hay un error al tomar la medida de Temperatua (interna)
      event result_t TmpError.error(uint8_t token){
            weather->tmpr=1;
            state=HUME;
            return SUCCESS;   }

//Una vez que se ha enviado el mensaje haga esto
      event result_t Send.sendDone(TOS_MsgPtr m, result_t success){
            pluv=0;
            call Leds.yellowOff();
            return SUCCESS;   }
}
```

A.3. LIBRERÍA "ADC01Msg.h"

```
/**
*
*@autor fvasconez
*@creatOn 2007/06
*
* Esta librería contiene la definición del mensaje para parámetros
* meteorológicos que ha de usar el TELOS
**/

enum {    AM_WEATHMSG  = 22   };

typedef struct weathMsg {//15 bytes
  uint8_t stid;                    //1
  uint16_t tmpr;             //2
  uint16_t hmdd;             //2
  uint16_t lcmp;             //2
  uint16_t lfts;             //2
  uint16_t batr;             //2
  uint16_t etmp;             //2
  uint16_t plvs;             //2
} weathMsg;
```

B. CÓDIGO DEL PROGRAMA EN JAVA

B.1. APLICACIÓN "RecibeTelos.java"

```
/*
 * RecibeTelos.java
 *
 * Created on 3 de marzo de 2006, 02:19 PM
 * @author  fvasconez
 *
 */

import java.sql.*;
import java.io.*;
import java.util.*;
import javax.comm.*;
import java.awt.*;

import net.tinyos.message.Message;
import net.tinyos.message.MessageListener;
import net.tinyos.message.MoteIF;
import net.tinyos.packet.BuildSource;

public class RecibeTelos extends javax.swing.JFrame implements
MessageListener {

      MoteIF mf;
    boolean connected=false;
    boolean base=false;
    int contador=0;
    static CommPortIdentifier portId;
    static Enumeration portList;
    static OutputStream outputStream;

    private Connection connection;
    private ResultSet resultset;
    private Statement statement;
    private int result;

    int bandera=0;
    int Count=0;
    int inicio=0;
    int i=0, j=0;
    int dataFlag=0;
    int pppCounter=0;
    int tempraw_1=0,tempraw_2=0;
    double temp_1=0, temp_2=0;
    boolean connectionFlag=false;
    boolean isBatShowed=false;

    String Puerto = new String("COM2");
    String FCHA = new String("");
    java.util.Calendar fecha;

    public int[] readBuffer = new int[100];

   TOSMSG data;

    /** Creates new form RecibeTelos */
```

```
    public RecibeTelos() {
        this.setBounds(100, 100, 600, 600);

        initComponents();
    }


    private void initComponents() {
        jPanel1 = new javax.swing.JPanel();
        ConectarDesconectar_Button = new javax.swing.JButton();
        Salir_Button = new javax.swing.JButton();
        jPanel2 = new javax.swing.JPanel();
        MainMsg_Label = new javax.swing.JLabel();
        SecondMsg_Label = new javax.swing.JLabel();
        Menu_MenuBar = new javax.swing.JMenuBar();
        Archivo_Menu = new javax.swing.JMenu();
        Salir_MenuItem = new javax.swing.JMenuItem();
        Puerto_Menu = new javax.swing.JMenu();
        COM2_RadioButtonMenuItem = new
javax.swing.JRadioButtonMenuItem();
        COM3_RadioButtonMenuItem = new
javax.swing.JRadioButtonMenuItem();
        COM18_RadioButtonMenuItem = new
javax.swing.JRadioButtonMenuItem();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Ingreso de datos: Clima de Quito");
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setMaximizedBounds(new java.awt.Rectangle(0, 0, 60, 60));
        jPanel1.setLayout(null);

        jPanel1.setPreferredSize(new java.awt.Dimension(350, 120));
        ConectarDesconectar_Button.setText("Conectar");
        ConectarDesconectar_Button.setEnabled(false);
        ConectarDesconectar_Button.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ConectarDesconectar_ButtonActionPerformed(evt);
            }
        });

        jPanel1.add(ConectarDesconectar_Button);
        ConectarDesconectar_Button.setBounds(30, 10, 130, 30);

        Salir_Button.setText("Salir");
        Salir_Button.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Salir_ButtonActionPerformed(evt);
            }
        });

        jPanel1.add(Salir_Button);
        Salir_Button.setBounds(200, 10, 120, 30);

        getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);

        jPanel2.setLayout(new java.awt.GridLayout(1, 0));
```

```
MainMsg_Label.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        MainMsg_Label.setText("DESCONECTADO:  ");
        MainMsg_Label.setMaximumSize(new java.awt.Dimension(80, 19));
        MainMsg_Label.setMinimumSize(new java.awt.Dimension(80, 19));
        MainMsg_Label.setPreferredSize(new java.awt.Dimension(80, 19));
        jPanel2.add(MainMsg_Label);

        SecondMsg_Label.setText("Elija el Puerto de conexi\u00f3n");
        jPanel2.add(SecondMsg_Label);

        getContentPane().add(jPanel2, java.awt.BorderLayout.SOUTH);

        Archivo_Menu.setText("Archivo");
        Salir_MenuItem.setText("Salir");
        Salir_MenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                Salir_MenuItemActionPerformed(evt);
            }
        });
        Salir_MenuItem.addMouseListener(new java.awt.event.MouseAdapter()
{
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                Salir_MenuItemMouseClicked(evt);
            }
        });

        Archivo_Menu.add(Salir_MenuItem);

        Menu_MenuBar.add(Archivo_Menu);

        Puerto_Menu.setText("Puerto");
        COM2_RadioButtonMenuItem.setText("COM2");
        COM2_RadioButtonMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                COM2_RadioButtonMenuItemActionPerformed(evt);
            }
        });

        Puerto_Menu.add(COM2_RadioButtonMenuItem);

        COM3_RadioButtonMenuItem.setText("COM3");
        COM3_RadioButtonMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                COM3_RadioButtonMenuItemActionPerformed(evt);
            }
        });

        Puerto_Menu.add(COM3_RadioButtonMenuItem);

        COM18_RadioButtonMenuItem.setText("COM18");
        COM18_RadioButtonMenuItem.setActionCommand("COM18");
        COM18_RadioButtonMenuItem.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                COM18_RadioButtonMenuItemActionPerformed(evt);
            }
```

```
        });

        Puerto_Menu.add(COM18_RadioButtonMenuItem);

        Menu_MenuBar.add(Puerto_Menu);

        setJMenuBar(Menu_MenuBar);

        pack();
    }

    public boolean connect(){
        try{
            Class.forName("com.mysql.jdbc.Driver");

connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/clima
bdd","root","");
            return true;
        }catch(ClassNotFoundException e){System.out.println("Class Not
Found Exception");}
         catch(SQLException e){System.out.println("SQL Exception1");}
        return false;
    }

    private void
COM2_RadioButtonMenuItemActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_COM7_RadioButtonMenuItemActionPerformed
        Puerto="COM2";
        ConectarDesconectar_Button.setEnabled(true);
        COM3_RadioButtonMenuItem.setSelected(false);
        COM18_RadioButtonMenuItem.setSelected(false);
    }//GEN-LAST:event_COM7_RadioButtonMenuItemActionPerformed

    private void
COM3_RadioButtonMenuItemActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_COM3_RadioButtonMenuItemActionPerformed
        Puerto="COM3";
        ConectarDesconectar_Button.setEnabled(true);
        COM2_RadioButtonMenuItem.setSelected(false);
        COM18_RadioButtonMenuItem.setSelected(false);
    }//GEN-LAST:event_COM3_RadioButtonMenuItemActionPerformed

    private void
COM18_RadioButtonMenuItemActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_COM4_RadioButtonMenuItemActionPerformed
        Puerto="COM18";
        ConectarDesconectar_Button.setEnabled(true);
        COM2_RadioButtonMenuItem.setSelected(false);
        COM3_RadioButtonMenuItem.setSelected(false);
    }//GEN-LAST:event_COM4_RadioButtonMenuItemActionPerformed

    private void Salir_MenuItemActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_Salir_MenuItemActionPerformed
        if(connected) serialPort.close();
        System.exit(0);
    }//GEN-LAST:event_Salir_MenuItemActionPerformed

    private void Salir_MenuItemMouseClicked(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_Salir_MenuItemMouseClicked
        if(connected) serialPort.close();
```

```
        System.exit(0);
    }//GEN-LAST:event_Salir_MenuItemMouseClicked

    private void Salir_ButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_Salir_ButtonActionPerformed
        if(connected) serialPort.close();
        System.exit(0);
    }//GEN-LAST:event_Salir_ButtonActionPerformed

    private void
ConectarDesconectar_ButtonActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_ConectarDesconectar_ButtonActionPerformed
     if(connect()==false){
            System.out.println("No se ha conectado con la Base de
Datos");
            MainMsg_Label.setText("Error al conectar base de datos");
        }
        else{
        System.out.println("Conectado");
        ConectarDesconectar_Button.setText("Desconectar");
        base=true;
        }

     mf=new
MoteIF(BuildSource.makePhoenix("serial@"+Puerto+":telos",net.tinyos.util.
PrintStreamMessenger.err));
       mf.registerListener(new weathMsg(),this);


    }//GEN-LAST:event_ConectarDesconectar_ButtonActionPerformed


    public void messageReceived(int toaddr, Message msg){
      if(msg.amType()==weathMsg.AM_TYPE){
            weathMsg wmsg= (weathMsg)msg;
            try{
              statement=connection.createStatement();
                  fecha=Calendar.getInstance();
                  FCHA=fecha.get(Calendar.YEAR)+"-"+
                        +(fecha.get(Calendar.MONTH)+1)+"-"+
                        +fecha.get(Calendar.DAY_OF_MONTH)+" "+
                        +fecha.get(Calendar.HOUR_OF_DAY)+":"+
                        +fecha.get(Calendar.MINUTE)+":"+
                        +fecha.get(Calendar.SECOND);
                  String sql="insert into
rdts(IDNE,HMDD,TMPR,LTSR,LPAR,BATR,ETMP,PLVS,FCHA)
values("+wmsg.get_stid()+","+

      +wmsg.get_hmdd()+","+wmsg.get_tmpr()+","+wmsg.get_lcmp()+","+wmsg.g
et_lfts()+","+

      +wmsg.get_batr()+","+wmsg.get_etmp()+","+wmsg.get_plvs()+",'"+FCHA+
"')";
                  System.out.println(sql);
                  result=statement.executeUpdate(sql);
            }catch(SQLException e){System.out.println("SQL Exception2");}
      }
    }
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new RecibeTelos().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JMenu Archivo_Menu;
    private javax.swing.JRadioButtonMenuItem COM2_RadioButtonMenuItem;
    private javax.swing.JRadioButtonMenuItem COM3_RadioButtonMenuItem;
    private javax.swing.JRadioButtonMenuItem COM18_RadioButtonMenuItem;
    private javax.swing.JButton ConectarDesconectar_Button;
    private javax.swing.JLabel MainMsg_Label;
    private javax.swing.JMenuBar Menu_MenuBar;
    private javax.swing.JMenu Puerto_Menu;
    private javax.swing.JButton Salir_Button;
    private javax.swing.JMenuItem Salir_MenuItem;
    private javax.swing.JLabel SecondMsg_Label;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    // End of variables declaration//GEN-END:variables


}
```

C. CÓDIGO DE LA APLICACIÓN JSF “RCQuito”

Esta aplicación consta de 8 páginas, cada una tiene una parte jsp y una parte Java. Además contiene ‘beans’, que son pedazos de código Java que interactúan entre sí. A continuación se presenta el código de las páginas más importantes y de los beans necesarios para que la aplicación funcione adecuadamente.

C.1. ApplicationBean1.java

```
/*
 * ApplicationBean1.java
 *
 * Created on 06/04/2008, 04:24:33 PM
 *
 */


package rcquito;

import com.sun.rave.web.ui.appbase.AbstractApplicationBean;
import java.util.Date;
import javax.faces.FacesException;
```

```
/*
 *
 * @author fvasconez
 *
 */

public class ApplicationBean1 extends AbstractApplicationBean {

    private void _init() throws Exception {
    }

    public ApplicationBean1() {
    }

    @Override
    public void init() {
        super.init();
        try {
            _init();
        } catch (Exception e) {
            log("ApplicationBean1 Initialization Failure", e);
            throw e instanceof FacesException ? (FacesException) e: new
FacesException(e);
        }

    }

    @Override
    public void destroy() {
    }

    @Override
    public String getLocaleCharacterEncoding() {
        return super.getLocaleCharacterEncoding();
    }

    private String dscr = "";

    public void setDscr(String newDscr){
        dscr = newDscr;
    }

    public String getDscr(){
        return dscr;
    }

    private String mnsj = "";

    public void setMnsj(String newMnsj){
        mnsj = newMnsj;
    }

    public String getMnsj(){
        return mnsj;
    }

    private String icono= "/resources/encab/icoEr.png";

    public String getIcono(){
```

```
        return icono;
    }

    public void setIcono(String newIcono){
        icono = newIcono;
    }
}
```

C.2. Bdatos.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package rcquito;

/**
 *
 * @author fvasconez
 */
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.faces.context.FacesContext;
import javax.servlet.*;

public class Bdatos implements Serializable {

    /** Creates a new instance of Bdatos */
    public Bdatos() {

    }

    private boolean conBdd(String URL, String usr, String psw) throws
ClassNotFoundException, SQLException {
        if (debug) {
            System.out.println("Intentando conectar con la BDD");
        }
        Class.forName("com.mysql.jdbc.Driver");
        if(local){URL = "//localhost:3306/climabdd";
                  usr = "root";
                  psw = "";}
        connection = DriverManager.getConnection("jdbc:mysql:" + URL,
usr, psw);
        statement = connection.createStatement();
        baseConectada = true;
        if (debug) {
            System.out.println("BDD en lÃnea");
        }
        return baseConectada;
    }

    public String[] getActual() {
        ResultSet resultSet;
        String[] actual = new String[10];
        String ultContador = "";
        if (debug) {
```

```
            System.out.println("Iniciando la recuperaciÃ³n del Ã°ltimo
dato ingresado");
        }
        try {
            if (conBdd(bddURL, Usr, Pwd)) {
                resultSet = statement.executeQuery("select max(CNTD) from
rdts");
                resultSet.next();
                ultContador = resultSet.getString("max(CNTD)");
                resultSet = statement.executeQuery("select * from rdts
where CNTD='" + ultContador + "'");
                resultSet.next();
                actual[0] = "" + resultSet.getDate("FCHA");
                actual[1] = "" + resultSet.getTime("FCHA");
                actual[2] = "" + (int) ((resultSet.getFloat("TMPR") -
4500) / 110);
                actual[3] = "" + (int) ((resultSet.getInt("HMDD") - 16) /
26);
                actual[4] = resultSet.getString("LTSR");
                actual[5] = resultSet.getString("PLVS");
                actual[6] = resultSet.getString("LPAR");
                actual[7] = ""+ (resultSet.getInt("ETMP")/41-12);
                resultSet.close();
            }
        } catch (ClassNotFoundException cnfx) {
            System.out.println("No se pudo conectar con la Base de Datos:
" + cnfx);
        } catch (SQLException sqlx) {
            System.out.println("Error en la Base de Datos: " + sqlx);
        }
        if (debug) {
            System.out.println("Actual recuperado!");
        }
        return actual;
    }

    private boolean recogerInfo(boolean[] escogido) {
        boolean exito = false;
        float[] buffer = {205,205,205};
        if (debug) {
            System.out.println("Iniciando la recuperaciÃ³n principal de
datos");
        }
        try {
            if (conBdd(bddURL, Usr, Pwd)) {
                ResultSet resultSet;
                String sensores = "BATR";
                if (escogido[0]) {
                    sensores += ",TMPR,ETMP";
                    csens += 2;
                }
                if (escogido[1]) {
                    sensores += ",HMDD";
                    csens++;
                }
                if (escogido[2]) {
                    sensores += ",LTSR,LPAR";
                    csens += 2;
                }
                if (escogido[3]) {
```

```
                    sensores += ",PLVS";
                    csens++;
                }

                String comando = "select " + sensores + ",CNTD,FCHA from
" + tabla + " where FCHA between '" + getFcha_ini() + "' and '" +
getFcha_fin() + "'";
                if (debug) {
                    System.out.println("Query:  " + comando);
                }
                resultSet = statement.executeQuery(comando);
                //Con esto 'j' tendrÃ¡ el contador del Ãºltimo dato
                resultSet.afterLast();
                if (resultSet.last()) {
                    j = resultSet.getInt("CNTD");
                    setEtiqdos(resultSet.getString("FCHA"));
                }
                resultSet.beforeFirst();
                resultSet.next();
                numMuestras = j + 1 - resultSet.getInt("CNTD");
                setEtiquno(resultSet.getString("FCHA"));
                resultSet.afterLast();
                //InicializaciÃ³n de variables
                Data = new float[7][numMuestras];
                Fcha = new String[numMuestras];
                Hora = new String[numMuestras];
                //Recoge las muestras entre las dos fechas fijadas, con
un mÃ¡ximo de MaxSamples
                N = 0;
                while (resultSet.previous() && cntd < MaxSamples) {
                    //Recoge la Fecha y la hora de cada registro
                    Fcha[cntd] = "" + resultSet.getDate("FCHA");
                    Hora[cntd] = "" + resultSet.getTime("FCHA");
                    //Recoge el valor dado del sensor discriminando el
tipo de datos en cada caso
                    for (int i = 0; i < 7; i++) {
                        switch (i) {
                            case 0:
                                if (escogido[1]) {
                                    Data[H][cntd] =
((resultSet.getInt("HMDD") - 16) / 26);
                                    dat[H] += resultSet.getString("HMDD")
+ " ";
                                }
                                //addTotal(Dat[cntd]);
                                break;
                            case 1:
                                if (escogido[0]) {
                                    Data[T][cntd] =
(resultSet.getFloat("TMPR") - 4000) / 100;
                                    dat[T] += resultSet.getString("TMPR")
+ " ";
                                }
                                //addTotal(DatT[cntd]);
                                break;
                            case 2:
                                if (escogido[2]) {
                                    Data[L][cntd] =
resultSet.getFloat("LTSR");
```

```
                                    dat[L] += resultSet.getString("LTSR")
+ " ";
                                }
                                //addTotal(DatL[cntd]);
                                break;
                            case 3:
                                if (escogido[2]) {
                                    Data[L3][cntd] =
resultSet.getFloat("LPAR");
                                    dat[L3] +=
resultSet.getString("LPAR") + " ";
                                }
                                break;
                            case 4:
                                Data[B][cntd] =
resultSet.getFloat("BATR");
                                dat[B] += resultSet.getString("BATR") + "
";
                                break;
                            case 5:
                                if (escogido[0]) {

                                    Data[T5][cntd] =
(resultSet.getFloat("ETMP")* 3) / 41;
                                    dat[T5] +=
resultSet.getString("ETMP") + " ";
                                }
                                break;
                            case 6:
                                if (escogido[3]) {
                                    Data[P][cntd] =
resultSet.getFloat("PLVS");
                                    dat[P] += resultSet.getString("PLVS")
+ " ";
                                }
                                //addTotal(Dat[cntd]);
                                break;
                        }
                    }
                    addN();
                    cntd++;
                }
                cntd = 0;
                if (debug) {
                    System.out.println("NÃºmero de registros: " + N);
                }
                resultSet.close();
                exito = true;
            }
            if (debug) {
                System.out.println("Se ha recolectado la informaciÃ³n
requerida de la BDD");
            }
        } catch (ClassNotFoundException cnfx) {
            System.out.println("No se pudo conectar con la Base de Datos:
" + cnfx);
        } catch (SQLException sqlx) {
            System.out.println("Error en la Base de Datos: " + sqlx);
        }
```

```
        if(!genArchivo(escogido))exito=false;
        setEstadistica(Data);
        return exito;
    }

    public String[] Dia() {
        Calendar cal = Calendar.getInstance();
        String[] fechas = {"",""};
        int dia,mes,anio,hora,min;
        anio = cal.get(Calendar.YEAR);
        mes = (cal.get(Calendar.MONTH)+1);
        dia = cal.get(Calendar.DATE);
        hora = cal.get(Calendar.HOUR_OF_DAY);
        min = cal.get(Calendar.MINUTE);
        fechas[1]=""+anio+"/"+mes+"/"+dia+" "+hora+":"+min+":00";
        if(dia==1){
            if(mes==3){
                mes=2;
                if ((anio % 4 == 0) && ((anio % 100 != 0) || (anio % 400
== 0)))dia=29;
                else {dia=28;}
            }else if(mes==2||mes==4||mes==6||mes==8||mes==9||mes==11){
                mes--;
                dia=31;
            }else if(mes==5||mes==7||mes==10||mes==12){
                mes--;
                dia=30;
            }else if(mes==1){
                anio--;
                mes=12;
                dia=31;
            }
        }else dia--;
        fechas[0]=""+anio+"/"+mes+"/"+dia+" "+hora+":"+min+":00";
        setFcha_ini(fechas[0]);
        setFcha_fin(fechas[1]);

        return fechas;
    }

    public boolean recogeDia(){
        boolean exito = false;
        boolean[] iniciales = {true, true, false, false};
        Dia();
        if(recogerInfo(iniciales))exito=true;
        return exito;
    }

    public boolean genArchivo(boolean[] escogido) {
        try {
            //Escribe los datos en un archivo
            FacesContext fcxt = FacesContext.getCurrentInstance();
            ServletContext scxt = (ServletContext)
fcxt.getExternalContext().getContext();
            String RutaArchivo = (String)
scxt.getAttribute("RUTA_ARCHIVO");
            setSufijo(RutaArchivo + "Reporte.xls");
            File arch = new File(getSufijo());

            if (arch.exists()) {
```

```
                arch.delete();
                arch.createNewFile();
            } else {
                arch.createNewFile();
            }
            if (debug) {
                System.out.print("Iniciando la generaciÃ³n del archivo");
            }
            fos = new FileOutputStream(arch);
            dos = new DataOutputStream(fos);
            if (debug) {
                System.out.println("Cantidad de Sensores: " +
getCsens());
            }
            dos.writeBytes("<table border=\"2\" colspan=\"2\"><tr><td
colspan=\"" + (getCsens() + 1) + "\"align=\"center\"><b>ParÃ¡metros
MeteorolÃ³gicos en Quito</b></td></tr>\n");
            dos.writeBytes("<tr><td align=\"center\"><b>Fecha</b></td>");
            dos.writeBytes("<td align=\"center\"><b>Hora</b></td>");
            if (escogido[0]) {
                dos.writeBytes("<td align=\"center\"><b>Temperatura
[Â°C]</b></td>");
                dos.writeBytes("<td align=\"center\"><b>Temp. (sensor
externo) [Â°C]</b></td>");
            }
            if (escogido[1]) {
                dos.writeBytes("<td align=\"center\"><b>Humedad Relativa
[%]</b></td>");
            }
            if (escogido[2]) {
                dos.writeBytes("<td align=\"center\"><b>Luminosidad TSR*
[*10^9 lux]</b></td>");
                dos.writeBytes("<td align=\"center\"><b>Luminosidad PAR**
[*10^8 lux]</b></td>");
            }
            if (escogido[3]) {
                dos.writeBytes("<td align=\"center\"><b>Cantidad de
Lluvia [mm]</b></td>");
            }
            dos.writeBytes("</tr>");
            for (int i = N - 1; i >= 0; i--) {            //for(int
i=DatT.length-1;i>=0;i--){
                dos.writeBytes("<tr><td align=\"center\">" + Fcha[i] +
"</td><td align=\"center\">" + Hora[i] + "</td>");
                if (escogido[0]) {
                    dos.writeBytes("<td align=\"center\">" + Data[T][i] +
"</td>");
                    dos.writeBytes("<td align=\"center\">" + Data[T5][i]
+ "</td>");
                }
                if (escogido[1]) {
                    dos.writeBytes("<td align=\"center\">" + Data[H][i] +
"</td>");
                }
                if (escogido[2]) {
                    dos.writeBytes("<td align=\"center\">" + Data[L][i] +
"</td>");
                    dos.writeBytes("<td align=\"center\">" + Data[L3][i]
+ "</td>");
                }
```

```
                if (escogido[3]) {
                    dos.writeBytes("<td align=\"center\">" + Data[P][i] +
"</td>");
                }
                dos.writeBytes("</tr>");
            }
            if (escogido[2]) {
                dos.writeBytes("<tr><td colspan=\"" + (getCsens() + 1) +
"\">* Sensible al espectro visible completo.\n");
                dos.writeBytes("<br>** Sensible al espectro de absorciÃ³n
fotosintÃ©tica.</td></tr>");
            }
            dos.writeBytes("</table>\n");

            dos.close();
            csens = 1;
            if (debug) {
                System.out.println("Archivo 'Reporte.xls' generado
exitosamente");
            }
            return true;
        } catch (FileNotFoundException fnfx) {
            System.out.println("Error de archivo " + fnfx);
            return false;
        } catch (IOException iofx) {
            System.out.println("Error de archivo " + iofx);
            return false;
        }
    }

    public boolean[] usuarioCorrecto(String nbUsuario, String contrase){
        boolean [] correcto = {false,false};
        ResultSet resultSet;
        try{
            if(conBdd(bddURL, Usr, Pwd)){
                resultSet = statement.executeQuery("select CNTR from usrs
where NUSR='"+nbUsuario+"'");
                correcto[0]=resultSet.next();
                if(correcto[0] &&
resultSet.getString("CNTR").equals(contrase))correcto[1]=true;
                resultSet.close();
            }
        }catch(SQLException sqlx){
            System.out.println("ValidaciÃ³n de Usuario: Error al intentar
conectar con la base de datos: "+sqlx);
        }catch(ClassNotFoundException cnfx){
            System.out.println("ValidaciÃ³n de Usuario: ConexiÃ³n con bdd
fallida: "+cnfx);
        }
        return correcto;
    }

    public String[] infoUsuario(String nombre){
        String[] info = new String[7];
        try{
            if(conBdd(bddURL, Usr, Pwd)){
                ResultSet rs;
                rs = statement.executeQuery("select NMBR, APLL, EDRC,
PRSC, RSPT, CNTR from usrs where NUSR='"+nombre+"'");
                if(rs.next()){
```

```
                    info[0]=rs.getString("NMBR");
                    info[1]=rs.getString("APLL");
                    info[2]=nombre;
                    info[3]=rs.getString("EDRC");
                    info[4]=rs.getString("PRSC");
                    info[5]=rs.getString("RSPT");
                    info[6]=rs.getString("CNTR");
                    System.out.println("Para probar: "+info[6]);
                }
            }
        }catch(ClassNotFoundException cnfx){
            System.out.println("Error en la carga de la informaciÃ³n del
usuario registrado: "+cnfx);
        }catch(SQLException sqlx){
            System.out.println("Error en la carga de la info del usuario:
"+sqlx);
        }
        return info;
    }

    public boolean registrarUsuario(String[] usr) {
        boolean exito = false;
        try{
            if(conBdd(bddURL, Usr, Pwd)){
                ResultSet resultSet;
                resultSet = statement.executeQuery("insert into usrs
(NMBR,APLL,EDRC,NUSR,CNTR,DRC1,DRC2,TLFN,TRBJ,PRSC,RSPT) values
("+usr[0]+","+usr[1]+","+usr[2]+","+usr[3]+","+usr[4]+","+usr[5]+","+usr[
6]+","+usr[7]+","+usr[8]+","+usr[9]+","+usr[10]+")");
                if(resultSet!=null)exito=true;
            }
        }catch(ClassNotFoundException cnfx){
            System.out.println("Error al registrar ususario: "+cnfx);
        }catch(SQLException sqlx){
            System.out.println("Error al registrar usuario: "+sqlx);
        }
        return exito;
    }

    public void setFcha_ini(String newFecha) {
        fcha_ini = newFecha;
    }

    public String getFcha_ini() {
        return fcha_ini;
    }

    public void setFcha_fin(String newFecha) {
        fcha_fin = newFecha;
    }

    public String getFcha_fin() {
        return fcha_fin;
    }

    public int getCsens() {
        return csens;
    }

    public void setParametros(boolean[] newParam) {
```

```
        parametros = newParam;
        recogerInfo(getParametros());
    }

    public boolean[] getParametros() {
        return parametros;
    }

    public String[] getDat() {
        if(debug){
            System.out.print("Trama: ");
            for(int f=0;f<dat.length;f++){
                System.out.print(dat[f]+" ");
            }
            System.out.println(";fin de trama");
        }
        return dat;
    }

    public void resetData() {
        for (int x = 0; x < 7; x++) {
            dat[x] = "";
        }
    }

    private void setEstadistica(float muestra[][]) {
        if (N != 0) {
            for(int i=0;i<7;i++){
                total = 0;
                for(int k=0;k<N;k++){
                    total+=muestra[i][k];
                }
                estadistica[0][i]=total/N;

estadistica[0][i]=(float)(Math.rint(estadistica[0][i]*10))/10;
            }
            for(int i=0;i<7;i++){
                total = 0;
                for(int k=0;k<N;k++){
                    total+=(muestra[i][k]-
estadistica[0][i])*(muestra[i][k]-estadistica[0][i]);
                }
                estadistica[1][i]=total/(N-1);
                estadistica[1][i]=(float)Math.sqrt(estadistica[1][i]);

estadistica[1][i]=(float)(Math.rint(estadistica[1][i]*10)/10);
            }
        } else {
            for(int i=0;i<7;i++){
            estadistica[0][i] = 0;
            estadistica[1][i] = 0;
            }
        }
        total = 0;
        N = 0;
    }

    public float[][] getEstadistica(){
        return estadistica;
    }
```

```
public void addN() {
    N++;
}

private void setSufijo(String newSufix) {
    sufijo = newSufix;
}

public String getSufijo() {
    return sufijo;
}

public void setEtiquno(String newEtiqueta){
    etiquno = newEtiqueta;
}

public void setEtiqdos(String newEtiqueta){
    etiqdos = newEtiqueta;
}

public String getEtiquno(){
    return etiquno;
}

public String getEtiqdos(){
    return etiqdos;
}

//Propiedades del Bean
private String fcha_ini = new String();
private String fcha_fin = new String();
private int csens = 1;
private boolean[] parametros = {true, false, false, false};
private String etiquno = "";
private String etiqdos = "";
//Datos obtenidos
private String[] dat = {"0","0","0","0","0","0","0"};
private float[][] Data;
//variables de clase
private String[] Fcha,  Hora;
//variables para el manejo de la base de datos (BDD)
private Statement statement;
private Connection connection;
private boolean baseConectada = false;
private String tabla = "rdts";
private int numMuestras = 15,  j = 0,  cntd = 0;
private String bddURL = "//69.65.157.50:3306/climabdd";
private String Usr = "mysqlig";
private String Pwd = "igepn";
//variables para el manejo de archivos
private FileOutputStream fos;
private DataOutputStream dos;
private String sufijo;
//variables de apoyo
private float[][] estadistica = new float[2][7];
private float total = 0;
private int N = 0;
//definiciones
```

```
    private final int H = 0,  T = 1,  T5 = 5,  L = 2,  L3 = 3,  B = 4,  P
= 6;
    private final int MaxSamples = 1500;
    private final boolean debug = false, local = false;

}
```

C.3. Inicio.java

```
/*
 * Inicio.java
 *
 * Created on 06/04/2008, 04:24:33 PM
 */

package rcquito;

import com.sun.rave.web.ui.appbase.AbstractPageBean;
import com.sun.webui.jsf.component.Body;
import com.sun.webui.jsf.component.Button;
import com.sun.webui.jsf.component.Form;
import com.sun.webui.jsf.component.Head;
import com.sun.webui.jsf.component.Html;
import com.sun.webui.jsf.component.ImageComponent;
import com.sun.webui.jsf.component.Link;
import com.sun.webui.jsf.component.Page;
import com.sun.webui.jsf.component.PageSeparator;
import com.sun.webui.jsf.component.PanelGroup;
import com.sun.webui.jsf.component.StaticText;
import java.util.EventObject;
import javax.faces.FacesException;

/**
 *
 * @author fvasconez
 */
public class Inicio extends AbstractPageBean implements CambioListener{

    private void _init() throws Exception {
    }

    private Page page1 = new Page();

    public Page getPage1() {
        return page1;
    }

    public void setPage1(Page p) {
        this.page1 = p;
    }

    private Html html1 = new Html();

    public Html getHtml1() {
        return html1;
    }
```

```
public void setHtml1(Html h) {
    this.html1 = h;
}

private Head head1 = new Head();

public Head getHead1() {
    return head1;
}

public void setHead1(Head h) {
    this.head1 = h;
}

private Link link1 = new Link();

public Link getLink1() {
    return link1;
}

public void setLink1(Link l) {
    this.link1 = l;
}

private Body body1 = new Body();

public Body getBody1() {
    return body1;
}

public void setBody1(Body b) {
    this.body1 = b;
}

private Form form1 = new Form();

public Form getForm1() {
    return form1;
}

public void setForm1(Form f) {
    this.form1 = f;
}
private PageSeparator pageSeparator1 = new PageSeparator();

public PageSeparator getPageSeparator1() {
    return pageSeparator1;
}

public void setPageSeparator1(PageSeparator ps) {
    this.pageSeparator1 = ps;
}
private StaticText ult_temperatura = new StaticText();

public StaticText getUlt_temperatura() {
    return ult_temperatura;
}

public void setUlt_temperatura(StaticText st) {
```

```
    this.ult_temperatura = st;
}
private StaticText ult_humedad = new StaticText();

public StaticText getUlt_humedad() {
    return ult_humedad;
}

public void setUlt_humedad(StaticText st) {
    this.ult_humedad = st;
}
private StaticText mensaje = new StaticText();

public StaticText getMensaje() {
    return mensaje;
}

public void setMensaje(StaticText st) {
    this.mensaje = st;
}
private PanelGroup groupPanel1 = new PanelGroup();

public PanelGroup getGroupPanel1() {
    return groupPanel1;
}

public void setGroupPanel1(PanelGroup pg) {
    this.groupPanel1 = pg;
}
private PanelGroup groupPanel2 = new PanelGroup();

public PanelGroup getGroupPanel2() {
    return groupPanel2;
}

public void setGroupPanel2(PanelGroup pg) {
    this.groupPanel2 = pg;
}
private StaticText ultRegistro1 = new StaticText();

public StaticText getUltRegistro1() {
    return ultRegistro1;
}

public void setUltRegistro1(StaticText st) {
    this.ultRegistro1 = st;
}
private StaticText bienvenido_txt = new StaticText();

public StaticText getBienvenido_txt() {
    return bienvenido_txt;
}

public void setBienvenido_txt(StaticText st) {
    this.bienvenido_txt = st;
}
private Button btCerrarSesion = new Button();

public Button getBtCerrarSesion() {
    return btCerrarSesion;
```

```
    }

    public void setBtCerrarSesion(Button b) {
        this.btCerrarSesion = b;
    }
    private StaticText inicioTxt = new StaticText();

    public StaticText getInicioTxt() {
        return inicioTxt;
    }

    public void setInicioTxt(StaticText st) {
        this.inicioTxt = st;
    }
    private ImageComponent image1 = new ImageComponent();

    public ImageComponent getImage1() {
        return image1;
    }

    public void setImage1(ImageComponent ic) {
        this.image1 = ic;
    }
    private ImageComponent image2 = new ImageComponent();

    public ImageComponent getImage2() {
        return image2;
    }

    public void setImage2(ImageComponent ic) {
        this.image2 = ic;
    }


    public Inicio() {
    }

    @Override
    public void init() {
        super.init();
        getSessionBean1().addCambioListener(this);
        getApplicationBean1().setIcono("/resources/encab/nada.png");
        try {
            _init();
        } catch (Exception e) {
            log("Page1 Initialization Failure", e);
            throw e instanceof FacesException ? (FacesException) e: new
FacesException(e);
        }

        getSessionBean1().getBdatos().resetData();
        getApplicationBean1().setDscr("");
        this.inicioTxt.setText("RCQ es una Red de RecolecciÃ³n y
DistribuciÃ³n de informaciÃ³n climatolÃ³gica a travÃ©s de Internet. Esta
red estÃ¡ ubicada en la ciudad de Quito y recolecta datos de Temperatura,
Humedad Relativa, Luminosidad y Cantidad de lluvia.");
        getApplicationBean1().setMnsj("Para observar variaciones del
clima en periodos de tiempo personalizados y obtener reportes en formato
Excel seleccione estas opciones");
```

```
        getSessionBean1().setActuales();
        getSessionBean1().getBdatos().recogeDia();
    }

    @Override
    public void preprocess() {
    }

    @Override
    public void prerender() {
    }

    @Override
    public void destroy() {
    }

    protected SessionBean1 getSessionBean1() {
        return (SessionBean1) getBean("SessionBean1");
    }

    protected RequestBean1 getRequestBean1() {
        return (RequestBean1) getBean("RequestBean1");
    }

    protected ApplicationBean1 getApplicationBean1() {
        return (ApplicationBean1) getBean("ApplicationBean1");
    }

    public void enteradoCambio(EventObject e) {
        if(getSessionBean1().getUsuarioReg()){
            this.bienvenido_txt.setText("Bienvenido
"+getSessionBean1().getBdatos().infoUsuario(getSessionBean1().getUsuarioA
ctual())[0]);
            this.btCerrarSesion.setImageURL("/resources/cerrSesion.png");
        }else {
            this.bienvenido_txt.setText("");
            this.btCerrarSesion.setImageURL("/resources/blanco.png");

        }
    }

    public String cerrSesion_bt_action() {
        getSessionBean1().setUsuarioReg(false);
        return null;
    }

}
```

C.4. **Inicio.jsp**

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Document   : Page1
    Created on : 06/04/2008, 04:24:33 PM
    Author     : fvasconez
-->
```

```
<jsp:root version="2.1" xmlns:f="http://java.sun.com/jsf/core"
xmlns:grf="/WEB-INF/tlds/etiquetaApplet"
xmlns:h="http://java.sun.com/jsf/html"
    xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:webuijsf="http://www.sun.com/webui/webuijsf">
    <jsp:directive.page contentType="text/html;charset=UTF-8"
pageEncoding="UTF-8"/>
    <f:view>
        <webuijsf:page binding="#{Inicio.page1}" id="page1">
            <webuijsf:html binding="#{Inicio.html1}" id="html1" lang="">
                <webuijsf:head binding="#{Inicio.head1}" id="head1"
title="Clima en Quito: Bienvenido">
                    <webuijsf:link binding="#{Inicio.link1}" id="link1"
url="/resources/stylesheet.css"/>
                    <webuijsf:link rel="shortcut icon" type="image/x-
icon" url="/resources/rcqicon.ico"/>
                    <webuijsf:script url="/resources/menuScript.js"/>
                </webuijsf:head>
                <webuijsf:body binding="#{Inicio.body1}" id="body1"
style="background-color: rgb(255, 255, 255); -rave-layout: grid">
                    <webuijsf:form binding="#{Inicio.form1}" id="form1">
                        <webuijsf:pageSeparator
binding="#{Inicio.pageSeparator1}" id="pageSeparator1" style="height:
31px; left: 72px; top: 672px; position: absolute; width: 744px"/>
                        <webuijsf:staticText
binding="#{Inicio.ultRegistro1}" id="ultRegistro1"
                            style="background-image:
url(resources/fondo3.png); color: rgb(51, 153, 0); direction: ltr; font-
size: 14px; font-weight: bold; height: 55px; left: 24px; top: 648px;
position: absolute; text-align: center; text-indent: 5px; vertical-align:
bottom; width: 262px; letter-spacing: 1.0;word-spacing: 2.0;"
text="Ãšltimo registro ingresado: #{SessionBean1.actuales[0]} a las
#{SessionBean1.actuales[1]}"/>
                        <webuijsf:staticText
binding="#{Inicio.ult_temperatura}" id="ult_temperatura"
                            style="font-size: 14px; font-weight: bold;
height: 24px; left: 312px; top: 264px; position: absolute; width: 502px"
text="Temperatura Actual: #{SessionBean1.actuales[7]} Â°C"/>
                        <webuijsf:staticText
binding="#{Inicio.ult_humedad}" id="ult_humedad"
                            style="font-size: 14px; font-weight: bold;
height: 24px; left: 312px; top: 492px; position: absolute; width: 502px"
text="Humedad Actual: #{SessionBean1.actuales[3]} %"/>
                        <webuijsf:staticText binding="#{Inicio.mensaje}"
id="mensaje"
                            style="font-family:
'Verdana','Arial','Helvetica',sans-serif; font-size: 14px; font-weight:
bold; height: 46px; left: 24px; top: 720px; position: absolute; text-
align: justify; width: 790px" text="#{ApplicationBean1.mnsj}"/>
                        <div style="height: 358px; left: 24px; top:
192px; position: absolute; width: 262px">
                            <jsp:directive.include file="Menu.jspf"/>
                        </div>
                        <webuijsf:panelGroup
binding="#{Inicio.groupPanel1}" id="groupPanel1" style="height: 162px;
left: 312px; top: 288px; position: absolute; width: 510px">
                            <grf:jsfhello
datT="#{SessionBean1.bdatos.dat[1]}"
datT5="#{SessionBean1.bdatos.dat[5]}" h="162" hellomsg="" tempe="true"
```

```
                                tmsg="Temperatura en las Ã°ltimas 24
horas" w="510" fini="#{SessionBean1.bdatos.etiquno}"
ffin="#{SessionBean1.bdatos.etiqdos}"/>
                        </webuijsf:panelGroup>
                        <webuijsf:panelGroup
binding="#{Inicio.groupPanel2}" id="groupPanel2" style="height: 162px;
left: 312px; top: 516px; position: absolute; width: 510px">
                            <grf:jsfhello
datH="#{SessionBean1.bdatos.dat[0]}" h="162" hellomsg="" hmsg="Humedad en
las Ã°ltimas 24 horas" humedad="true" w="510"
fini="#{SessionBean1.bdatos.etiquno}"
ffin="#{SessionBean1.bdatos.etiqdos}"/>
                        </webuijsf:panelGroup>
                        <div style="height: 94px; left: 24px; top: 768px;
position: absolute; width: 790px">
                            <jsp:directive.include file="Personal.jspf"/>
                        </div>
                        <webuijsf:staticText
binding="#{Inicio.bienvenido_txt}" id="bienvenido_txt" style="color:
blue; font-family: Georgia,'Times New Roman',times,serif; font-size:
18px; height: 22px; left: 24px; top: 168px; position: absolute; width:
262px"/>
                        <webuijsf:button
actionExpression="#{Inicio.cerrSesion_bt_action}"
binding="#{Inicio.btCerrarSesion}" id="btCerrarSesion"
                            imageURL="/resources/blanco.png"
style="height: 24px; left: 935px; top: 168px; position: absolute; width:
91px" text="Cerrar SesiÃ³n"/>
                        <div style="height: 334px; left: 840px; top:
192px; position: absolute; width: 216px">
                            <jsp:directive.include file="Registro.jspf"/>
                        </div>
                        <div style="height: 336px; left: 840px; top:
530px; position: absolute; width: 216px">
                            <jsp:directive.include file="Enlaces.jspf"/>
                        </div>
                        <webuijsf:image binding="#{Inicio.image1}"
id="image1" style="height: 142px; left: 24px; top: 0px; position:
absolute; width: 1030px" url="/resources/encab/nube4.JPG"/>
                        <webuijsf:staticText
binding="#{Inicio.inicioTxt}" id="inicioTxt" style="font-family:
Georgia,'Times New Roman',times,serif; font-size: 14px; height: 72px;
left: 312px; top: 192px; position: absolute; text-align: justify;
vertical-align: super; width: 504px"/>
                        <webuijsf:image binding="#{Inicio.image2}"
height="96" id="image2"
                            style="height: 90px; left: 84px; top: 36px;
position: absolute; width: 700px" url="/resources/encab/titulo4.png"
width="648"/>
                    </webuijsf:form>
                </webuijsf:body>
            </webuijsf:html>
        </webuijsf:page>
    </f:view>
</jsp:root>
```

C.5. Datos.java

```
/*
 * Datos.java
 *
 * Created on 06/04/2008, 04:25:56 PM
 */

package rcquito;

import com.sun.rave.web.ui.appbase.AbstractPageBean;
import com.sun.webui.jsf.component.Body;
import com.sun.webui.jsf.component.Form;
import com.sun.webui.jsf.component.Head;
import com.sun.webui.jsf.component.HiddenField;
import com.sun.webui.jsf.component.Html;
import com.sun.webui.jsf.component.Hyperlink;
import com.sun.webui.jsf.component.ImageComponent;
import com.sun.webui.jsf.component.Link;
import com.sun.webui.jsf.component.Page;
import com.sun.webui.jsf.component.StaticText;
import java.util.Date;
import java.util.EventObject;
import javax.faces.FacesException;
import javax.faces.component.html.HtmlCommandButton;
import javax.faces.component.html.HtmlPanelGrid;
import javax.faces.component.html.HtmlPanelGroup;
import misEtiquetas.appletEtiqueta;

/**
 * <p>Page bean that corresponds to a similarly named JSP page.  This
 * class contains component definitions (and initialization code) for
 * all components that you have defined on this page, as well as
 * lifecycle methods and event handlers where you may add behavior
 * to respond to incoming events.</p>
 *
 * @author fvasconez
 */
public class Datos extends AbstractPageBean implements CambioListener{
    // <editor-fold defaultstate="collapsed" desc="Managed Component
Definition">

    private void _init() throws Exception {
    }

    private Page page1 = new Page();

    public Page getPage1() {
        return page1;
    }

    public void setPage1(Page p) {
        this.page1 = p;
    }

    private Html html1 = new Html();

    public Html getHtml1() {
```

```
    return html1;
}

public void setHtml1(Html h) {
    this.html1 = h;
}

private Head head1 = new Head();

public Head getHead1() {
    return head1;
}

public void setHead1(Head h) {
    this.head1 = h;
}

private Link link1 = new Link();

public Link getLink1() {
    return link1;
}

public void setLink1(Link l) {
    this.link1 = l;
}

private Body body1 = new Body();

public Body getBody1() {
    return body1;
}

public void setBody1(Body b) {
    this.body1 = b;
}

private Form form1 = new Form();

public Form getForm1() {
    return form1;
}

public void setForm1(Form f) {
    this.form1 = f;
}
private HiddenField hiddenField1 = new HiddenField();

public HiddenField getHiddenField1() {
    return hiddenField1;
}

public void setHiddenField1(HiddenField hf) {
    this.hiddenField1 = hf;
}

private int[] par= {-1, -1, -1, -1};

public int[] getPar(){
    return par;
```

```
}

private appletEtiqueta graf = new appletEtiqueta();

public appletEtiqueta getGraf(){
    return graf;
}

public void setGraf(appletEtiqueta app){
    graf=app;
}
private HtmlPanelGrid gridPanel1 = new HtmlPanelGrid();

public HtmlPanelGrid getGridPanel1() {
    return gridPanel1;
}

public void setGridPanel1(HtmlPanelGrid hpg) {
    this.gridPanel1 = hpg;
}
private StaticText staticText1 = new StaticText();

public StaticText getStaticText1() {
    return staticText1;
}

public void setStaticText1(StaticText st) {
    this.staticText1 = st;
}
private HtmlPanelGrid gridPanel3 = new HtmlPanelGrid();

public HtmlPanelGrid getGridPanel3() {
    return gridPanel3;
}

public void setGridPanel3(HtmlPanelGrid hpg) {
    this.gridPanel3 = hpg;
}
private HtmlCommandButton reporte = new HtmlCommandButton();

public HtmlCommandButton getReporte() {
    return reporte;
}

public void setReporte(HtmlCommandButton hcb) {
    this.reporte = hcb;
}
private StaticText lb_medios = new StaticText();

public StaticText getLb_medios() {
    return lb_medios;
}

public void setLb_medios(StaticText st) {
    this.lb_medios = st;
}
private StaticText lb_Temp = new StaticText();

public StaticText getLb_Temp() {
    return lb_Temp;
```

```
}

public void setLb_Temp(StaticText st) {
    this.lb_Temp = st;
}
private StaticText lb_Temp2 = new StaticText();

public StaticText getLb_Temp2() {
    return lb_Temp2;
}

public void setLb_Temp2(StaticText st) {
    this.lb_Temp2 = st;
}
private StaticText lb_Humedad = new StaticText();

public StaticText getLb_Humedad() {
    return lb_Humedad;
}

public void setLb_Humedad(StaticText st) {
    this.lb_Humedad = st;
}
private StaticText lb_lumi = new StaticText();

public StaticText getLb_lumi() {
    return lb_lumi;
}

public void setLb_lumi(StaticText st) {
    this.lb_lumi = st;
}
private StaticText lb_lumin2 = new StaticText();

public StaticText getLb_lumin2() {
    return lb_lumin2;
}

public void setLb_lumin2(StaticText st) {
    this.lb_lumin2 = st;
}
private StaticText lb_pluv = new StaticText();

public StaticText getLb_pluv() {
    return lb_pluv;
}

public void setLb_pluv(StaticText st) {
    this.lb_pluv = st;
}
private HtmlPanelGroup groupPanel1 = new HtmlPanelGroup();

public HtmlPanelGroup getGroupPanel1() {
    return groupPanel1;
}

public void setGroupPanel1(HtmlPanelGroup hpg) {
    this.groupPanel1 = hpg;
}
```

```
    public Datos() {
    }

    @Override
    public void init() {
        super.init();
        getApplicationBean1().setIcono("/resources/encab/icoRp.png");
        this.t=""+getSessionBean1().getPares()[0];
        this.h=""+getSessionBean1().getPares()[1];
        this.l=""+getSessionBean1().getPares()[2];
        this.p=""+getSessionBean1().getPares()[3];
        int cantidad = 0;
        for(int i=0;i<4;i++){
            if(getSessionBean1().getPares()[i])cantidad++;
        }
        switch(cantidad){
            case 1: this.setEstilo("height: 144px; left: 24px; top:
520px; position: absolute");
                    break;
            case 2: this.setEstilo("height: 144px; left: 24px; top:
555px; position: absolute");
                    break;
            case 3: this.setEstilo("height: 144px; left: 24px; top:
710px; position: absolute");
                    break;
            case 4: this.setEstilo("height: 144px; left: 24px; top:
885px; position: absolute");
                    break;
        }
        getSessionBean1().addCambioListener(this);
        try {
            _init();
        } catch (Exception e) {
            log("Datos Initialization Failure", e);
            throw e instanceof FacesException ? (FacesException) e: new
FacesException(e);
        }

        getApplicationBean1().setDscr("Observe cÃ³mo ha variado el clima
en Quito en el perÃodo de tiempo que usted elija. TendrÃ¡ a disposiciÃ³n
un archivo Excel con una tabla de los datos graficados para descargar.");
        int i=0; boolean alguno=false;
    }

    @Override
    public void preprocess() {
    }

    @Override
    public void prerender() {
    }

    @Override
    public void destroy() {
        getSessionBean1().getBdatos().resetData();

    }

    protected SessionBean1 getSessionBean1() {
```

```
        return (SessionBean1) getBean("SessionBean1");
    }

    protected ApplicationBean1 getApplicationBean1() {
        return (ApplicationBean1) getBean("ApplicationBean1");
    }

    protected Bdatos getBdatos(){
        System.out.println("getBdatos en Datos.jsp");
        return (Bdatos) getBean("Bdatos");
    }

    protected RequestBean1 getRequestBean1() {
        return (RequestBean1) getBean("RequestBean1");
    }

    public String reporte_action() {
        return "case2";
    }

    public String getT(){
        return t;
    }
    public String getH(){
        return h;
    }
    public String getL(){
        return l;
    }
    public String getP(){
        return p;
    }
    private String t,h,l,p;
    private final Date primeraFecha = new Date(11949300000000L);
//2007/11/13

    public void enteradoCambio(EventObject e) {
        this.t=""+getSessionBean1().getPares()[0];
        this.h=""+getSessionBean1().getPares()[1];
        this.l=""+getSessionBean1().getPares()[2];
        this.p=""+getSessionBean1().getPares()[3];
        int cantidad = 0;
        for(int i=0;i<4;i++){
            if(getSessionBean1().getPares()[i])cantidad++;
        }
        switch(cantidad){
            case 1: this.setEstilo("height: 144px; left: 24px; top:
520px; position: fixed");
                    break;
            case 2: this.setEstilo("height: 144px; left: 24px; top:
555px; position: fixed");
                    break;
            case 3: this.setEstilo("height: 144px; left: 24px; top:
710px; position: fixed");
                    break;
            case 4: this.setEstilo("height: 144px; left: 24px; top:
885px; position: fixed");
                    break;
        }
    }
```

```
    private String estilo="height: 144px; left: 24px; top: 520px;
position: fixed";

    public void setEstilo(String newEstilo){
        estilo = newEstilo;
    }

    public String getEstilo(){
        return estilo;
    }
}
```

C.6. **Datos.jsp**

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Document   : Datos
    Created on : 06/04/2008, 04:25:56 PM
    Author     : fvasconez
-->
<jsp:root version="2.1" xmlns:f="http://java.sun.com/jsf/core"
xmlns:grf="/WEB-INF/tlds/etiquetaApplet"
xmlns:h="http://java.sun.com/jsf/html"
    xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:webuijsf="http://www.sun.com/webui/webuijsf">
    <jsp:directive.page contentType="text/html;charset=UTF-8"
pageEncoding="UTF-8"/>
    <f:view>
        <webuijsf:page binding="#{Datos.page1}" id="page1">
            <webuijsf:html binding="#{Datos.html1}" id="html1">
                <webuijsf:head binding="#{Datos.head1}" id="head1"
title="Clima en Quito: GrÃ¡ficos">
                    <webuijsf:link binding="#{Datos.link1}" id="link1"
url="/resources/stylesheet.css"/>
                    <webuijsf:link rel="shortcut icon" type="image/x-
icon" url="/resources/rcqicon.ico"/>
                    <webuijsf:script url="/resources/menuScript.js"/>
                </webuijsf:head>
                <webuijsf:body binding="#{Datos.body1}" id="body1"
style="-rave-layout: grid">
                    <webuijsf:form autoComplete="false"
binding="#{Datos.form1}" id="form1" style="margin-left: 2px; width:
780px">
                        <!--Panel donde estÃ¡n los controles para que el
usuario escoja lo que quiere ver-->
                        <div style="height: 142px; left: 24px; top: 0px;
position: absolute; width: 1030px">
                            <jsp:directive.include
file="Encabezado.jspf"/>
                        </div>
                        <!--Panel donde se va a graficar el parÃ¡metro
que quiero-->
                        <div style="height: 72px; left: 24px; top: 168px;
position: absolute; width: 648px">
                            <jsp:directive.include file="Personal.jspf"/>
                        </div>
```

```
                        <h:panelGrid binding="#{Datos.gridPanel1}"
id="gridPanel1"
                            style="background-image:
url(resources/fondo4.png); height: 388px; left: 840px; top: 168px;
position: absolute" width="216">
                            <webuijsf:staticText
binding="#{Datos.staticText1}" id="staticText1"
                                style="font-family: Georgia,'Times New
Roman',times,serif; font-size: 18px; font-weight: bold; left: 51px; top:
3px; position: absolute; text-align: center" text="EstadÃsticas:"/>
                            <webuijsf:staticText
binding="#{Datos.lb_medios}" id="lb_medios"
                                style="font-family: Georgia,'Times New
Roman',times,serif; font-size: 14px; font-weight: bold" text="Valores
Medios:"/>
                            <webuijsf:staticText
binding="#{Datos.lb_Temp}" id="lb_Temp"
                                style="font-family: Georgia,'Times New
Roman',times,serif; font-size: 14px"
                                text="Temperatura (interno):
#{SessionBean1.bdatos.estadistica[0][1]} Â°C"
visible="#{SessionBean1.pares[0]}"/>
                            <webuijsf:staticText
binding="#{Datos.lb_Temp2}" id="lb_Temp2"
                                style="font-family: Georgia,'Times New
Roman',times,serif; font-size: 14px"
                                text="Temperatura (externo):
#{SessionBean1.bdatos.estadistica[0][5]} Â°C"
visible="#{SessionBean1.pares[0]}"/>
                            <webuijsf:staticText
binding="#{Datos.lb_Humedad}" id="lb_Humedad"
                                style="font-family: Georgia,'Times New
Roman',times,serif; font-size: 14px"
                                text="Humedad Relativa:
#{SessionBean1.bdatos.estadistica[0][0]} %"
visible="#{SessionBean1.pares[1]}"/>
                            <webuijsf:staticText
binding="#{Datos.lb_lumi}" id="lb_lumi"
                                style="font-family: Georgia,'Times New
Roman',times,serif; font-size: 14px"
                                text="Luminosidad (TSR):
#{SessionBean1.bdatos.estadistica[0][2]} lux"
visible="#{SessionBean1.pares[2]}"/>
                            <webuijsf:staticText
binding="#{Datos.lb_lumin2}" id="lb_lumin2"
                                style="font-family: Georgia,'Times New
Roman',times,serif; font-size: 14px"
                                text="Luminosidad (PAR):
#{SessionBean1.bdatos.estadistica[0][3]} lux"
visible="#{SessionBean1.pares[2]}"/>
                            <webuijsf:staticText
binding="#{Datos.lb_pluv}" id="lb_pluv"
                                style="font-family: Georgia,'Times New
Roman',times,serif; font-size: 14px"
                                text="Pluviosidad:
#{SessionBean1.bdatos.estadistica[0][6]} mm"
visible="#{SessionBean1.pares[3]}"/>
                        </h:panelGrid>
                        <h:commandButton action="#{Datos.reporte_action}"
binding="#{Datos.reporte}" id="reporte" image="/resources/btRep.png"
```

```
style="height: 54px; left: 864px; top: 576px; position: absolute; width:
162px"/>
                        <div style="height: 192px; left: 24px; top:
288px; position: absolute; width: 262px">
                            <jsp:directive.include file="Menu.jspf"/>
                        </div>
                        <h:panelGroup binding="#{Datos.groupPanel1}"
id="groupPanel1" style="height: 216px; left: 312px; top: 288px; position:
absolute; width: 504px">
                            <grf:jsfhello binding="#{Datos_2.graf}"
datH="#{SessionBean1.bdatos.dat[0]}" datL="#{SessionBean1.bdatos.dat[2]}"
                                datL3="#{SessionBean1.bdatos.dat[3]}"
datP="#{SessionBean1.bdatos.dat[6]}" datT="#{SessionBean1.bdatos.dat[1]}"
                                datT5="#{SessionBean1.bdatos.dat[5]}"
h="162" hellomsg="#{SessionBean1.mensaje}"
                                hmsg="Humedad Relativa en Quito
(EstaciÃ³n Centro)" humedad="#{Datos_2.h}" id="graf" lumin="#{Datos_2.l}"
pluvio="#{Datos_2.p}"
                                tempe="#{Datos_2.t}" tmsg="Temperatura en
Quito (EstaciÃ³n Centro)" w="510"
                                fini="#{SessionBean1.bdatos.etiquno}"
ffin="#{SessionBean1.bdatos.etiqdos}"/>
                        </h:panelGroup>
                    </webuijsf:form>
                </webuijsf:body>
            </webuijsf:html>
        </webuijsf:page>
    </f:view>
</jsp:root>
```

C.7. Componente.java

```
/*
 *
 * JGraphic is a Java Bean that is used to draw signals, in this case
seismic signals.

 * Copyright(C) 2006 Pablo Marcillo Lara

 * This program is free software; you can redistribute it and/or modify
it under the
 * terms of the GNU General Public License as published by the Free
Software
 * Foundation; either version 2 of the License, or (at your option) any
later version.

 * This program is distributed in the hope that it will be useful, but
WITHOUT ANY
 * WARRANTY; without even the implied warranty of MERCHANTABILITY or
FITNESS FOR A
 * PARTICULAR PURPOSE. See the GNU General Public License for more
details.

 * You should have received a copy of the GNU General Public License
along with
 * this program; if not, write to the Free Software Foundation, Inc., 675
Mass Ave,
```

```
 * Cambridge, MA 02139, USA.

 * Contact me at email:  pablomarcillo@yahoo.com, pmarcillo@igepn.edu.ec
 *
 */

import java.io.*;
import java.awt.*;
import java.beans.*;
import java.lang.*;
import java.util.*;
import javax.swing.*;

class CenterPanel extends javax.swing.JPanel {
    private int array[][], arr2[][];
    private int arrayAux[][],arrAux2[][];
    private int counterArray;
    private int i;
    private int totalTime=1;
    private int divisionsX;
    private int divisionsY;
    private int zoom=1;
    private int amplify=1;
    private int maximumPrecision=1;
    private boolean kind, isDoubleGraph=false;
    private boolean negative;
    private double Yd;
    private boolean limitsFlag;
    private int leftLimit;
    private int rightLimit;
    private int number=1;
    private int X1;
    private int X2;
    private int Y1;
    private int Y2;

    /** Creates new form JGraphic */
    public CenterPanel() {
        initComponents();
    }

    private void initComponents() {

        setLayout(new java.awt.BorderLayout());

        setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0,
0, 0)));
        addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                formMouseClicked(evt);
            }
        });

    }

    private void formMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        switch(number){
            case 1:
                leftLimit=evt.getX();
```

```
                number=2;
                setLimitsFlag(false);
                break;
            case 2:
                rightLimit=evt.getX();
                if(rightLimit>leftLimit){
                    number=1;
                    setLimitsFlag(true);
                }
                break;
        }
    }

    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);


        for(i=1;i<divisionsX;i++){
            g.setColor(Color.LIGHT_GRAY);

g.drawLine((int)(i*getWidth()/divisionsX),0,(int)(i*getWidth()/divisionsX
),getHeight());
        }

        for(i=1;i<divisionsY;i++){
            g.setColor(Color.LIGHT_GRAY);

g.drawLine(0,(int)(i*getHeight()/divisionsY),getWidth(),(int)(i*getHeight
()/divisionsY));
        }

        if(negative==true){
            g.setColor(Color.BLACK);
            g.drawLine(0,getHeight()/2,getWidth(),getHeight()/2);
        }
        for(i=0;i<getArr2().length-1;i++){
           g.setColor(Color.RED);

g.drawLine(getArr2()[i][0],getArr2()[i][1],getArr2()[i+1][0],getArr2()[i+
1][1]);         //g.drawLine(X1,Y1,X2,Y2);
           g.drawLine(getArr2()[i][0],getArr2()[i][1]-
1,getArr2()[i+1][0],getArr2()[i+1][1]-1);
        }

        if(isDoubleGraph){
           for(i=0;i<getArrAux2().length-1;i++){
               g.setColor(Color.ORANGE);

g.drawLine(getArrAux2()[i][0],getArrAux2()[i][1],getArrAux2()[i+1][0],get
ArrAux2()[i+1][1]);         //g.drawLine(X1,Y1,X2,Y2);
               g.drawLine(getArrAux2()[i][0],getArrAux2()[i][1]-
1,getArrAux2()[i+1][0],getArrAux2()[i+1][1]-1);
           }
        }

    }

    public void clearScreen(){
        counterArray=0;
```

```
    array=new int[0][0];
    repaint();
}

public void setArr2(int[][] newArr){
    arr2=newArr;
}

public int[][]getArr2(){
    return arr2;
}

public void setArrAux2(int[][] newArr){
    arrAux2=newArr;
}

public int[][]getArrAux2(){
    return arrAux2;
}

public void setLimitsFlag(boolean newLimitsFlag){
    limitsFlag=newLimitsFlag;
}

public boolean getLimitsFlag(){
    return limitsFlag;
}

public int getLeftLimit(){
    return leftLimit;
}

public int getRightLimit(){
    return rightLimit;
}

public void setSizePanel(int newHeight,int newWidth){
    setSize(newHeight,newWidth);
}

public void setDivisionsX(int newDivisionsX){
    divisionsX=newDivisionsX;
    repaint();
}

public void setDivisionsY(int newDivisionsY){
    divisionsY=newDivisionsY;
    repaint();
}

public void setNegative(boolean newNegative){
    negative=newNegative;
    repaint();
}

public void setAmplify(int newAmplify){
    amplify=newAmplify;
    repaint();
}
```

```
    public void fillPointsArray(int newX,int newY){
        if(counterArray==0){
            arrayAux=new int[counterArray+1][2];
            if(kind==false)
                arrayAux[counterArray][0]=0;
            else
                arrayAux[counterArray][0]=newX;
            arrayAux[counterArray][1]=newY;
            counterArray++;
        }

        array=new int[counterArray+1][2];
        System.arraycopy(arrayAux,0,array,0,counterArray);
        array[counterArray][0]=newX;
        array[counterArray][1]=newY;
        arrayAux=new int[counterArray+1][2];
        System.arraycopy(array,0,arrayAux,0,counterArray+1);
        counterArray++;
    }

    public void setTotalTime(int newTotalTime){
        totalTime=newTotalTime;
        repaint();
    }

    public void setMaximumPrecision(int newMaximumPrecision){
        maximumPrecision=newMaximumPrecision;
    }

    public void setKind(boolean newKind){
        kind=newKind;
    }

    public void setIsDoubleGraph(boolean newState){
        isDoubleGraph=newState;
    }

    public void setZoom(int newZoom){
      zoom=newZoom;
      repaint();
    }

    public void doRepaint(){
        repaint();
    }

    public int getPointX(int newX){
        return getWidth()*newX/totalTime;
    }

    public int getPointY(int newY){
        Yd=getHeight()-(int)((getHeight()/maximumPrecision)*newY);
        //Yd=getHeight()-
(double)((double)(getHeight()*newY)/maximumPrecision);
      //Yd=(double)(Yd+zoom*(Yd-getHeight()/2));
      return (int)Yd;
    }

    // Variables declaration - do not modify
    // End of variables declaration
```

```
}

class SouthPanel extends javax.swing.JPanel {
    private final int SECONDS_PER_MINUTE=60;
    private final int MINUTES_PER_HOUR=60;
    private final int SECONDS_PER_HOUR=3600;
    private final int HOURS_DAY=24;
    private final int MARGIN=15;
    private Calendar calendar;
    private int divisionsX;
    private int i;
    private int hour;
    private int minute;
    private int totalSeconds;
    private int totalASeconds;
    private boolean kind,fromDB=true;
    private boolean pointPerData;
    private int totalTime;
    private String ini, fin;

    public SouthPanel() {
        initComponents();
        initExtraComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {

        setLayout(new java.awt.BorderLayout());

        setFont(new java.awt.Font("Tahoma", 1, 11));
    }

    public void initExtraComponents(){
        setTime();
    }

    public void paintComponent(Graphics g){
        super.paintComponent(g);

        if(totalTime!=0){
            for(i=1;i<=divisionsX;i++){
                g.setColor(Color.black);

g.drawString(getDivisionTime(i),MARGIN+(int)(i*(getWidth()-
50)/divisionsX),10);
            }
        }
       if(fromDB){
           g.setColor(Color.BLACK);
           g.setFont(new java.awt.Font("Tahoma", 1, 10));
           g.drawString(getIni(),MARGIN,10);
           g.drawString(getFin(),MARGIN+(getWidth()-130),10);
       }
    }
```

```
    public void clearScreen(){
        setTime();
        repaint();
    }

    public void setPointPerData(boolean newPointPerData){
        pointPerData=newPointPerData;
        repaint();
    }

    public void setDivisionsX(int newDivisionsX){
        divisionsX=newDivisionsX;
        repaint();
    }

    public void setTotalTime(int newTotalTime){
        totalTime=newTotalTime;
        repaint();
    }



    public String getDivisionTime(int newDivision){

            int
totalASeconds=totalSeconds+(int)(newDivision*totalTime/divisionsX);
            hour=(int)(totalASeconds/SECONDS_PER_HOUR);
            minute=(totalASeconds%SECONDS_PER_HOUR)/SECONDS_PER_MINUTE;

            while(hour>=HOURS_DAY){
                hour=hour-HOURS_DAY;
            }
            return String.valueOf(hour)+":"+String.valueOf(minute);

    }

    public void setTime(){
        if(kind==false){
            calendar=Calendar.getInstance();

totalSeconds=calendar.get(Calendar.HOUR_OF_DAY)*SECONDS_PER_HOUR+calendar
.get(Calendar.MINUTE)*SECONDS_PER_MINUTE+calendar.get(Calendar.SECOND);
        }
        else
            totalSeconds=0;
    }

    public void setKind(boolean newKind){
        kind=newKind;
        setTime();
        repaint();
    }

    public void setIni(String newTime){
        ini=newTime;
    }

    public String getIni(){
        return ini;
    }
```

```
     public void setFin(String newTime){
        fin=newTime;
    }

    public String getFin(){
        return fin;
    }

    public void cleanScreen(){
        repaint();
    }
    // Variables declaration - do not modify
    // End of variables declaration

}

class WestPanel extends javax.swing.JPanel {
    private static int MARGIN=10;
    private int divisionsY;
    private int i;
    private float lowGain;
    private float highGain;
    private int maximumPrecision;
    private float totalGain;
    private float gain;
    private String string;
    private boolean negative;

    public WestPanel() {
        initComponents();
    }

    private void initComponents() {

        setLayout(new java.awt.BorderLayout());

        setFont(new java.awt.Font("Tahoma", 1, 11));
        addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                formMouseClicked(evt);
            }
        });

    }

    private void formMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_formMouseClicked
        // TODO add your handling code here:

    }//GEN-LAST:event_formMouseClicked

    public void paintComponent(Graphics g){
        super.paintComponent(g);
        if(maximumPrecision!=0){
            for(i=0;i<divisionsY;i++){
                g.setColor(Color.black);
                string=String.valueOf(highGain-
(i*(float)(highGain+Math.abs(lowGain))/divisionsY));
```

```
g.drawString(string.substring(0,string.indexOf('.')+2),1,(int)(i*getHeigh
t()/divisionsY)+MARGIN);
            }
        }
    }

    public void clearScreen(){
        repaint();
    }

    public void setDivisionsY(int newDivisionsY){
        divisionsY=newDivisionsY;
        repaint();
    }

    public void setMaximumPrecision(int newMaximumPrecision){
        maximumPrecision=newMaximumPrecision;
        repaint();
    }

    public void setTotalGain(float newTotalGain){
        totalGain=newTotalGain;
        setGain(totalGain);
    }

    public void setGain(float newGain){
        gain=newGain;
        if(negative==true){
            highGain=gain/2;
            lowGain=gain/2*-1;
        }
        else{
            highGain=gain;
            lowGain=0;
        }
        repaint();
    }

    public void setNegative(boolean newNegative){
        negative=newNegative;
        setGain(gain);
    }

    // Variables declaration - do not modify
    // End of variables declaration

}

class JGraphic extends javax.swing.JPanel implements Serializable{
    private java.beans.PropertyChangeSupport propertyChangeSupport=new
java.beans.PropertyChangeSupport(this);
    private static int SOUTHPANEL_HEIGHT=25;
    private static int LABELY_HEIGHT=15;
    private static int LABELX_HEIGHT=15;
    private static int TIME_HEIGHT=15;
    private static int WESTPANEL_WIDTH=50;
    private static int RIGHT_MARGIN=10;
    private static int TITLE_HEIGHT=24;
    private static int LEFT_MARGIN=45;
```

```
    private Color panelColor=Color.lightGray;
    private FontMetrics fontMetrics;
    private String axisX="X Title";
    private String axisY="Y Title";
    private String title="Component Title";
    private int divisionsX;
    private int divisionsY;
    private float totalGain;
    private int maximumPrecision=0;
    private int totalTime=0;
    private int amplify;
    private boolean kind=true;
    private boolean negative;
    private boolean pointPerData;

    public JGraphic() {
        initComponents();
        initExtraComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jPanel1 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();

        setLayout(new java.awt.BorderLayout());

        setBorder(new javax.swing.border.TitledBorder(null, "Title
Component", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 1, 11)));
        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel1.setForeground(java.awt.Color.blue);
        jLabel1.setText("Y Title");
        add(jLabel1, java.awt.BorderLayout.NORTH);

        jLabel2.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel2.setForeground(java.awt.Color.blue);
        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        jLabel2.setText("X Title");
        add(jLabel2, java.awt.BorderLayout.SOUTH);

        jPanel1.setLayout(new java.awt.BorderLayout());

        jLabel4.setFont(new java.awt.Font("Tahoma", 1, 11));
        jPanel1.add(jLabel4, java.awt.BorderLayout.SOUTH);

        add(jPanel1, java.awt.BorderLayout.CENTER);

    }//GEN-END:initComponents

    private void initExtraComponents(){
      setPreferredSize(new Dimension(510,154));
        setSize(new Dimension(504,154));
```

```
        jLabel1.setPreferredSize(new
Dimension(getWidth(),LABELY_HEIGHT));
        jLabel1.setSize(getWidth(),LABELY_HEIGHT);
        jLabel2.setPreferredSize(new
Dimension(getWidth(),LABELX_HEIGHT));
        jLabel2.setSize(getWidth(),LABELX_HEIGHT);
        jLabel4.setPreferredSize(new Dimension(getWidth(),TIME_HEIGHT));
        jLabel4.setSize(getWidth(),TIME_HEIGHT);
        WestPanel1=new WestPanel();
        WestPanel1.setPreferredSize(new
Dimension(WESTPANEL_WIDTH,getHeight()));
        WestPanel1.setMaximumSize(new
Dimension(WESTPANEL_WIDTH,getHeight()));
        CenterPanel1=new CenterPanel();
        SouthPanel1=new SouthPanel();
        SouthPanel1.setPreferredSize(new
Dimension(getWidth(),SOUTHPANEL_HEIGHT));
        SouthPanel1.setMaximumSize(new
Dimension(getWidth(),SOUTHPANEL_HEIGHT));
        jPanel1.add(WestPanel1,java.awt.BorderLayout.WEST);
        jPanel1.add(CenterPanel1,java.awt.BorderLayout.CENTER);
        jPanel1.add(SouthPanel1,java.awt.BorderLayout.SOUTH);
    }

    public void setPanelColor(Color newPanelColor){
        Color oldPanelColor=panelColor;
        panelColor=newPanelColor;
        CenterPanel1.setBackground(panelColor);

propertyChangeSupport.firePropertyChange("panelColor",oldPanelColor,newPa
nelColor);
    }

    public Color getPanelColor(){
        return panelColor;
    }

    public void setTitle(String newTitle){
      String oldTitle=title;
      title=newTitle;
      setBorder(new javax.swing.border.TitledBorder(null,title,
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 1, 11)));

      propertyChangeSupport.firePropertyChange("title",oldTitle,newTitle)
;
    }

    public String getTitle(){
      return title;
    }

    public boolean getFLagLimits(boolean newFlagLimit){
        return newFlagLimit;
    }

    public int getLeftLimit(int newLeftLimit){
        return newLeftLimit;
    }
```

```
    public int rightLeftLimit(int newRightLimit){
        return newRightLimit;
    }

    public void setAxisX(String newAxisX){
        String oldAxisX=axisX;
        axisX=newAxisX;
        jLabel2.setText(axisX);

propertyChangeSupport.firePropertyChange("axisX",oldAxisX,newAxisX);
    }

    public void setAxisY(String newAxisY){
        String oldAxisY=axisY;
        axisY=newAxisY;
        jLabel1.setText(axisY);

propertyChangeSupport.firePropertyChange("axisY",oldAxisY,newAxisY);
    }

    public String getAxisX(){
        return axisX;
    }

    public String getAxisY(){
        return axisY;
    }

    public void setTotalGain(float newTotalGain){
        float oldTotalGain=totalGain;
        totalGain=newTotalGain;
        WestPanel1.setTotalGain(totalGain);
        propertyChangeSupport.firePropertyChange("totalGain",new
Float(oldTotalGain),new Float(newTotalGain));
    }

    public float getTotalGain(){
        return totalGain;
    }

    public void setGain(int newGain){
        this.WestPanel1.setGain(newGain);
    }

    public void setAmplify(int newAmplify){
        int oldAmplify=amplify;
        amplify=newAmplify;
        CenterPanel1.setAmplify(amplify);

propertyChangeSupport.firePropertyChange("amplify",oldAmplify,newAmplify)
;
    }

    public int getAmplify(){
        return amplify;
    }

    public void setMaximumPrecision(int newMaximumPrecision){
        int oldMaximumPrecision=maximumPrecision;
```

```
        maximumPrecision=newMaximumPrecision;
        CenterPanel1.setMaximumPrecision(maximumPrecision);
        WestPanel1.setMaximumPrecision(maximumPrecision);

propertyChangeSupport.firePropertyChange("maximumPrecision",oldMaximumPre
cision,newMaximumPrecision);
    }

    public int getMaximumPrecision(){
        return maximumPrecision;
    }

    public void setZoom(int newZoom){
      CenterPanel1.setZoom(newZoom);
    }

    public boolean getKind(){
        return kind;
    }

    public void setKind(boolean newKind){
        boolean oldKind=kind;
        kind=newKind;
        SouthPanel1.setKind(kind);
        CenterPanel1.setKind(kind);
        propertyChangeSupport.firePropertyChange("kind",oldKind,newKind);
    }

    public void setNegative(boolean newNegative){
        boolean oldNegative=negative;
        negative=newNegative;
        WestPanel1.setNegative(negative);
        CenterPanel1.setNegative(negative);

propertyChangeSupport.firePropertyChange("negative",oldNegative,newNegati
ve);
    }

    public boolean getNegative(){
        return negative;
    }

    public void setPointPerData(boolean newPointPerData){
        boolean oldPointPerData=pointPerData;
        pointPerData=newPointPerData;
        SouthPanel1.setPointPerData(pointPerData);

propertyChangeSupport.firePropertyChange("pointPerData",oldPointPerData,n
ewPointPerData);
    }

    public boolean getPointPerData(){
        return pointPerData;
    }

    public Dimension getCanvasSize(){
      return new Dimension(getWidth()-WestPanel1.getWidth()-RIGHT_MARGIN-
LEFT_MARGIN,getHeight()-TITLE_HEIGHT-SouthPanel1.getHeight()-
jLabel1.getHeight()-jLabel2.getHeight()-jLabel4.getHeight());
    }
```

```
    public void setCanvasSize(Dimension dimension){
      CenterPanel1.setSizePanel(dimension.height,dimension.width);
    }

    public void setTotalTime(int newTotalTime){
        int oldTotalTime=totalTime;
        totalTime=newTotalTime;
        CenterPanel1.setTotalTime(totalTime);
        SouthPanel1.setTotalTime(totalTime);

propertyChangeSupport.firePropertyChange("totalTime",oldTotalTime,newTota
lTime);
    }

    public int getTotalTime(){
        return totalTime;
    }

    public void clearScreen(){
        CenterPanel1.clearScreen();
        SouthPanel1.clearScreen();
        WestPanel1.clearScreen();
    }

    public void setDivisionsX(int newDivisionsX){
        int oldDivisionsX=divisionsX;
        divisionsX=newDivisionsX;
        CenterPanel1.setDivisionsX(divisionsX);
        SouthPanel1.setDivisionsX(divisionsX);

propertyChangeSupport.firePropertyChange("divisionsX",oldDivisionsX,newDi
visionsX);
    }

    public void setDivisionsY(int newDivisionsY){
        int oldDivisionsY=divisionsY;
        divisionsY=newDivisionsY;
        CenterPanel1.setDivisionsY(divisionsY);
        WestPanel1.setDivisionsY(divisionsY);

propertyChangeSupport.firePropertyChange("divisionsY",oldDivisionsY,newDi
visionsY);
    }

    public CenterPanel CenterPanel1;
    public WestPanel WestPanel1;
    public SouthPanel SouthPanel1;

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration//GEN-END:variables

}

public class Componente extends JApplet{
    private JGraphic jGraphic1;
```

```
    private String Data=new String();
    private String Dat2=new String();
    private String Title=new String();
    private int[] DataInt=new int[1500];
    private int[] Dat2Int=new int[1500];
    private int[][] pointsToDraw, points2;
    private Dimension dimension;
    private String Units;
    private int idns;
    private int idne;
    private int cntSens;
    private int k=0,l=0;
    private int H,L;

    public void init(){

        jGraphic1 = new JGraphic();
        setCanvasSize();
        getParameters();
        jGraphic1.setTitle(Title);
        jGraphic1.setNegative(false);
        jGraphic1.setAxisX("Tiempo");
        jGraphic1.CenterPanel1.setKind(true);

      getContentPane().setLayout(new FlowLayout());
      getContentPane().add(jGraphic1);

    }

    private void getParameters(){
        idne = Integer.parseInt(getParameter("idne"));
        idns = Integer.parseInt(getParameter("idns"));
        cntSens = Integer.parseInt(getParameter("csns"));
        Data = getParameter("data");
        Title = getParameter("titl");
        Units = getParameter("unds");
        jGraphic1.SouthPanel1.setIni(getParameter("inlb"));
        jGraphic1.SouthPanel1.setFin(getParameter("fnlb"));


        StringTokenizer st= new StringTokenizer(Data);
        while(st.hasMoreTokens()){
            DataInt[k++]=Integer.parseInt(st.nextToken());
            }
        if(k!=0){
            pointsToDraw= new int[--k][2];
            points2= new int[k][2];
            discrimine(idns, DataInt);
            for(int g=0;g<k;g++){
                pointsToDraw[g][0]=L-(L*g/(k-1));
                pointsToDraw[g][1]=H-
(int)((H/jGraphic1.getTotalGain())*DataInt[g]);
            }
        }else{
            discrimine(idns, DataInt);
            pointsToDraw = new int[2][2];
            points2 = new int[2][2];
            pointsToDraw [0][0]= 0; points2[0][0]=0;
            pointsToDraw[1][0]= 0; points2[1][0]=0;
            pointsToDraw[0][1]= 0; points2[0][1]=0;
```

```
            pointsToDraw[1][1]= 0; points2[1][1]=0;
        }
        jGraphic1.CenterPanel1.setArr2(pointsToDraw);
        if(cntSens==2){
            jGraphic1.CenterPanel1.setIsDoubleGraph(true);
            int afin=Integer.parseInt(getParameter("afin"));
            Dat2=getParameter("dat2");
            st=new StringTokenizer(Dat2);
            while(st.hasMoreTokens()){
                Dat2Int[l++]=Integer.parseInt(st.nextToken());
            }
            discrimine(afin, Dat2Int);
            for(int g=0;g<k;g++){
                points2[g][0]=L-(L*g/(k-1));
                points2[g][1]=H-
(int)((H/jGraphic1.getTotalGain())*Dat2Int[g]);
            }
            jGraphic1.CenterPanel1.setArrAux2(points2);
        }

    }

    private void setCanvasSize(){
        dimension=jGraphic1.getCanvasSize();
        H=74;
        L=dimension.width;
        jGraphic1.setCanvasSize(dimension);
    }

   public void discrimine(int sensor, int[] Dat){
    int[] buff = {205,205,205};
    int bx = 0;
    switch(sensor){
        case 0: //humidity = -4 + 0.0405*SOrh + (-2.8 * 10^-6)*(SOrh^2)
                    jGraphic1.setMaximumPrecision(16384);
                    jGraphic1.setAxisY("Humedad Relativa"+"
["+Units+"]");
                    jGraphic1.setNegative(false);
                    jGraphic1.setTotalGain(100);
                    jGraphic1.setDivisionsX(12);
                    jGraphic1.setDivisionsY(5);
                    for(int i=0;i<k;i++){
                        Dat[i]-=16;
                        Dat[i]=(int)Dat[i]/26;
                        }
                    break;

            case 1: //temperatura = -39.60 + 0.01*dato
                    jGraphic1.setMaximumPrecision(16384);
                    jGraphic1.setAxisY("Temperatura"+" ["+Units+"]");
                    jGraphic1.setNegative(false);
                    jGraphic1.setTotalGain(40);
                    jGraphic1.setDivisionsX(12);
                    jGraphic1.setDivisionsY(4);
                    for(int i=0;i<k;i++){
                        Dat[i]-=4000;
                        Dat[i]=(int)(Dat[i]/100);
                        }
                    break;
```

```
            case 2: //S1087    lx = 1e6 * I * 1000
                    jGraphic1.setMaximumPrecision(4096);
                    jGraphic1.setAxisY("Luminosidad"+" ["+Units+"]");
                    jGraphic1.setNegative(false);
                    jGraphic1.setTotalGain(150);
                    jGraphic1.setDivisionsX(12);
                    jGraphic1.setDivisionsY(7);
                    for(int i=0;i<k;i++){
                        Dat[i]=(int)Dat[i]/10;
                    }

                    break;

            case 3: //S1087-01 lx = 1e5 * I * 1000
                    jGraphic1.setMaximumPrecision(4096);
                    jGraphic1.setAxisY("Luminosidad"+" ["+Units+"]");
                    jGraphic1.setNegative(false);
                    jGraphic1.setTotalGain(150);
                    jGraphic1.setDivisionsX(12);
                    jGraphic1.setDivisionsY(7);
                    for(int i=0;i<k;i++){
                        Dat[i]=(int)Dat[i]/10;
                    }

                    break;

            case 4: //value/4096 * Vref
                    jGraphic1.setMaximumPrecision(4096);
                    jGraphic1.setAxisY("BaterÃa"+" ["+Units+"]");
                    jGraphic1.setNegative(false);
                    jGraphic1.setTotalGain(12);
                    jGraphic1.setDivisionsX(12);
                    jGraphic1.setDivisionsY(6);
                    for(int i=0;i<k;i++){
                       Dat[i]=(int)(Dat[i]/1365);
                    }
                    break;

            case 5: //Temperatura Externa
                    jGraphic1.setMaximumPrecision(4096);
                    jGraphic1.setAxisY("Temperatura"+" ["+Units+"]");
                    jGraphic1.setNegative(false);
                    jGraphic1.setTotalGain(40);
                    jGraphic1.setDivisionsX(12);
                    jGraphic1.setDivisionsY(4);
                    Dat[i]=(int)( Dat[i]*3/41)-9;
                    break;

            case 6: //Pluviï¿½metro
                    jGraphic1.setMaximumPrecision(4096);
                    jGraphic1.setNegative(false);
                    jGraphic1.setAxisY("Cantidad de Lluvia"+"
["+Units+"]");
                    jGraphic1.setTotalGain(6);
                    jGraphic1.setDivisionsX(12);
                    jGraphic1.setDivisionsY(6);
                    break;

          }
    }
```

```
    private final Color col0= Color.RED;
    private final Color col1= Color.ORANGE;

      }
```

C.8. appletEtiqueta.java

```
package misEtiquetas;

/**
 *
 * @author fvasconez
 */
import javax.faces.component.UIComponentBase;
import javax.faces.context.FacesContext;
import java.io.IOException;
import javax.faces.context.ResponseWriter;

public class appletEtiqueta extends UIComponentBase {

    public void encodeBegin(FacesContext context) throws IOException {
        ResponseWriter writer = context.getResponseWriter();
        String hellomsg = (String) getAttributes().get("hellomsg");
        String tempe= String.valueOf(getAttributes().get("tempe"));
        String T = (String)getAttributes().get("tmsg");
        String humedad = String.valueOf(getAttributes().get("humedad"));
        String H = (String)getAttributes().get("hmsg");
        String luminosidad =
String.valueOf(getAttributes().get("lumin"));
        String pluvio = String.valueOf(getAttributes().get("pluvio"));
        String bateria = (String)getAttributes().get("bateria");
        String datT= (String) getAttributes().get("datT");
        String datT5= (String) getAttributes().get("datT5");
        String datH= (String) getAttributes().get("datH");
        String datL= (String) getAttributes().get("datL");
        String datL3= (String) getAttributes().get("datL3");
        String datP= (String) getAttributes().get("datP");
        String ancho = (String) getAttributes().get("w");
        String alto = (String) getAttributes().get("h");
        String inilb = (String) getAttributes().get("fini");
        String finlb = (String) getAttributes().get("ffin");

        writer.startElement("h3", this);
            if(hellomsg != null)
                writer.writeText(hellomsg, "hellomsg");
            else
                writer.writeText("Componente grÃ¡fico para parÃ¡metros
climatolÃ³gicos", null);
        writer.endElement("h3");

        if(tempe.equals("true")){

            writer.startElement("object", this);
                writer.writeAttribute("codebase", "./comp", null);
                writer.writeAttribute("code", "Componente.class", null);
```

```
                writer.writeAttribute("type", "application/x-java-
applet", null);
                writer.writeAttribute("width", ancho, null);
                writer.writeAttribute("height", alto, null);


                writer.startElement("param", this);
                writer.writeAttribute("name", "idns", null);
                writer.writeAttribute("value", "1", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "idne", null);
                writer.writeAttribute("value", "0", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "csns", null);
                writer.writeAttribute("value", 2, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "data", null);
                writer.writeAttribute("value", datT, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "dat2", null);
                writer.writeAttribute("value", datT5, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "afin", null);
                writer.writeAttribute("value", "5", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "titl", null);
                writer.writeAttribute("value", T, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "inlb", null);
                writer.writeAttribute("value", inilb, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "fnlb", null);
                writer.writeAttribute("value", finlb, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "unds", null);
                writer.writeAttribute("value", "Â°C", null);
                writer.endElement("param");

            writer.endElement("object");
            writer.startElement("br", null);
            writer.startElement("img", null);
```

```
                    writer.writeAttribute("src", "resources/rojo.png",
null);
                    writer.writeAttribute("border", "0", null);
                    writer.writeAttribute("height", "10", null);
                writer.endElement("img");
                writer.writeText("Sensor interno. ", null);
                writer.startElement("img", null);
                    writer.writeAttribute("src", "resources/naranja.png",
null);
                    writer.writeAttribute("border", "0", null);
                    writer.writeAttribute("height", "10", null);
                writer.endElement("img");
                writer.writeText("Sensor externo. ", null);
            }
        /**
        *   Si se ha escojido la Humedad, se visualizarÃ¡ mediante este
cÃ³digo
        *
        **/
         if(humedad.equals("true")){
            writer.startElement("object", this);
                writer.writeAttribute("codebase", "./comp", null);
                writer.writeAttribute("code", "Componente.class", null);
                writer.writeAttribute("type", "application/x-java-
applet", null);
                writer.writeAttribute("width", ancho, null);
                writer.writeAttribute("height", alto, null);


                writer.startElement("param", this);
                writer.writeAttribute("name", "idns", null);
                writer.writeAttribute("value", "0", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "idne", null);
                writer.writeAttribute("value", "0", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "csns", null);
                writer.writeAttribute("value", "1", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "data", null);
                writer.writeAttribute("value", datH, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "dat2", null);
                writer.writeAttribute("value", "", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "afin", null);
                writer.writeAttribute("value", "S", null);
                writer.endElement("param");

                writer.startElement("param", this);
```

```
                writer.writeAttribute("name", "titl", null);
                writer.writeAttribute("value", H, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "inlb", null);
                writer.writeAttribute("value", inilb, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "fnlb", null);
                writer.writeAttribute("value", finlb, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "unds", null);
                writer.writeAttribute("value", "%", null);
                writer.endElement("param");

        writer.endElement("object");
        }

         /**
        *   Si se ha escojido la Luminosidad, se visualizarÃ¡ mediante
este cÃ³digo
        *
        **/

        if(luminosidad.equals("true")){
            writer.startElement("object", this);
            writer.writeAttribute("codebase", "./comp", null);
            writer.writeAttribute("code", "Componente.class", null);
            writer.writeAttribute("type", "application/x-java-applet",
null);
            writer.writeAttribute("width", ancho, null);
            writer.writeAttribute("height", alto, null);


                writer.startElement("param", this);
                writer.writeAttribute("name", "idns", null);
                writer.writeAttribute("value", "2", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "idne", null);
                writer.writeAttribute("value", "0", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "csns", null);
                writer.writeAttribute("value", "2", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "data", null);
                writer.writeAttribute("value", datL, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "dat2", null);
```

```
                writer.writeAttribute("value", datL3, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "afin", null);
                writer.writeAttribute("value", "3", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "titl", null);
                writer.writeAttribute("value", "Luminosidad en Quito
(EstaciÃ³n Centro)", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "inlb", null);
                writer.writeAttribute("value", inilb, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "fnlb", null);
                writer.writeAttribute("value", finlb, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "unds", null);
                writer.writeAttribute("value", "x10^6 lux", null);
                writer.endElement("param");

        writer.endElement("object");
            writer.startElement("br", null);
                writer.startElement("img", null);
                    writer.writeAttribute("src", "resources/rojo.png",
null);
                    writer.writeAttribute("border", "0", null);
                    writer.writeAttribute("height", "10", null);
                writer.endElement("img");
                writer.writeText("Espectro visible. ", null);
                writer.startElement("img", null);
                    writer.writeAttribute("src", "resources/naranja.png",
null);
                    writer.writeAttribute("border", "0", null);
                    writer.writeAttribute("height", "10", null);
                writer.endElement("img");
                writer.writeText("Espectro parcial. ", null);

        }

         /**
        *   Si se ha escojido la Pluviosidad, se visualizarÃ¡ mediante
este cÃ³digo
        *
        **/

        if(pluvio.equals("true")){
            writer.startElement("object", this);
            writer.writeAttribute("codebase", "./comp", null);
            writer.writeAttribute("code", "Componente.class", null);
            writer.writeAttribute("type", "application/x-java-applet",
null);
```

```
            writer.writeAttribute("width", ancho, null);
            writer.writeAttribute("height", alto, null);


                writer.startElement("param", this);
                writer.writeAttribute("name", "idns", null);
                writer.writeAttribute("value", "6", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "idne", null);
                writer.writeAttribute("value", "0", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "csns", null);
                writer.writeAttribute("value", "1", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "data", null);
                writer.writeAttribute("value", datP, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "dat2", null);
                writer.writeAttribute("value", "", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "afin", null);
                writer.writeAttribute("value", "S", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "titl", null);
                writer.writeAttribute("value", "Lluvias en Quito
(EstaciÃ³n Centro)", null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "inlb", null);
                writer.writeAttribute("value", inilb, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "fnlb", null);
                writer.writeAttribute("value", finlb, null);
                writer.endElement("param");

                writer.startElement("param", this);
                writer.writeAttribute("name", "unds", null);
                writer.writeAttribute("value", "mm", null);
                writer.endElement("param");

        writer.endElement("object");
        }
    }

    public String getFamily() {
```

```
        return "HelloFamily";
    }
      }
```

C.9. SessionBean1.java

```
/*
 * SessionBean1.java
 *
 * Created on 06/04/2008, 04:24:33 PM
 */

package rcquito;

import com.sun.rave.web.ui.appbase.AbstractSessionBean;
import java.util.Date;
import java.util.Vector;
import javax.faces.FacesException;

/**
 *
 * @author fvasconez
 */
public class SessionBean1 extends AbstractSessionBean {
    private void _init() throws Exception {
    }

    public SessionBean1() {
    }

    @Override
    public void init() {
        super.init();
        try {
            _init();
        } catch (Exception e) {
            log("SessionBean1 Initialization Failure", e);
            throw e instanceof FacesException ? (FacesException) e: new
FacesException(e);
        }

    }

    @Override
    public void passivate() {
    }

    @Override
    public void activate() {
    }

    @Override
    public void destroy() {


    }
```

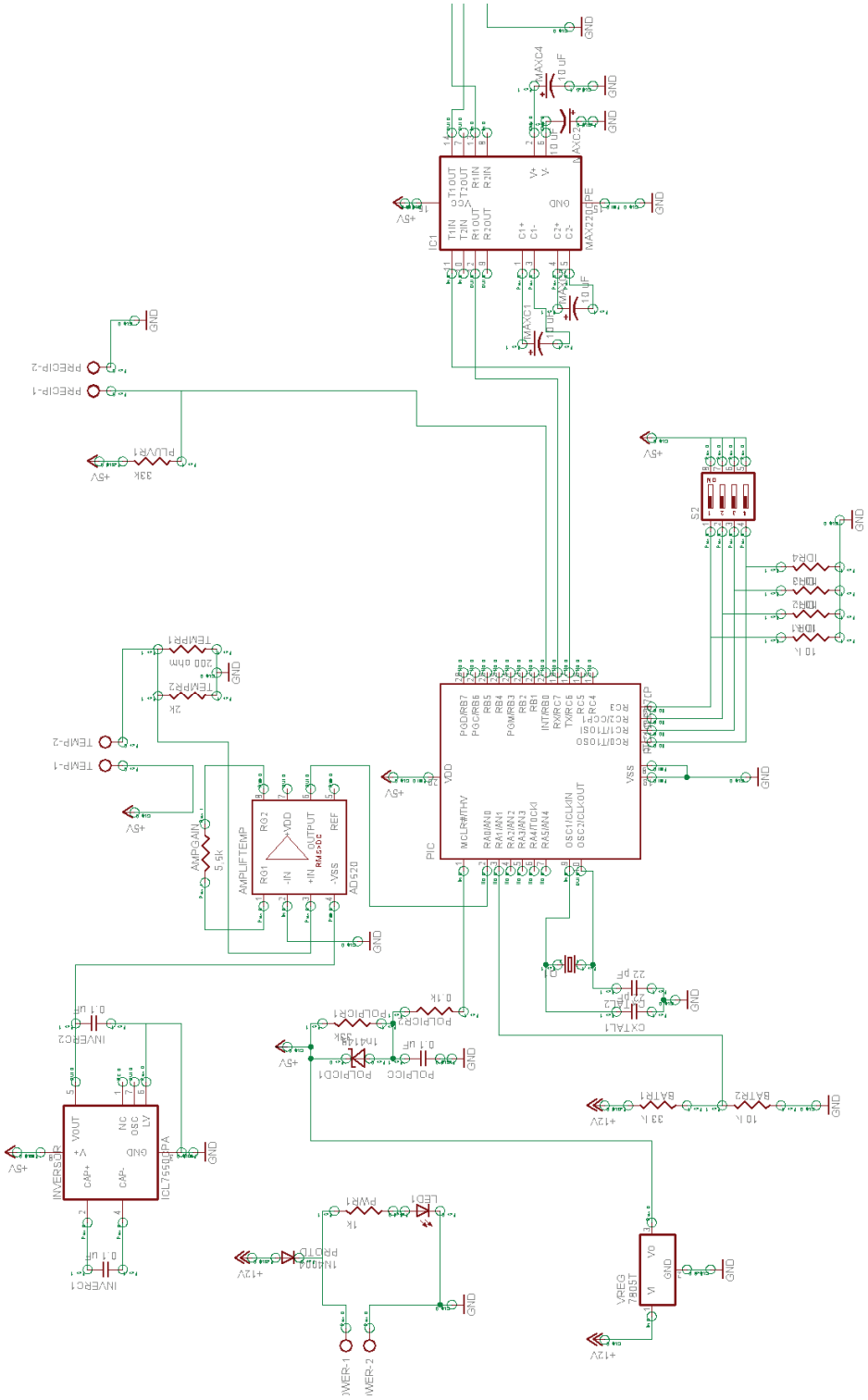
```
    protected ApplicationBean1 getApplicationBean1() {
        return (ApplicationBean1) getBean("ApplicationBean1");
    }

    public void setFchaIni(Date nuevaFecha){
        fchaIni=nuevaFecha;
        escIni=true;
    }

    public Date getFchaIni(){
        return fchaIni;

    }

    public void setFchaFin(Date nuevaFecha){
        fchaFin=nuevaFecha;
        escFin=true;
    }

    public Date getFchaFin(){
        return fchaFin;
    }

    public Bdatos getBdatos(){
        //base= new Bdatos();

        return base;
    }

    public void setActuales(){
        actuales=getBdatos().getActual();
    }

    public String[] getActuales(){
        return actuales;
    }

    public void setMensaje(String newMsg){
       mensaje=newMsg;
    }

    public String getMensaje(){
        return mensaje;
    }
    public boolean[] getPares(){
        return pares;
    }

    public void setPares(boolean newCheck, int pos){
        try{
            boolean antPar = pares[pos];
            pares[pos]=newCheck;
            if(antPar!=newCheck){
                CambioEvt event=new CambioEvt(this, antPar, newCheck);
                notificarCambio(event);
            }

        }catch(Exception obx){System.out.println("Error al escoger los
parÃ¡metros: "+obx);}
    }
```

```
    public void escogerParametros(){
        getBdatos().setParametros(pares);
    }

    public void setFechas(){

base.setFcha_ini(""+(getFchaIni().getYear()+1900)+"/"+(getFchaIni().getMo
nth()+1)+"/"+getFchaIni().getDate());

base.setFcha_fin(""+(getFchaFin().getYear()+1900)+"/"+(getFchaFin().getMo
nth()+1)+"/"+getFchaFin().getDate());
    }


    public synchronized void addCambioListener(CambioListener listener){
        cambioListeners.addElement(listener);
    }

    public synchronized void removeCambioListener(CambioListener
listener){
        cambioListeners.removeElement(listener);
    }

    private void notificarCambio(CambioEvt event){
        Vector lista;
        synchronized(this){
            lista=(Vector)cambioListeners.clone();
        }
        for(int i=0; i<lista.size(); i++){
            CambioListener listener=(CambioListener)lista.elementAt(i);
            listener.enteradoCambio(event);
        }
  }

    public void setUsuarioReg(boolean newEstado){
        try{
            boolean antEstado = usuarioReg;
            usuarioReg = newEstado;
            if(antEstado!=newEstado){
                CambioEvt event=new CambioEvt(this, antEstado,
newEstado);
                notificarCambio(event);
            }

        }catch(Exception obx){System.out.println("Error al registrar al
usuario: "+obx);}
    }

    public boolean getUsuarioReg(){
        return usuarioReg;
    }

    public void setUsuarioActual(String newNomb){
        usuarioActual = newNomb;
    }

    public String getUsuarioActual(){
        return usuarioActual;
    }
```

```
    private boolean usuarioReg = false;
    private String usuarioActual;
    private Vector cambioListeners=new Vector();
    private String[] actuales= new String[10];
    private Date fchaIni;
    private Date fchaFin;
    private Bdatos base= new Bdatos();
    private String mensaje="Escoja el/los parÃ¡metros para visualizar y
las fechas entre las que quiere obtener datos";
    private boolean escIni=false, escFin=false;
    private boolean[] pares={true, false, false, false};
}
```

D. ESQUEMA DE CONEXIONES

D.1. SISTEMA TRADICIONAL



D.2. SISTEMA TELOS