



## INSTITUTO TECNOLÓGICO DE MEXICALI

**RED DE SENSORES INALÁMBRICA PARA LA DETECCIÓN DE  
INCENDIOS FORESTALES**

### TESIS

**PARA OBTENER EL GRADO DE  
MAESTRO EN INGENIERÍA ELECTRÓNICA**

**PRESENTA**

**ING. JORGE ANTONIO ATEMPA CAMACHO**

**DIRECTOR DE TESIS**

**DR. ARNOLDO DÍAZ RAMÍREZ**



MEXICALI B.C. JUNIO DE 2014

## **Agradecimientos**

Afortunadamente, existen motivos y personas a quien les puedo agradecer todo este proceso de crecimiento personal y profesional, en primer lugar gracias a Dios. En mi carrera y en mi vida, ha sido muy importante contar con el apoyo de mis padres y hermanos ante todo tipo de circunstancias, momentos buenos y malos. Su enseñanza, me ha permitido formar una visión de constante aprendizaje, afrontar desafíos y entre muchas cosas a valorar lo importante. El ejemplo de mis maestros de todos los niveles académicos, en su búsqueda por motivar al alumno me inspira y fortalece, para seguir aprendiendo, para seguir avanzando, para seguir comprendiendo que la única forma de cambiar las circunstancias es enfrentandolas. La paciencia y el apoyo de mis amigos y compañeros, el respaldo y su confianza me permite retroalimentar mis ideas de maneras muy divertidas, pero a la vez, muy enfocadas a mejorar continuamente. Gracias a la DGEST, por haberme respaldado con una beca.

## **Dedicatoria**

Claudia Camacho Herrera (mamá) y Sergio Atempa Flores (papá), espero sigan conmigo por muchos años más, que aun tenemos cosas por vivir y por hacer.

Arnoldo Díaz Ramírez, unirme a su equipo de investigación, fue de las mejores cosas que me han pasado y de la cual me siento muy agradecido.

Veronica Quintero, Guadalupe Amado, Amalia Medina, Marisela Ponce, Claudia Martínez, Carolina Ruiz, Miriam Rendon, Carlos Lopez, Arnoldo Díaz, Francisco Ibañez, Fernando Garcia, Jaime Olvera, Mario Chong, Heber Hernandez, maestros que me proveyeron de herramientas y valores útiles para mi desarrollo personal y profesional.

Miguel, Nancy, Maripaz, Claudia y Karla Atempa, Neftali y Josué Casillas son una luz en mi camino y un impulso para seguir aprendiendo.

Por último, pero no menos importante el CAII, lugar donde conocí personas muy talentosas, su dedicación me entusiasma y me motiva.

# Índice general

<b>Índice general</b>	<b>3</b>
<b>Índice de figuras</b>	<b>5</b>
<b>Índice de tablas</b>	<b>8</b>
<b>1. Introducción</b>	<b>9</b>
1.1. Objetivo General . . . . .	11
1.2. Objetivos Específicos . . . . .	11
<b>2. Redes ad hoc</b>	<b>12</b>
2.1. Características . . . . .	13
2.2. Clasificación . . . . .	14
2.3. Protocolos de encaminamiento . . . . .	15
<b>3. Red de sensores inalámbrica</b>	<b>16</b>
3.1. Características . . . . .	17
3.2. Clasificación . . . . .	18
3.3. Estándar IEEE 802.15.4 y ZigBee . . . . .	19
3.4. Fusión de información . . . . .	21
3.4.1. Clasificación . . . . .	21
3.4.1.1. Basada en las relaciones de las fuentes . . . . .	21
3.4.1.2. Basada en los niveles de abstracción . . . . .	22
3.4.1.3. Basada en los datos de entrada y salida . . . . .	22
3.5. Desarrollo a la fecha . . . . .	23
3.5.1. Técnicas para áreas residenciales . . . . .	23
3.5.2. Técnicas para eventos en bosques . . . . .	28
3.5.3. Técnicas para detección temprana . . . . .	36

<b>4. TinyOS y nesC</b>	<b>37</b>
4.1. TinyOS . . . . .	37
4.2. nesC . . . . .	38
4.2.1. Convenciones del lenguaje nesC . . . . .	39
4.3. Instalación de TinyOS . . . . .	40
4.3.1. Estructura de directorios . . . . .	41
4.3.2. Configuración adicional . . . . .	42
4.4. Desarrollo de aplicaciones en TinyOS . . . . .	43
4.4.1. Aplicación Blink . . . . .	43
4.4.2. Aplicación para la transmisión de datos ambientales . . . . .	46
4.4.2.1. Aplicación para el nodo sensor . . . . .	46
4.4.2.2. Aplicación para la estación base . . . . .	52
<b>5. Método propuesto para la detección de incendios forestales</b>	<b>54</b>
5.1. Motivación . . . . .	54
5.2. Observaciones . . . . .	55
5.3. Técnicas utilizadas para el desarrollo del modelo . . . . .	57
5.3.1. Análisis de regresión . . . . .	57
5.3.2. Método de mínimos cuadrados . . . . .	57
5.3.3. Razón de cambio . . . . .	61
5.3.4. Error cuadrático promedio . . . . .	62
5.4. Arquitectura del modelo propuesto . . . . .	63
5.4.1. Red de sensores inalámbrica . . . . .	63
5.4.2. Middleware . . . . .	64
5.4.3. Algoritmo de detección de incendios . . . . .	65
<b>6. Evaluación</b>	<b>68</b>
6.1. Plataforma de hardware y software utilizada . . . . .	69
6.2. Metodología . . . . .	70
6.3. Desarrollo del modelo base . . . . .	71
6.4. Evaluación del algoritmo propuesto . . . . .	75
6.5. Resultados . . . . .	78
<b>7. Conclusiones y Trabajo futuro</b>	<b>81</b>
<b>Bibliografía</b>	<b>83</b>

# Índice de figuras

1.1. Arquitectura de una red de sensores inalámbrica. . . . .	9
1.2. Causas de incendios forestales. . . . .	10
2.1. Ejemplo de una red ad hoc. . . . .	12
3.1. Arquitectura del nodo sensor. . . . .	16
3.2. Red de sensores inalámbrica. . . . .	17
3.3. Arquitectura de ZigBee. . . . .	19
3.4. Topologías: a) Estrella b) Punto a punto c) Árbol . . . . .	20
3.5. Arquitectura de red. . . . .	24
3.6. Resultado de los experimentos. . . . .	26
3.7. Arquitectura del sistema FireNet. . . . .	27
3.8. Red de sensores inalámbrica para la detección de incendios forestales en tiempo real. . . . .	28
3.9. Procesamiento de Paquete RR utilizando el método de redes neuronales. . . . .	29
3.10. Estructura del sistema FWI. . . . .	29
3.11. Máquina de estados para definir las condiciones del ambiente. . . . .	30
3.12. Arquitectura del sistema de monitoreo. . . . .	31
3.13. Arquitectura de LACU. . . . .	32
3.14. Sistema de detección de incendios basado en reglas de lógica difusa. . . . .	34
3.15. Simulador de eventos de incendio. . . . .	35
3.16. Sistema EIDOS para combate de incendios forestales. . . . .	35
3.17. Estructura de las tres capas del sistema para la detección de fuego. . . . .	36
4.1. Sección por defecto en motelist. . . . .	42
4.2. Sección modificada en motelist. . . . .	42
4.3. Reconocimiento del dispositivo. . . . .	42
4.4. Diagrama de componentes de Blink. . . . .	43
4.5. Archivo de configuración. . . . .	44

4.6.	Declaración de interfaces en el archivo de implementación. . . . .	44
4.7.	Método de arranque de la aplicación Blink. . . . .	44
4.8.	Eventos de los temporizadores. . . . .	45
4.9.	Enlace de componentes. . . . .	45
4.10.	Reglas de compilación. . . . .	45
4.11.	Diagrama de componentes de Mts400Tester. . . . .	47
4.12.	Estructura para el almacenamiento de datos. . . . .	47
4.13.	Componente de configuración de Mts400TesterC.nc. . . . .	48
4.14.	Componente de implementación de Mts400TesterP.nc. . . . .	48
4.15.	Declaración de variables para nuestra implementación de componentes. . . . .	48
4.16.	Evento inicial de la aplicación. . . . .	49
4.17.	Implementación de los métodos de SplitControl. . . . .	49
4.18.	Método para la lectura de la temperatura ambiental. . . . .	49
4.19.	Almacenamiento de la temperatura en una variable temporal. . . . .	50
4.20.	Creación del paquete de datos. . . . .	50
4.21.	Envío de datos por medio de un mensaje de multi-difusión (Broadcast). . . . .	50
4.22.	Evento de confirmación del envío de mensajes. . . . .	51
4.23.	Enlace de componentes de la aplicación Mts400Tester. . . . .	51
4.24.	Parámetros inciales en el archivo Makefile. . . . .	51
4.25.	Configuración estándar de MIG. . . . .	51
4.26.	Importando <b>tinyos.jar</b> desde el Buildpath del proyecto. . . . .	53
4.27.	Resultado de la evaluación en el entorno de Eclipse. . . . .	53
5.1.	Comportamiento en condiciones normales. . . . .	55
5.2.	Experimentos de fuego controlado. . . . .	55
5.3.	Comparación entre: a) Condiciones normales b) Experimento de fuego controlado. . . . .	56
5.4.	La relación entre las unidades manifiesta un comportamiento decreciente para ambos casos. . . . .	56
5.5.	Gráfica de los datos de la Tabla 5.1. . . . .	59
5.6.	Descripción de los datos de la Tabla 5.1. . . . .	60
5.7.	Función obtenida por el método de mínimos cuadrados. . . . .	61
5.8.	Razón de cambio entre dos puntos. . . . .	62
5.9.	Pendiente de la recta constante. . . . .	62
5.10.	Arquitectura propuesta para el sistema de detección de incendios. . . . .	63

6.1.	Escenarios de recolección de datos. . . . .	68
6.2.	Kit de desarrollo para WSN. . . . .	69
6.3.	Diagrama de la base de datos. . . . .	69
6.4.	Exposición del nodo sensor ante fuego artificial. . . . .	70
6.5.	Datos de la etapa de función base. . . . .	71
6.6.	a) 24 funciones base, b) 17 funciones base. . . . .	72
6.7.	Resultado del análisis de regresión. . . . .	72
6.8.	Evaluación del modelo base. . . . .	73
6.9.	Razones de cambio con alto grado de similitud. . . . .	77
6.10.	Modelo base de 2014. . . . .	80

# Índice de tablas

3.1. Formato del paquete CC1100. . . . .	24
5.1. Datos para la evaluación del método de mínimos cuadrados. . . . .	59
5.2. Sumas de apoyo. . . . .	60
5.3. Asignación básica de masa del Algoritmo 5.4. . . . .	67
6.1. Horario de los experimentos. . . . .	68
6.2. Experimento del 9 de septiembre de 2013. . . . .	75
6.3. Resumen de las evaluaciones. . . . .	76
6.4. Parámetros requeridos por el sistema de detección. . . . .	78
6.5. Parámetros para la evaluación. . . . .	78
6.6. Experimentos totales: 24. Período de muestreo: 3 segundos. . . . .	78
6.7. Experimentos totales: 24. Período de muestreo: 4 segundos. . . . .	79
6.8. Experimentos totales: 23. Período de muestreo: 6 segundos. . . . .	79
6.9. Experimentos totales: 22. Período de muestreo: 8 segundos. . . . .	79
6.10. Evaluación de experimentos realizados en Abril 2014. . . . .	80
6.11. Resultados de la evaluacion de Abril 2014. . . . .	80

# Capítulo 1

## Introducción

Las redes MANET (*Mobile Ad Hoc Network*, por sus siglas en inglés) representan una alternativa como tecnología emergente para distintos campos de aplicación como: salud, seguridad, detección de eventos, entretenimiento, entre otros. Una red MANET o *red ad hoc*, es una red transitoria formada dinámicamente por una colección de dispositivos móviles con capacidad de comunicación inalámbrica. Estas redes son creadas, por ejemplo, cuando un grupo de personas se reúne, y utilizan dispositivos móviles (teléfonos inteligentes, computadoras portátiles, tabletas electrónicas, etc.) para compartir información, sin necesidad de contar con una infraestructura preexistente [1].

Una red de sensores inalámbrica o WSN (*Wireless Sensor Network*, por sus siglas en inglés), es un tipo de red ad hoc. Sin embargo, en este tipo de redes los nodos son dispositivos pequeños, autónomos y de recursos de energía y procesamiento computacional limitado. Cada nodo están equipado con sensores que pueden detectar, medir y recopilar información del medio ambiente, para después transmitirla a una estación base [2], como se observa en la Fig. 1.1.

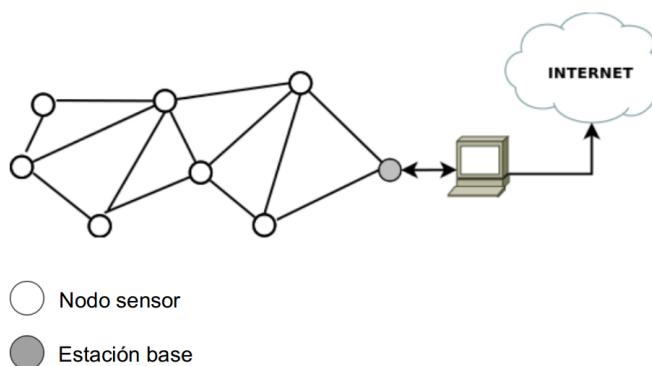


Figura 1.1: Arquitectura de una red de sensores inalámbrica.

Los incendios forestales son una de las principales causas de degradación ambiental en el mundo. Estos dañan severamente a los ecosistemas terrestres, provocan cuantiosas pérdidas materiales y en muchos de los casos también pérdida de vidas humanas. Sólo en 2012, en México se registraron 7,170 incendios forestales, obteniendo como consecuencia 347,226 hectáreas afectadas en diferentes regiones del país [3]. En promedio el 98 % fueron causados por actividades humanas y 2 % por causas naturales, como se observa en la Fig. 1.2.

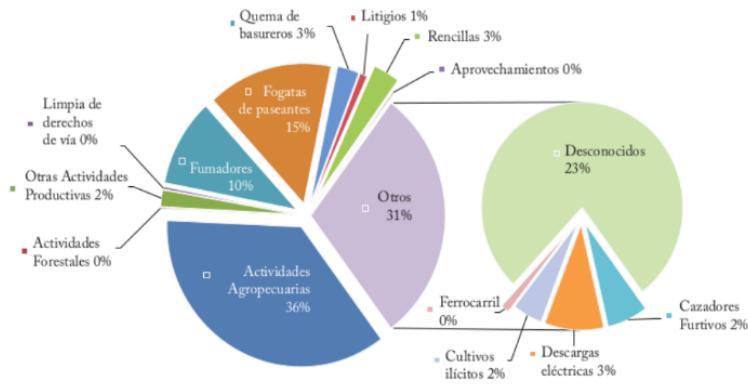


Figura 1.2: Causas de incendios forestales.

Utilizar un red de sensores inalámbrica, facilita la tarea de obtener información acerca de parámetros relacionados, por ejemplo la temperatura, humedad relativa y/o niveles de CO (monóxido de carbono), desde distintos puntos dentro de un área definida de manera continua. Estos datos son transmitidos en tiempo real para su procesamiento y análisis, y en caso de ser necesario alertar al usuario sobre una posible amenaza de incendio.

## **1.1. Objetivo General**

Desarrollar un sistema de detección de incendios forestales en su etapa inicial, utilizando una red de sensores inalámbrica y métodos de fusión de información.

## **1.2. Objetivos Específicos**

- Determinar el tipo y número de sensores mínimos para la detección de incendio forestal en su etapa inicial.
- Utilizar métodos para caracterizar un incendio cuya aproximación se ajuste mejor a los datos obtenidos.
- Desarrollar un modelo para la detección de incendios forestales, utilizando técnicas de fusión de información.
- Implementar un prototipo con el modelo propuesto utilizando un nodo.

# Capítulo 2

## Redes ad hoc

Una MANET (Mobile ad hoc network por sus siglas en inglés) o red ad hoc, es una colección autónoma de dispositivos móviles que se comunican entre sí a través de enlaces inalámbricos; cooperan de una forma distribuida con el fin de proporcionar la funcionalidad de red necesaria en la ausencia de una infraestructura fija [4]. Como se observa en la Figura 2.1, cada nodo es capaz de comunicarse directamente con cualquier otro, siempre y cuando se encuentren dentro de una misma área de transmisión.

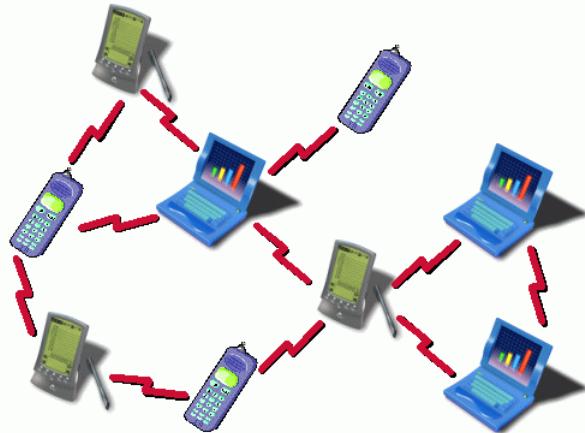


Figura 2.1: Ejemplo de una red ad hoc.

Para la comunicación entre nodos en diferente área de transmisión, se necesita utilizar nodos intermedios para la retransmisión de paquetes. Por lo general las rutas entre los nodos de una red ad hoc suelen incluir múltiples saltos, por lo tanto también se les conoce como “*Redes inalámbricas ad hoc multi-salto*”.

## 2.1. Características

Las redes ad hoc son una alternativa flexible, de fácil instalación y bajo costo. Algunas de sus características, se describen a continuación:

- **Autonomía y de poca infraestructura.** Las redes ad hoc no cuentan con una infraestructura establecida o administración centralizada por ejemplo un punto de acceso. Cada nodo genera datos independientes y la administración de la red tiene que ser distribuida entre los diferentes nodos.
- **Encaminamiento multi-salto.** Cuando un nodo intenta enviar información a otros nodos fuera de su rango de comunicación, el paquete debe ser enviado a través de uno o más nodos intermedios. No existe una ruta por defecto, por lo que cada nodo es un punto de acceso y encaminador en la retransmisión de paquetes.
- **Topología dinámica.** En este tipo de redes, los nodos pueden reposicionarse o moverse arbitrariamente, por lo tanto, la topología cambia con frecuencia y de manera impredecible, como resultado, se presentan cambios en las rutas y posiblemente pérdida de paquetes.
- **Variación en las capacidades del nodo y enlace.** Cada nodo puede estar equipado con una o más interfaces de radio que tienen diferentes capacidades de transmisión y recepción, que operan a través de diferentes bandas de frecuencia. El diseño de protocolos de red y algoritmos para este tipo de redes heterogéneas puede ser muy complejo, debido a que requieren adaptación dinámica a las condiciones cambiantes.
- **Administración de la energía.** Los nodos cuentan con una batería limitada y a no ser que dispongan de algún mecanismo de carga como por ejemplo un panel solar, no tienen capacidad de recarga. Esta es la consideración de diseño más importante de las redes ad hoc.
- **Escalabilidad:** En algunos tipos de redes, el número de nodos puede crecer de cientos a miles, por ejemplo en una red de sensores inalámbrica. La escalabilidad es crítica para el éxito en el despliegue de estas redes. Al no existir un punto de acceso concreto la incorporación y/o el descarte de nodos es un proceso transparente para el cliente.

## 2.2. Clasificación

De acuerdo con su área de cobertura, existen diferentes tipos de redes MANET:

- **Red de área corporal:** Una red de área corporal inalámbrica o BAN(*Body Area Network* por su significado en idioma inglés), consiste en pequeños dispositivos inteligentes conectados o implantados en el cuerpo humano, con la capacidad de establecer un enlace de comunicación inalámbrica [5]. Estos dispositivos proporcionan vigilancia continua sobre aspectos relacionados con la salud y permiten la retroalimentación de la información en tiempo real [6]. El rango de comunicación en la red de área corporal corresponde al alcance del cuerpo humano, es decir, 1-2 m. El cableado de dispositivos en el cuerpo humano por lo general es molesto e incómodo, por lo tanto, tecnologías inalámbricas constituyen la mejor solución para la interconexión de dispositivos portátiles.
- **Red de área personal:** Una red de área personal o PAN (*Personal Area Network* por su significado en idioma inglés) es una red en el medio ambiente en torno a las personas, es decir, generalmente se crean a partir de enlaces entre dispositivos móviles tales como PDA's, tabletas electrónicas, entre otros. El rango de comunicación de una PAN normalmente es de 10 m.
- **Redes inalámbricas de área local:** Comúnmente denominada Wi-Fi (Wireless Fidelity), se trata de una tecnología LAN inalámbrica (Red de área local inalámbrica o WLAN por sus siglas en inglés). Los nodos suelen ser dispositivos complejos de recursos elevados, como computadoras portátiles, dispositivos móviles, etcétera. Las WLAN deben estar diseñadas para hacer frente a algunas cuestiones específicas para el entorno inalámbrico, como la seguridad en el aire, el consumo de energía, la movilidad, y la limitación del ancho de banda de la interfaz aérea. El rango de comunicación de una WLAN es de 100 a 500 m.
- **Redes inalámbricas de área extendida y metropolitana:** Este tipo de redes presentan muchos retos por resolver, por ejemplo, direccionamiento, encaminamiento, gestión de localización, seguridad, etcétera.

## 2.3. Protocolos de encaminamiento

Uno de los principales desafíos en el uso de redes ad hoc, es el encaminamiento de paquetes, debido a la movilidad continua de los nodos se presentan problemas de almacenamiento y de potencia en la comunicación inalámbrica. Un protocolo de encaminamiento, es un mecanismo de comunicación que permite a los nodos transmitir y recibir información a través de la red. El propósito de utilizar un protocolo de encaminamiento incluye:

- Descubrimiento de redes remotas.
- Mantenimiento de información de encaminamiento actualizada.
- Selección de la mejor ruta hacia el destino.
- Capacidad de encontrar una nueva ruta si la ruta actual deja de estar disponible.

Los protocolos de encaminamiento de las redes ad hoc se clasifican en tres categorías:

- **Protocolos proactivos:** Estos protocolos periódicamente envían información de encaminamiento para que en cualquier momento cualquier nodo pueda comunicarse con cualquier otro de la red. El uso de un protocolo de encaminamiento proactivo, permite evaluar continuamente las rutas a todos los nodos de la red y tratan de mantener información coherente y actualizada al momento de retransmitir un paquete.
- **Protocolos reactivos o bajo demanda:** Estos protocolos realizan operaciones de descubrimiento de ruta, sólo cuando es necesario. El procedimiento de descubrimiento de ruta termina cuando se ha encontrado una ruta válida hacia el destino o cuando no existe ninguna ruta disponible después de un análisis exhaustivo.
- **Protocolos híbridos:** Los protocolos de encaminamiento híbridos, combinan las ventajas de los protocolos proactivos y reactivos. Normalmente, los protocolos de encaminamiento híbridos, se utilizan en arquitecturas de redes jerárquicas. La ruta es establecida con mecanismos de encaminamiento proactivo, mientras que la difusión de mensajes utiliza mecanismos de encaminamiento reactivo para los nodos que se estén incorporando a la red, es decir, nodos nuevos.

# Capítulo 3

## Red de sensores inalámbrica

Una red de sensores inalámbrica o WSN (Wireless Sensor Network), consiste en un gran número de nodos sensores con capacidad de comunicación inalámbrica, desplegados densamente en el medio ambiente. Un nodo sensor se caracteriza por su pequeño tamaño y su capacidad de detectar los fenómenos ambientales a través de un conjunto de transductores. Cada nodo sensor como se observa en la Fig. 3.1, esta equipado con un micro-controlador, una memoria, una fuente de energía, una radio de baja potencia, un actuador, uno o más sensores capaces de medir las condiciones del ambiente, tales como temperatura, humedad, presión barométrica, movimiento, presencia de plagas o de componentes químicos, entre otras.

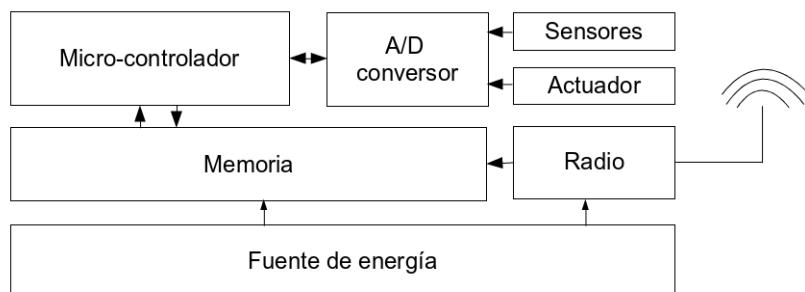


Figura 3.1: Arquitectura del nodo sensor.

Una red de sensores inalámbrica, se puede implementar en diversos campos como por ejemplo: aplicaciones militares, ambientales, de salud y comerciales, entre otras. El diseño de una red de sensores inalámbrica esta condicionado en gran medida por el tipo de aplicación para la que será utilizada. Pero también existen otros factores que influyen en su diseño, entre los que se encuentran el grado requerido de tolerancia a fallos, escalabilidad, costo, medio de transmisión, topología y consumo de energía por parte de los nodos.

### 3.1. Características

Una red de sensores inalámbrica como se observa en la Fig. 3.2, generalmente cuenta con poca o nula infraestructura. Una red de sensores inalámbrica sin estructura comúnmente se utiliza en áreas de difícil acceso, por lo que contienen una densa colección de nodos distribuidos de manera aleatoria en el área de interés, y requieren de poca atención. La desventaja de este tipo de redes consiste en la dificultad para administrarlas y detectar fallos, debido en gran medida a la gran cantidad de nodos existentes. En contraste, una red de sensores inalámbrica con estructura puede utilizarse en áreas de fácil acceso, en la que los sensores son desplegados de una manera planificada. Este tipo de redes tiene la ventaja de que puede implementarse con un número mucho menor de nodos y que tiene menores costos de mantenimiento y administración.

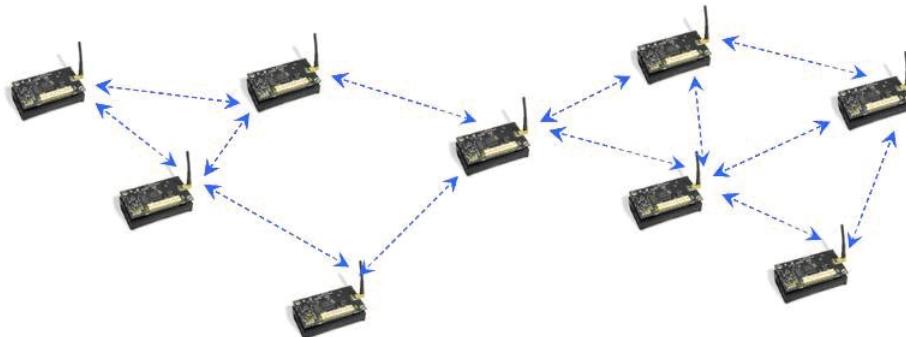


Figura 3.2: Red de sensores inalámbrica.

La gama de tareas de las redes de sensores inalámbrica, se puede clasificar en tres grupos. El primer grupo es el sistema. Cada nodo sensor es un sistema individual. A fin de apoyar a las diferentes aplicaciones en un sistema de sensores, el desarrollo de nuevas plataformas, sistemas operativos y sistemas de almacenamiento son necesarios. El segundo grupo son los protocolos de comunicación, que permiten la comunicación entre la aplicación y los sensores. El tercer y último grupo son los servicios que se desarrollan para mejorar las aplicaciones y el rendimiento del sistema. La información recabada por los sensores es generalmente procesada por los nodos. Sin embargo, debido a las limitaciones en memoria y capacidad de procesamiento, los nodos cuentan con una radio de baja potencia para transmitir de manera inalámbrica únicamente la información requerida y parcialmente procesada. Esta información es enviada a una estación base, por ejemplo, una computadora portátil o una tableta electrónica.

## 3.2. Clasificación

Como se mencionó anteriormente, dependiendo del entorno en el que se encuentre una red de sensores inalámbrica afronta desafíos y restricciones diferentes. A continuación se presenta la clasificación de las redes de sensores inalámbricas:

- **WSN terrestres:** Se caracterizan por contar con cientos o miles de sensores distribuidos en una zona determinada, su despliegue puede ser aleatorio o previamente programado. En su despliegue aleatorio los nodos sensores pueden ser lanzados desde un avión y se colocan al azar en la zona de destino. En el despliegue planificado previamente, existe la colocación en rejilla (grid), la colocación óptima 2-d y los modelos de colocación 3-d. Para una WSN terrestre, la energía se puede administrar utilizando el encaminamiento multi-salto, rangos de transmisión cortos y eliminando la redundancia de datos.
- **WSN subterráneas:** Consiste en nodos sensores enterrados bajo la tierra o en cuevas, y son utilizadas para monitorear las condiciones subterráneas. El desarrollo de una WSN subterránea es más caro que una WSN terrestre en términos de equipamiento, implementación y mantenimiento. Los nodos sensores subterráneos son caros porque las piezas de los equipos deben ser seleccionados apropiadamente para garantizar una comunicación fiable a través del suelo, las rocas, el agua y otros contenidos minerales.
- **WSN submarinas:** En comparación con el despliegue denso de nodos sensores en una WSN terrestre, en las WSN submarinas el despliegue de nodos de sensores colocados bajo el agua es poco denso. Las comunicaciones inalámbricas bajo el agua son establecidas a través de la transmisión de ondas acústicas. Uno de los retos en la comunicación acústica submarina es el ancho de banda limitado. El tema de la conservación de energía para redes de sensores inalámbricas bajo el agua, implica el desarrollo eficiente de la comunicación bajo el agua y las técnicas de creación de redes.
- **WSN multimedia:** Consisten en un número de nodos de sensores de bajo costo, equipados con cámaras y micrófonos. Desafíos en las WSN multimedia incluyen alta demanda de ancho de banda, alto consumo de energía, calidad de servicio (QoS), procesamiento y compresión de datos.
- **WSN móviles:** Una WSN móvil puede comenzar con poco despliegue y luego extenderse para recopilar información. Los nodos móviles pueden alcanzar un mayor grado de cobertura y conectividad en comparación con los nodos sensores estáticos. En pres-

encia de obstáculos en el campo, los nodos móviles pueden planificar el futuro y seguir adecuadamente sobre regiones obstruidas.

### 3.3. Estándar IEEE 802.15.4 y ZigBee

La Alianza ZigBee es una asociación de empresas que trabajan conjuntamente en el desarrollo de estándares y productos para la conexión de redes inalámbricas fiables y rentables, de bajo consumo energético. ZigBee es el nombre de la especificación de un conjunto de protocolos de comunicación de alto nivel sobre la base del estándar IEEE 802.15.4, el cual define las características de la capa PHY (física) y MAC (control de acceso al medio) para redes de área personal de bajo consumo o LR-WPAN (Low Rate Wireless Personal Area Network, por sus siglas en inglés), como se observa en la Fig. 3.3.

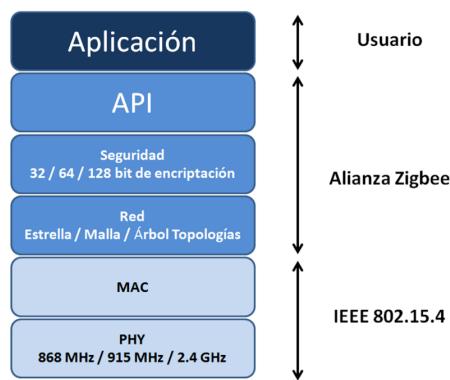


Figura 3.3: Arquitectura de ZigBee.

Los principales objetivos de una LR-WPAN son: facilidad de instalación, transmisión segura de datos, operación de corto alcance, muy bajo costo y duración de la batería razonable, manteniendo al mismo tiempo una pila de protocolos simple y flexible [7]. Algunas de las características de ZigBee incluyen:

- Operación global en la banda de frecuencia de 2,4 GHz de acuerdo con IEEE 802.15.4.
- Operación regional 915Mhz en América y 868Mhz en Europa.
- Más de 16 canales en la frecuencia de 2,4 GHz.
- Incorpora mecanismos de ahorro de energía para todas las clases de dispositivos.
- Mecanismo de descubrimiento de ruta con la confirmación plena de la aplicación.

- Mecanismo de generación de claves de seguridad.
- Utiliza el sistema de seguridad estándar AES-128.
- Soporta los estándares de la Alianza (perfiles de aplicación pública) o perfiles específicos del fabricante.

La capa de red define tres tipos de nodos: coordinador, encaminador y dispositivo final.

- **Coordinador:** El nodo coordinador o PAN coordinador, es el más sofisticado de los tres tipos, forma la raíz del árbol en la red y puede tender un puente a otras redes, por lo tanto, requiere mayor cantidad de recursos computacionales. Es un dispositivo con la capacidad de almacenar información de los nodos y administrar la red una vez establecida. Sólo puede existir un coordinador en cada red.
- **Encaminador:** Es un dispositivo de tipo FFD (Full Function Device) o dispositivo de funcionalidad completa. Actúa como nodo intermedio, transmitiendo los datos de otros dispositivos. Puede conectarse a una red ya existente. También es capaz de aceptar conexiones de otros dispositivos y ser una especie de retransmisor de paquetes dentro de la red. La red puede expandirse mediante el uso de encaminadores.
- **Dispositivo final:** Es un dispositivo de tipo RFD (Reduced Function Device) o dispositivo de funcionalidad reducida. Tiene la capacidad de recopilar información de los sensores y no pueden transmitir los datos de otros dispositivos, esto reduce su costo. Estos dispositivos no tienen que permanecer despiertos todo el tiempo, como es el caso de los dispositivos de tipo coordinador y encaminador. Cada red puede tener hasta 240 dispositivos finales.

Existen tres tipos de topologías de red que se consideran en IEEE 802.15.4: estrella, punto a punto y árbol, como se observa en la Fig. 3.4.

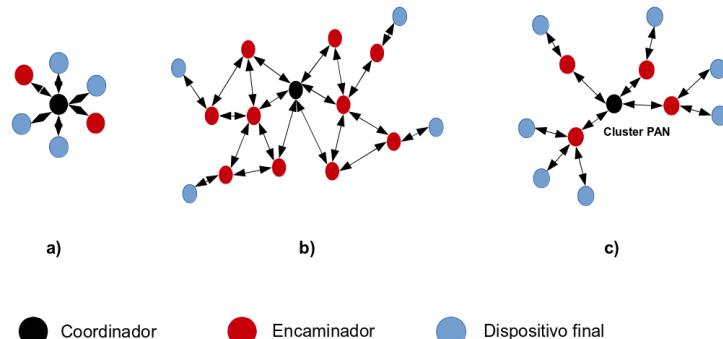


Figura 3.4: Topologías: a) Estrella b) Punto a punto c) Árbol

Entre las funcionalidades proporcionadas por la capa de red son el enrutamiento de saltos múltiples, el descubrimiento y mantenimiento de ruta, la seguridad e incorporación/descarte de los nodos en una red, con la asignación de direcciones cortas (16-bits) para dispositivos recién incorporados.

## 3.4. Fusión de información

Las redes de sensores inalámbricas generan una gran cantidad de datos para su procesamiento, envío y evaluación de acuerdo con los objetivos de la aplicación. La forma en que estos datos son manipulados por los nodos sensores es una cuestión fundamental. Como respuesta surge la fusión de información [8].

Fusión de información de acuerdo con Dasarathy [9] se refiere a: “... *la utilización de teoría, técnicas y herramientas creadas y aplicadas para explotar la sinergía en la información adquirida de múltiples fuentes (sensores, bases de datos, etc.) de tal manera que la decisión o acción resultante es en algún sentido mejor (cualitativa o cuantitativamente) de lo que sería posible si cualquiera de esas fuentes hubiese sido utilizada individualmente* “.

Las redes de sensores inalámbricas están destinadas a ser desplegadas en entornos donde los sensores pueden estar expuestos a condiciones que podrían interferir con sus mediciones. Las mediciones del sensor pueden ser imprecisas en escenarios tales como: fuerte variación de la temperatura y la presión, ruido electromagnético y la radiación. Aún cuando las condiciones ambientales sean ideales, los sensores no pueden proporcionar medidas perfectas. Esencialmente, un sensor es un dispositivo para obtener mediciones, y un valor de imprecisión se asocia generalmente con su observación. Tal imprecisión representa las imperfecciones de la tecnología y de los métodos utilizados para medir un fenómeno físico o una propiedad.

### 3.4.1. Clasificación

La fusión de información debe ser utilizada para mejorar el desempeño de una tarea mediante la comprensión de la situación actual, y el apoyo a las decisiones. La fusión de información se puede clasificar en función de varios aspectos.

#### 3.4.1.1. Basada en las relaciones de las fuentes

De acuerdo a la relación entre las fuentes la fusión de información se puede clasificar en:

- **Complementaria:** Cuando la información proporcionada por las fuentes representa diferentes partes de una escena más amplia, la fusión de información se puede aplicar

para obtener una pieza de información más completa. Por ejemplo, la temperatura de los lados este y oeste de la zona vigilada.

- **Redundante:** Si dos o más fuentes independientes proporcionan la misma información, estas piezas se pueden fusionar para aumentar la confianza asociada.
- **Cooperativa:** Dos fuentes independientes son cooperativas cuando la información proporcionada por ellas se fusiona en nueva información (por lo general más compleja que los datos originales) que, desde la perspectiva de la aplicación, es la que mejor representa la realidad.

#### 3.4.1.2. Basada en los niveles de abstracción

De acuerdo con el nivel de abstracción de los datos manipulados, la fusión de información se puede clasificar en:

- **Fusión de nivel bajo:** Los datos obtenidos (datos en bruto), se proporcionan como entrada, junto a un nuevo conjunto de datos que es más exacto (reducción de ruido) que las entradas individuales. Polastre [10] proporciona un ejemplo de la fusión de bajo nivel mediante la aplicación de un filtro de media móvil para calcular el ruido ambiental y determinar si el canal de comunicación es claro.
- **Fusión de nivel medio:** Los atributos o características de una entidad (por ejemplo, forma, textura, posición) se fusionan para obtener un mapa de características que puede ser utilizado para otras tareas (por ejemplo, la segmentación o la detección de un objeto). Este tipo de fusión también se conoce como *fusión a nivel de característica/atributo*.
- **Fusión de nivel alto:** En este nivel se toman las decisiones o representaciones simbólicas como entrada y se combinan para obtener una decisión global.
- **Fusión multinivel:** Cuando el proceso de fusión comprende los datos de diferentes niveles de abstracción, por ejemplo, una medición se fusiona con una característica para proporcionar una decisión, entonces la fusión se lleva a cabo.

#### 3.4.1.3. Basada en los datos de entrada y salida

Otra clasificación conocida que considera el nivel de abstracción es proporcionado por Dasarathy [11], en el que los procesos de fusión de información se clasifican basándose en el nivel de abstracción de la información de entrada y de salida. Dasarathy identifica cinco categorías:

- **Dato de entrada - Dato de salida:** En esta clase, la fusión de información se ocupa de los datos en bruto y el resultado son también los datos en bruto, aunque tal vez más precisa y fiable.
- **Dato de entrada - Característica de salida:** En esta clase, la fusión de información utiliza los datos en bruto para extraer características o atributos que describen una entidad, es decir, cualquier objeto, situación o abstracción del mundo real.
- **Característica de entrada - Característica de salida:** La fusión trabaja en un conjunto de características para mejorar/perfeccionar una característica o extraer nuevas.
- **Característica de entrada - Decisión de salida:** En esta clase, la fusión de información toma un conjunto de características de una entidad y genera una representación simbólica o una decisión.
- **Decisión de entrada - Decisión de salida:** Las decisiones pueden estar fusionadas con el fin de obtener nuevas decisiones o dar énfasis en las anteriores.

## 3.5. Desarrollo a la fecha

Los estudios realizados para clasificar las técnicas de desarrollo siguen ciertas perspectivas. Las cuales están clasificadas en técnicas para áreas residenciales, técnicas para eventos en bosques y técnicas para la detección temprana. En las técnicas para áreas residenciales se describen los estudios realizados a eventos que pueden llegar a ocurrir en lugares como edificios, lugares públicos o áreas donde se concentra una conglomeración de civiles. En las técnicas para eventos en bosques, se describen los estudios a eventos que llegan a afectar los bosques como son incendios, los cuales generan una gran pérdida ambiental. En las técnicas para la detección temprana, se mencionan algunos métodos que se espera llegaran a detectar y de cierta forma intentar predecir los eventos que pueden llegar a ocurrir con anticipación.

### 3.5.1. Técnicas para áreas residenciales

En [12] se propone un sistema de alarma de incendio para edificios de alto riesgo. La arquitectura de la red esta comprendida por detectores, repetidores y un centro local en cada piso del edificio, como se observa en la Fig. 3.5.

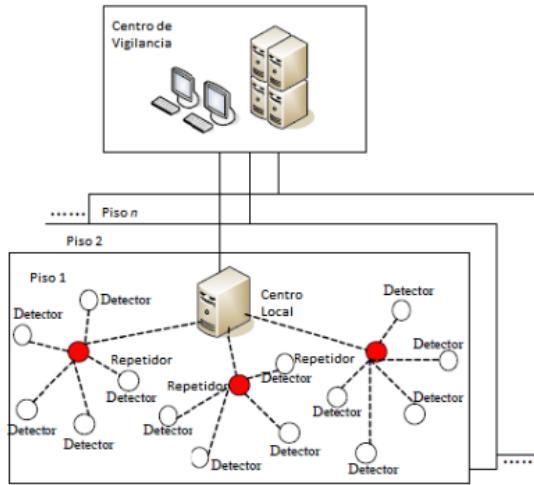


Figura 3.5: Arquitectura de red.

El prototipo fue realizado con 11 detectores, 3 repetidores y un centro de vigilancia distribuidos en un piso del edificio. Los detectores responsables de la detección de fuego y enviar los datos a los repetidores los cuales envían el reporte de la información monitoreada. Los repetidores no únicamente monitorizan si no que permiten tener acceso a la red de los detectores. La red puede estar comprendida por 255 repetidores y cada repetidor puede tener 255 detectores. Además cada repetidor tiene una dirección de red única, asignada por el centro de vigilancia.

El diseño de los detectores y repetidores utiliza el RF Chipset CC1100 de Texas Instrument. CC1100 es un transmisor de bajo costo diseñado para aplicaciones inalámbricas de bajo consumo de energía. El formato de las tramas de datos enviadas desde el transmisor se observa en la Tabla 3.1.

Preámbulo (1010 ... 1010)	Palabra de sincronización	Tamaño	Dirección	Datos	CRC-16
8xn bits	16/32 bits	8 bits	8 bits	8xn bits	16 bits

Tabla 3.1: Formato del paquete CC1100.

Debido a que el CC1100 consume mucha energía, únicamente se envían reportes ciertos períodos de tiempo. Los repetidores reciben un mensaje de confirmación el cual registra la información de un nuevo nodo, almacenando en una tabla de nodos vecinos el número serial, dirección, estatus, etcétera.

Los detectores reportan la información de detección a los repetidores una vez que se ha detectado una alarma de incendio. El formato que tiene el mensaje comprende el tipo de paquete, dirección origen, dirección destino, contador, estatus de alarma, concentración de humo y nivel de batería. Cuando el repetidor recibe el mensaje del detector; el repetidor envía un mensaje al centro de vigilancia con el siguiente formato: tipo de paquete, dirección fuente, dirección destino, período de reporte, período de activación, alarma y reset. Cuando el mensaje es recibido por el repetidor, este envía un mensaje de confirmación al detector. Para la detección se utiliza un sensor de humo fotoeléctrico y/o de temperatura.

En [13] se propone un framework para la detección de incendios y apoyo en labores de rescate para áreas civiles como por ejemplo: tiendas departamentales, estaciones de tren, construcciones públicas. Se realizaron experimentos para encontrar el intervalo de sensado idóneo de acuerdo con el número de sensores. El despliegue de la red es externo (al aire libre), lo cual dificulta el manejo de la red directamente, por lo que el framework permite un control remoto indirecto. El framework, sugiere 4 componentes en la implementación:

1. **Red de sensores inalámbrica para la detección de incendios:** Cada nodo tiene la función de monitorear y recolectar datos de temperatura, iluminación, humo, etcétera. Estos datos son utilizados para determinar un evento de incendio.
2. **Capa de recolección de información:** El principal propósito es entregar la medición y los eventos de alarma de incendio a la estación base. Cuando el incendio ocurre, la infraestructura puede ser destruida fácilmente, por lo que cada nodo se puede apoyar en los nodos vecinos para la entrega de información.
3. **Middleware:** Se encuentra en la terminal (computadora) conectada a la estación base. Los datos obtenidos son mostrados en la terminal. El middleware puede enviar la información hacia otros programas como es el software de monitoreo.
4. **Software de monitoreo:** Aquí se muestran los experimentos y demostraciones.

Para la implementación los nodos cuentan con un algoritmo de operación, reciben mensajes de otros nodos, pueden tomar decisiones, y pueden notificar las áreas que son seguras (rutas de escape), todas las notificaciones son enviadas a la estación base.

Para los experimentos fue definido el IMR (Razón del intervalo de mensajes o Interval Message Ratio, por sus siglas en inglés). El IMR se calcula como se observa en la ecuación 3.1:

$$IMR = \frac{\text{Intervalo de mensajes generados}}{\text{Intervalo de mensajes obtenidos por el middleware}} \quad (3.1)$$

Para las pruebas se utilizo únicamente el valor de la temperatura. Con lo que el intervalo ideal para obtener los mensajes en el experimento es de 1250 y 1500 milisegundos, logrando evitar una sobresaturación de la red como se observa en la Fig. 3.6.

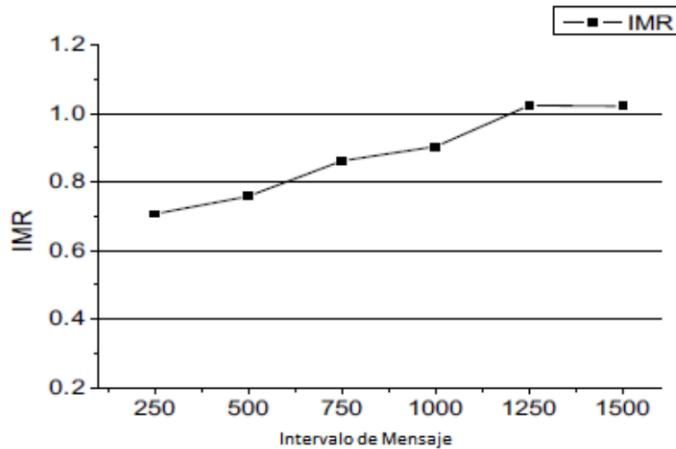


Figura 3.6: Resultado de los experimentos.

Para interiores de edificios [14] se utiliza un monitoreo en el que los sensores se comunican con una estación central. Los usuarios puede interactuar con la red de sensores de dos maneras:

1. Los usuarios remotos pueden acceder a una replica del servidor base en Berkeley.
2. Se puede utilizar un pequeño dispositivo parecido a un PDA para interacciones locales.

Con la ayuda del servidor de base de datos más aplicaciones se pueden integrar para monitorear el sistema. Los datos leídos son enviados al servidor central vía red GSM. El servidor central esta conectado con una base de datos en línea. Desde esta base de datos una aplicación remota puede ser ejecutada vía web y accesada desde una computadora, laptop o PDA. El modelo del sistema de monitoreo utiliza el protocolo ZigBee el cual esta diseñado específicamente para monitorización y control. La distancia para monitorización dentro de edificios se encuentra en el rango de 25 a 30 metros. La estación base continuamente se encuentra monitoreando a una frecuencia de 2.4 GHz, cuando detecta un paquete detiene la monitorización y toma al paquete y lo envía a la computadora a 57,6000 bps vía cable serial. La comunicación serial es controlada por UART (Transmisor/Receptor Asíncrono Universal). El servidor que realiza las alertas esta escrito en el lenguaje de programación Java de Oracle [15]. La alerta consiste en un mensaje que informa el área donde se detecto fuego y el programa reproduce un archivo. Cuando el sensor se detiene de enviar paquetes durante un tiempo tolerable, el programa concluye una falla y dice que es necesaria una futura prueba. El intervalo asignado es de 1 minuto.

El mensaje que llega a la estación base contiene el identificador del nodo, un contador, grupo del mensaje, lectura de la luz y lectura de la temperatura. La base de datos lleva un historial de las lecturas registradas. Los resultados obtenidos del sistema dan un eficiente monitoreo y habilitan la comunicación de los sensores con una estación central.

En [16] se propone la arquitectura para una red de sensores inalámbrica llamada FireNet para el apoyo a cuerpos de bomberos. La arquitectura de FireNet como se observa en la Fig. 3.7 consiste en:

- Departamento de bomberos.
- Vehículos de rescate donde se encuentra la estación base.
- Cada bombero posee un nodo sensor con sus datos (nombre, edad, especialidad, temperatura, humedad, entre otros datos).

En primer instancia el software necesita analizar la información recolectada del fuego para detectar eventos, después estos datos son enviados al comandante de incidentes y al departamento de bomberos. Después el software genera automáticamente la planificación del proceso de rescate basándose en los datos recolectados, por último el software notifica al comandante de incidentes cuando eventos importantes son detectados. Cada bombero lleva un sensor el cual toma las mediciones de la temperatura, humedad, químicos.

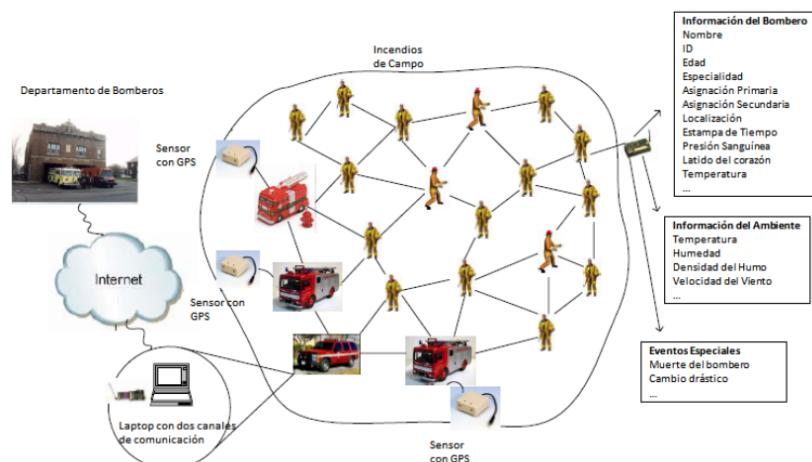


Figura 3.7: Arquitectura del sistema FireNet.

### 3.5.2. Técnicas para eventos en bosques

Para la detección de fuego en bosques se ha llegado a implementar redes neuronales para prolongar el tiempo de vida de la red de sensores. En [17] los nodos se ubican en el bosque toman los datos y los envían a un cluster el cual a su vez los envía a la estación base, como se observa en la Fig. 3.8.

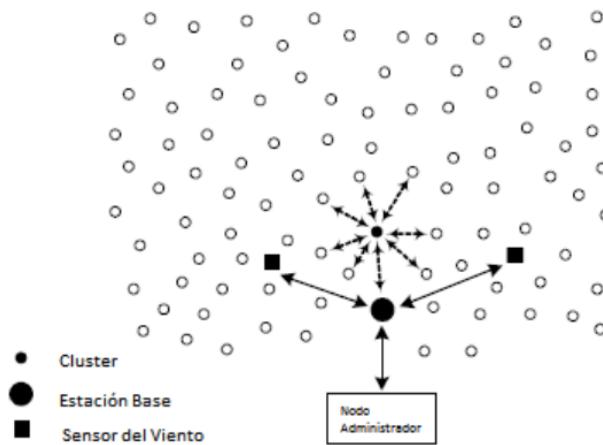


Figura 3.8: Red de sensores inalámbrica para la detección de incendios forestales en tiempo real.

Para el funcionamiento de la red se puede reducir la sobrecarga de mensajes y el consumo de energía. Madden [18] propuso que la gestión de datos en una red se pueda manipular a través de un estilo SQL. Este aprovecha que es posible ejecutar consultas en los sensores y proveer oportunidades de optimización. La propuesta para la detección de incendio consiste en un diseño que detecta fuego en el bosque a tiempo y previniendo peligros de incendio en el bosque de manera más precisa. Los nodos toman los datos de la temperatura, humedad relativa y humo, algunos cuentan con un módulo GPS. El cluster se selecciona dinámicamente por un algoritmo para dar un balance en el consumo de la energía de todos los nodos. Para la colección de datos cuando los nodos encuentran algún evento anormal, entonces inmediatamente generan y envían un paquete ER (Emergence Report) con la información del evento anormal. Para el procesamiento de los datos cuando la estación base recibe un ER lo considera de alta prioridad, si recibe un mensaje QR (Query Response) el cluster aplica un algoritmo de agregación. Para el procesamiento de paquetes RR (Regular Report) se construye un índice de clima, el índice se encapsula en un PR (Processed Report), el PR se envía por el cluster a la estación base, como se observa en la Fig. 3.9.

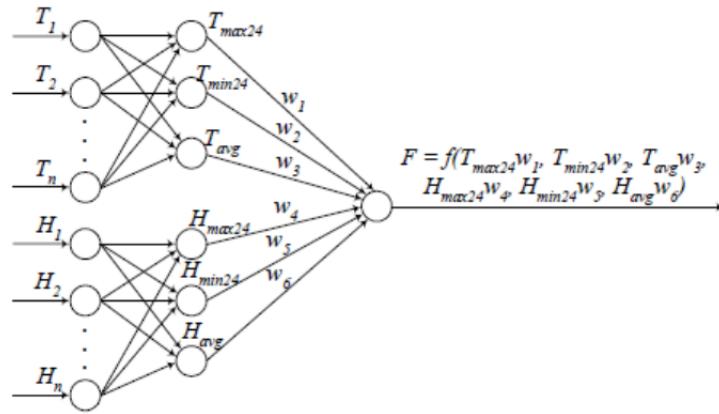


Figura 3.9: Procesamiento de Paquete RR utilizando el método de redes neuronales.

Se utilizan redes neuronales para prolongar el tiempo de vida de la red de sensores. Se comenta que los sensores pueden ser utilizados para ayudar en el pronóstico de incendio en bosques apoyándose con imágenes satélites.

El sistema de Monitoreo FWI (Fire Weather Index) [19] toma en consideración la temperatura, la humedad relativa, el viento y la lluvia. En base a estos datos se genera un valor el cual se compara con el código FFMC Ignition Potencial Based. El sistema FWI cuenta con buenos avances para monitoreo en bosques. La estructura del Sistema FWI se observa en la Fig. 3.10.

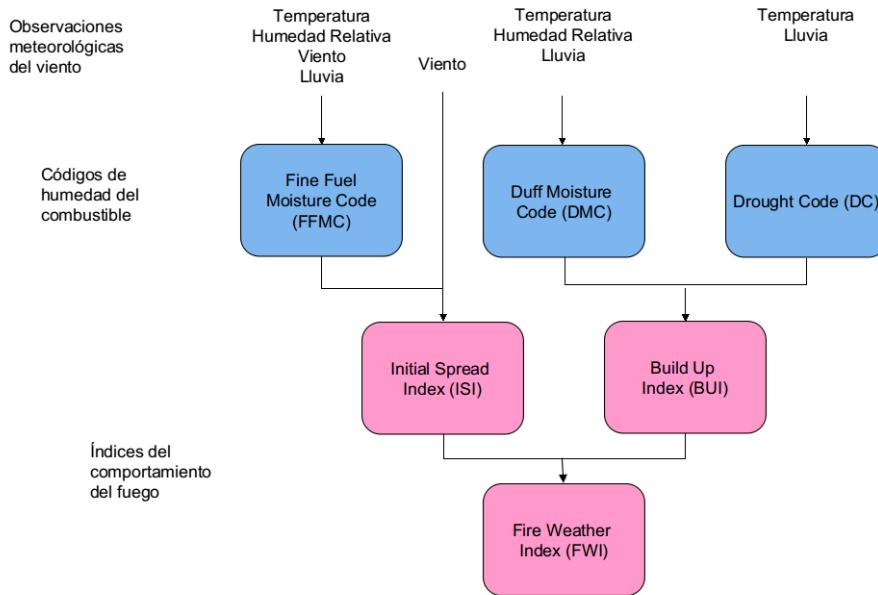


Figura 3.10: Estructura del sistema FWI.

En [20] se proponen dos métodos para la detección de incendios forestales:

1. **Método del umbral:** Método con el que se proponen 7 estados, considerando un estado cero como inicio; obteniendo la información de la base los datos acerca de la temperatura y la humedad relativa.
2. **Método de Dempster–Shafer:** Método basado en hipótesis, cuando existe fuego y cuando no existe fuego.

Para la adquisición de datos en los dos métodos se realizaron experimentos con los nodos sensores expuestos a fuego artificial (antorchas) con una duración de 5 a 15 minutos y distancias de 50 a 100 centímetros. En los experimentos lo más importante a destacar son los valores de falsos negativos (realizar experimento y no ser detectado) los cuales se consideran con mayor relevancia, que los falsos positivos (no realizar experimento y ser detectado).

Los dos métodos mostraron resultados muy diferentes en los que se encuentra que el método del umbral tiene menos falsos negativos en los eventos. El porcentaje de aciertos es mayor en el método del umbral. Se recalca en el artículo que aunque el método del umbral resultó con mayores resultados en cuanto a aciertos, el método Dempster-Shafer muestra una ventaja en permitir una extensión rápida de nuevas evidencias, cargándose con mayor eficiencia. Se busca en un futuro lograr distinguir la luz de los rayos del sol con la luz del fuego, debido a que por esta razón presentó falsos negativos en las pruebas. La máquina de estados utilizada con el método del umbral se observa en la Fig. 3.11.

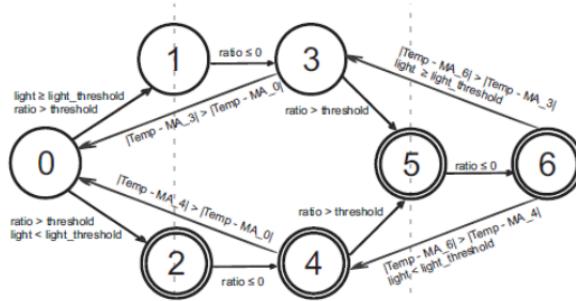


Figura 3.11: Máquina de estados para definir las condiciones del ambiente.

Para la monitorización en bosques en [21], se realizó un experimento de campo donde se construyeron bases y se montaron los nodos sensores (MTS420), Mica2 con GPS. Cada nodo obtiene datos de temperatura, humedad relativa, presión y posición geográfica. Se realizó un experimento donde se colocaron 10 nodos sensores, 6 de ellos dentro del área de incendio y 4 próximos al fuego. La arquitectura de del sistema se observa en la Fig. 3.12.

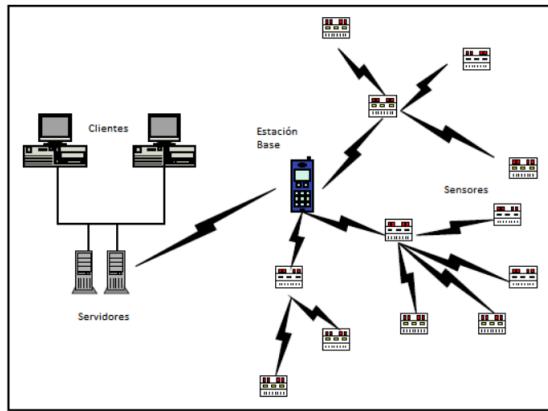


Figura 3.12: Arquitectura del sistema de monitoreo.

Los datos son enviados a la estación base para almacenarlos en un servidor base de datos. Se realizaron dos tipos de experimentos en los cuales se llegaron a destruir algunos nodos, mientras que otros dejaron de enviar datos debido a que se cortó la alimentación de la batería. Cuando ocurrió el evento de incendio se observa que la humedad y la presión atmosférica disminuyen mientras que la temperatura aumenta. Como observación se destaca que se requieren 4 minutos para que la presión barométrica regrese a su valor inicial.

En [22] se describe un componente llamado Local Alerting Control Unit (LACU), así como un método para la estimación de la localización del fuego. La arquitectura de LACU está dividida en cinco componentes:

1. Proxy de comunicación.
2. Proxy sensor.
3. Componente de fusión.
4. Componente de alerta.
5. Componente de base de datos.

El proxy de comunicación es el responsable de la comunicación entre LACU y el subsistema de cómputo. El proxy sensor recibe y almacena las lecturas transmitidas por los sensores distribuidos en el área a monitorear. El componente de fusión evalúa los datos medidos y determina si llega a ocurrir un incendio. El componente de alerta es activado por el componente de fusión, provee notificaciones al subsistema de cómputo y a los usuarios cuando una situación de emergencia ocurre. El componente de base de datos almacena el histórico, de la

identificación de cada sensor y la localización que se provee por el componente proxy sensor. Los sensores son ubicados en dos áreas:

- **Public LACU:** Instalados y operados por autoridades públicas.
- **Private LACU:** Son instalados por los ciudadanos para proteger sus propiedades privadas.

Para la detección de fuego se probó con una red de sensores, tomando en cuenta las lecturas de la temperatura obteniendo buenos resultados. En ocasiones suele dar falsos positivos principalmente en temporadas calurosas como en verano. Para la estimación de la localización del fuego es necesario las lecturas de 3 nodos para mejorar la estimación.

En [23] se describe una aplicación WUI (Wildland Urban Interface). La arquitectura del sistema SCIER esta compuesto por 3 capas:

- SS (Sensing Subsystem).
- LAS (Localized Alerting Subsystem).
- CS (Computing Subsystem).

En la capa SS del sistema SCIER se encuentra la red de sensores inalámbrica que toma los valores de la temperatura y la humedad relativa y son utilizados para la detección de incendio. Además se encuentran cámaras que toman imágenes de la escena para después aplicar algoritmos para detectar humo o llamas. En la capa LAS se encuentra LACU (Local Alerting Control Unit) donde se controla un área de sensores y se aplican los algoritmos para la detección de incendio como se observa en la Fig. 3.13.

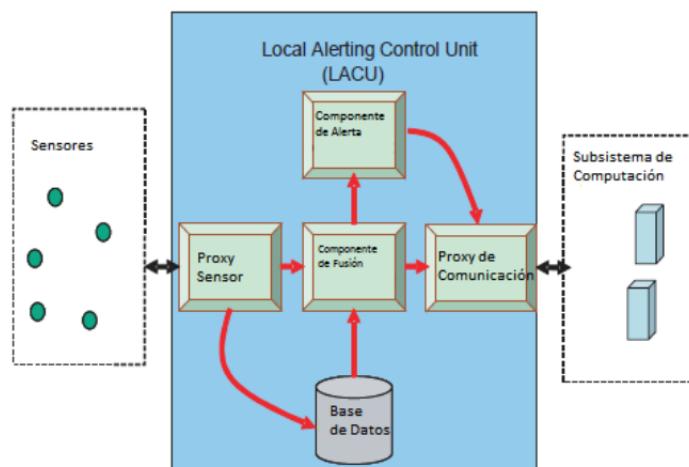


Figura 3.13: Arquitectura de LACU.

LACU es el intermediario entre SS y CS. LACU esta compuesto por:

- Proxy de comunicación: Es el responsable de la comunicación entre LACU y CS.
- Proxy sensor: Es el encargado de recibir y almacenar los datos de los sensores.
- Componente de fusión: Es donde se evalúan los datos de los sensores y se determina si existe fuego en el área. Además estima la localización exacta del fuego.
- Componente de alerta: Este es lanzado por el componente de fusión y provee notificaciones a CS cuando la situación de emergencia ocurre.
- Componente de base de datos: Se encuentra el histórico de los datos, la identificación y localización de cada nodo, los datos son obtenidos del proxy sensor.

En gran parte CS esta basado en infraestructura computacional GIS (sistema de información geográfica), donde se fusiona la información almacenada de los sensores, se procesa y visualiza usando datos adicionales del clima (velocidad del viento, dirección). Las principales funciones de CS son:

- Coleccionar y almacenar las mediciones de los sensores del área de interés.
- Procesar algoritmos de fusión de datos para evaluar el nivel de riesgo.
- Lanza una simulación en caso de alarma, mostrando lo que se estima ocurra en los siguientes 180 minutos.

La fusión de datos se lleva a cabo considerando el caso donde la temperatura en ambiente normal aumenta, de una manera anormal. La fusión de datos se lleva en dos niveles. En el primer nivel se toma en cuenta la media de las temperaturas dependiendo de la hora, fecha y mes del año. En el caso de incendio la humedad disminuye al contrario de la temperatura que incrementa. Cuando se detecta riesgo de fuego en la primera capa, la segunda capa realiza el proceso de fusión con datos del sensor de visión (cámara) y los datos que le provee LACU.

Los nodos sensores toman los datos de la humedad y temperatura y son enviados al proxy sensor que se encuentra en LACU. El mecanismo propuesto esta basado en fusión de datos multinivel. Para acoplar los diferentes tipos de sensores y entregar alarmas con gran acierto y confianza se adapto un esquema de fusión en capas.

En [24] se propone un mecanismo de detección de eventos utilizando reglas de lógica difusa. La lógica difusa permite valores intermedios, definidos entre las evaluaciones convencionales como por ejemplo: verdadero/falso, sí/no, alto/bajo, etcétera. Las reglas de lógica

difusa son utilizadas para obtener la probabilidad de incendio considerando las variaciones en la información obtenida por los nodos sensores durante las diferentes etapas del día. Cada nodo sensor detecta la temperatura, humedad relativa, intensidad de la luz, densidad de *CO* (monóxido de carbono) y el tiempo. La red de sensores está dividida en cluster's. Esto permite una mejor administración del consumo de energía en la red. Para la evaluación del sistema se utilizaron simulaciones desarrolladas en MATLAB. Por ejemplo, si los datos de entrada son [89 68 100 76 14] donde cada valor corresponde a la temperatura, intensidad de la luz, humedad relativa, densidad de *CO* y tiempo respectivamente. El resultado obtenido es una probabilidad de 90 % en un horario de las 2 pm, como se observa en la Fig. 3.14.

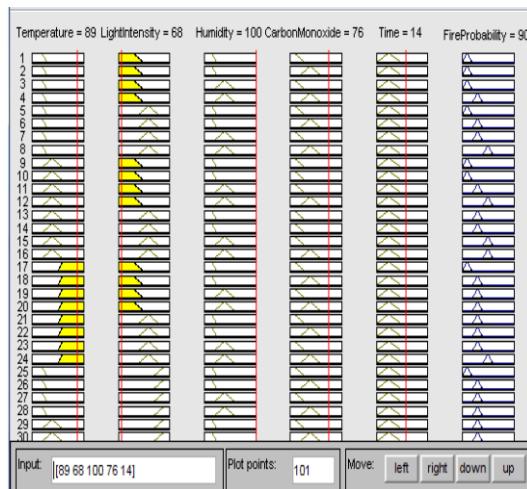


Figura 3.14: Sistema de detección de incendios basado en reglas de lógica difusa.

En [25] se propone un framework para la detección de incendios cuyo diseño considera todas las partes del ciclo de vida de una red de sensores inalámbrica. Consiste en las siguientes partes:

1. Un esquema de despliegue de los nodos sensores.
2. Una arquitectura de red de tipo cluster.
3. Un protocolo de comunicación intra-cluster (dentro del cluster).
4. Un protocolo de comunicación inter-cluster (entre clusters).

El objetivo del framework es detectar una amenaza de incendio en el menor tiempo posible y además administrar adecuadamente el consumo de energía de los nodos sensores considerando las condiciones ambientales que puedan afectar el nivel de actividad requerido de la red.

Para la evaluación del framework se diseño e implementó un simulador en lenguaje C# de la plataforma Visual Studio 2008 de Microsoft [26] como se observa en la Fig. 3.15.

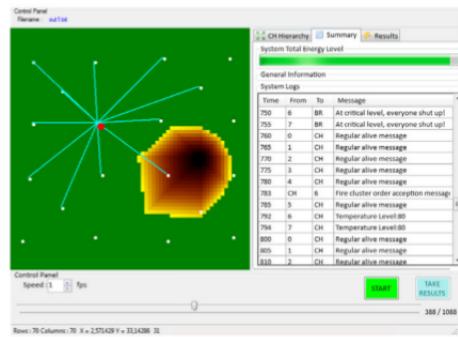


Figura 3.15: Simulador de eventos de incendio.

Se realizaron experimentos exhaustivos para evaluar el framework en términos de consumo de energía y la eficacia en la detección de incendios.

En [27] proponen un entorno de simulación EIDOS (Equipos destinados a orientación y seguridad) que puede crear un modelo de un incendio mediante el análisis de los datos reportados por los nodos sensores y mediante el uso de información geográfica sobre la zona. La estimación de la propagación de un incendio se envía a los dispositivos de mano de los bomberos para ayudarles en la lucha contra el fuego, como se observa en la Fig. 3.16.

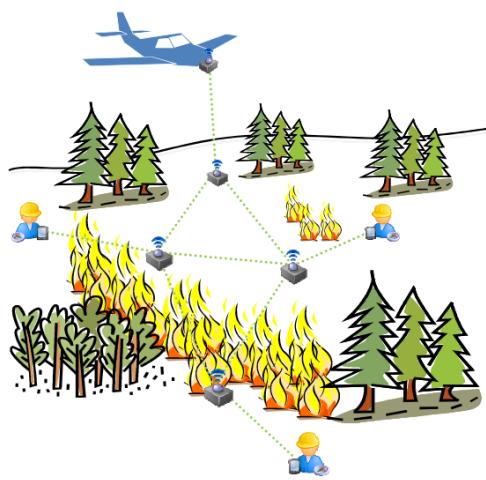


Figura 3.16: Sistema EIDOS para combate de incendios forestales.

### 3.5.3. Técnicas para detección temprana

En [28] se propone un algoritmo para la detección de incendio tomando los valores de la temperatura,  $CO$  y densidad del humo, utilizando tecnología de fusión de datos. Esta compuesto por tres capas como se observa en la Fig. 3.17:

1. Capa de señales.
2. Capa de características.
3. Capa de decisión.

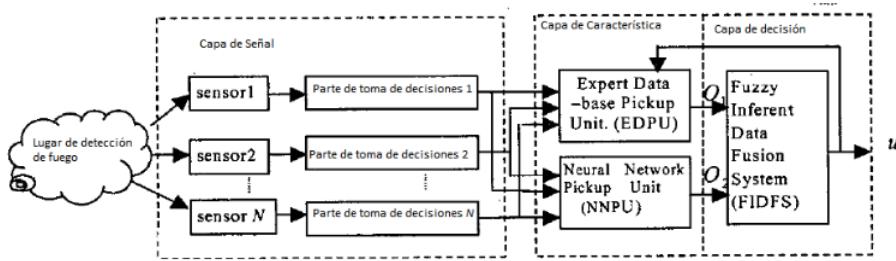


Figura 3.17: Estructura de las tres capas del sistema para la detección de fuego.

La primera capa toma y procesa los datos. La segunda capa toma la característica del fuego de los datos de la primera capa, almacenándolos en una base de datos. La tercera capa fusiona las diferentes características de fuego obtenidas de la segunda capa y toma una decisión de la probabilidad de fuego. Los nodos son simulados en MATLAB, se combina con datos de la estandarización de llamas de fuego en China llamado SH4. El algoritmo para la fusión de datos de fuego se basa en una red neuronal la cual se encuentra en la segunda capa. Después de hacer las simulaciones utilizando las 3 capas se concluye que es posible predecir el fuego en un ambiente de llamas de fuego o algodón ardiendo.

# Capítulo 4

## TinyOS y nesC

Para poder satisfacer los requerimientos de una red de sensores inalámbrica es importante contar con una plataforma de software específica para el manejo de este tipo de dispositivos. En este capítulo se describen las características del sistema operativo TinyOS.

### 4.1. TinyOS

TinyOS es un sistema operativo diseñado específicamente para trabajar con redes sensores inalámbricas, fue desarrollado por la Universidad de Berkeley en California, originalmente como un proyecto de investigación. TinyOS está escrito en lenguaje nesC y está liberado bajo licencia BSD (Berkeley Software Distribution) como software Open Source (código abierto). Su diseño tiene dos objetivos principales: reducir al mínimo el uso de recursos y la prevención de errores. TinyOS proporciona tres elementos de alto nivel con la finalidad de simplificar el desarrollo de sistemas y aplicaciones [29]:

1. **Modelo de componentes:** El modelo de componentes, define cómo escribir pequeñas y reutilizables piezas de código, así como la manera de integrarlas para construir abstracciones de mayor tamaño.
2. **Modelo de ejecución concurrente:** El modelo de ejecución concurrente, define la manera en que los componentes pueden entrelazar sus ejecuciones, y la manera en que el código interactúa con interrupciones y sin interrupciones.
3. **Interfaces de programación de aplicaciones (API):** Define servicios, bibliotecas de componentes, así como una estructura general que simplifica la escritura de nuevas aplicaciones y servicios.

## 4.2. nesC

nesC [30] es una variante del lenguaje de programación C y ha sido diseñado para ajustarse a los conceptos estructurales y modelo de ejecución de TinyOS. nesC tiene como objetivos el desarrollo de aplicaciones que reduzcan el tamaño del código y de la utilización de memoria RAM. Los conceptos básicos de nesC se explican a continuación:

- Separación de construcción y composición: Los programas son construidos a partir de componentes, los cuales son enlazados (*wired*). Los componentes son definidos en dos ámbitos, uno para la configuración (contiene los nombres de las instancias de las interfaces) y uno para la implementación. Los componentes tienen concurrencia interna en forma de tareas (tasks). Los hilos de control pueden pasar en un componente a través de las interfaces. Estos hilos tienen sus raíces en una tarea o en una interrupción por hardware.
- Especificación del comportamiento de los componentes en términos del conjunto de interfaces. Las interfaces pueden ser proporcionadas (*provide*) o utilizadas (*use*) por los componentes. Las interfaces buscan representar la funcionalidad que proporciona el componente al usuario. Las interfaces utilizadas representan la funcionalidad que el componente requiere para desempeñar su trabajo.
- Las interfaces son bidireccionales: Especifican un conjunto de funciones que serán implementadas por el proveedor de la interfaz (*commands*) y un conjunto de funciones que serán implementadas por el usuario de la interfaz (*events*). Esto permite que una interfaz represente una compleja interacción entre componentes (por ejemplo, el registro del algún evento, seguido de una notificación cuando el evento ocurre). Esto es crítico debido a que todos los comandos de larga duración en TinyOS, por ejemplo el envío de un paquete, son operaciones *sin bloqueo*; su terminación es notificada a través de un evento (envío realizado).
- Los componentes están ligados estéticamente entre sí a través de sus interfaces. Esto incrementa la eficiencia en tiempo de ejecución, promueve un diseño robusto y permite un mejor análisis estático de los programas.
- nesC ha sido diseñado bajo la premisa de que el código será generado por compiladores de programas completos. Esto también debe permitir una mejor generación de código y análisis del mismo.

### 4.2.1. Convenciones del lenguaje nesC

- Todos los archivos escritos en nesC deben terminar con una extensión **.nc**. Por ejemplo: BlinkAppC.nc.
- El compilador nesC requiere que el nombre del archivo sea el mismo nombre para evento principal del componente, por ejemplo BlinkAppC.nc debe contener como método principal **configuration BlinkAppC** si se trata de un componente de configuración o **module BlinkAppC** si se trata de un componente de implementación.
- Los nombres de los componentes pueden ser combinación de letras mayúsculas, minúsculas y números, comenzando con una letra mayúscula.
- Todos los componentes públicos deben terminar con el sufijo '**C**', por ejemplo BlinkAp-pC.
- Todos los componentes privados deben terminar con el sufijo '**P**'.
- Si dos componentes están relacionados, es útil si tienen el mismo nombre, excepto para el sufijo del componente.
- El nombre de los comandos, eventos, tareas y funciones pueden ser combinación de letras mayúsculas y minúsculas, comenzando con una letra minúscula.
- Las constantes deben escribirse siempre en mayúsculas, y si son más de dos palabras, entonces deben estar separadas por un guion bajo.
- Al especificar interfaces, por ejemplo, un componente no puede invocar el comando **send** al menos que proporcione una implementación del evento **sendDone**.
- nesC define los siguientes tipos de datos, además de permitir los tipos de datos de C:
  - **uint8\_t:** Es un entero sin signo de 8 bits.
  - **uint16\_t:** Es un entero sin signo de 16 bits.
  - **bool:** Es un dato de tipo booleano. Sus valores pueden ser TRUE o FALSE (sólo se pueden escribir en letras mayúsculas).
  - **result\_t:** Es un dato de tipo booleano. Sus valores pueden ser SUCCESS o FAIL (sólo se pueden escribir en letras mayúsculas).

### 4.3. Instalación de TinyOS

En esta sección se explica el proceso de instalación de TinyOS en la distribución de Linux, Ubuntu 12.04. El primer paso consiste en agregar algún repositorio que contenga TinyOS. Por ejemplo en [31], mantienen un repositorio de TinyOS para las siguientes versiones de Ubuntu:

edgy 6.10	hardy 8.04	lucid 10.04
feisty 7.04	jaunty 9.04	maverick 10.10
gusty 7.10	karmic 9.10	natty 11.04

Sin embargo, a pesar de que nuestra versión de desarrollo fue *precise* (12.04), al agregar el repositorio para la distribución *natty*, no se presento problema alguno en la instalación de los paquetes. Por lo tanto, el segundo paso consiste en agregar en el archivo **sources.list** la siguiente línea:

```
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu/ natty main
```

esto puede realizarse ejecutando desde una terminal (línea de comandos) el siguiente comando:

```
$ sudo nano /etc/apt/sources.list
```

el tercer paso, consiste en actualizar el listado de paquetes disponibles en los repositorios. Esto puede hacerse utilizando el siguiente comando:

```
$ sudo apt-get update
```

finalmente, para poder instalar TinyOS en su versión más reciente, debemos ejecutar el siguiente comando:

```
$ sudo apt-get install/tinyos-2.1.2
```

una vez instalado TinyOS debemos realizar algunos ajustes en la configuración del sistema. Es recomendable definir en las variables de entorno del sistema, el directorio en el que se encuentran los archivos importantes de TinyOS, tales como componentes, interfaces, bibliotecas, entre otros. Una manera de hacerlo es ejecutando desde una terminal lo siguiente:

```
$ sudo nano ~/.bashrc
```

tomando en cuenta que la ruta de instalación por defecto es dentro del directorio **opt**, las variables se definen en la parte final del documento de la siguiente manera:

```
TOSROOT="/opt/tinyos-2.1.2"  
TOSDIR="$TOSROOT/tos"  
CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java:.  
MAKERULES="$TOSROOT/support/make/Makerules"  
  
export TOSROOT  
export TOSDIR  
export CLASSPATH  
export MAKERULES
```

guardamos los cambios, cerramos el archivo y la terminal. La próxima vez que utilicemos una terminal, nuestras variables de entorno previamente declaradas serán consideradas como parte de la configuración del sistema.

### 4.3.1. Estructura de directorios

Por defecto la distribución de TinyOS contiene los siguientes directorios [32]:

- **tos/system:** Almacena los componentes básicos de TinyOS. Componentes de este directorio son necesarios para ejecutar TinyOS efectivamente.
- **tos/interfaces:** Contiene las interfaces principales de TinyOS, incluyendo abstracciones independientes del hardware.
- **tos/platforms:** Contiene código específico para plataformas de hardware para nodos sensores (por ejemplo iris, micaz, telosb).
- **tos/chips:** Contiene código específico para los chips en plataformas específicas (por ejemplo Atm1281, CC2420, CC1000).
- **tos/libs:** Contiene interfaces y componentes que amplían la utilidad de TinyOS pero que no son vistos como esenciales para su funcionamiento.
- **apps:** Contiene aplicaciones con ciertos objetivos demostrativos para la evaluación del algún componente en particular como temporizadores, transmisión de datos de forma inalámbrica, entre otros.

### 4.3.2. Configuración adicional

Puede ocurrir que la versión utilizada de TinyOS no incluya el soporte para la base programadora utilizada, o que la información de ésta sea incorrecta en el sistema. Por ejemplo, en nuestro caso, se utilizó la base programadora *mib520* (más adelante se describe el tipo de hardware utilizado en este proyecto), y fue necesario modificar el número **UsbProduct** igual a 6001, como se observa en la Fig. 4.1 dentro del archivo *motelist* ubicado en **/usr/bin/**

```
# Scan /sys/bus/usb/drivers/usb for FTDI or CP210X devices
my @ftdidevs =
| grep { (($_->{UsbVendor}|| "") eq "0403" && ($_->{UsbProduct}|| "") eq "6001") }
```

Figura 4.1: Sección por defecto en motelist.

reemplazándolo por 6010, como se observa en la la Fig. 4.2.

```
# Scan /sys/bus/usb/drivers/usb for FTDI or CP210X devices
my @ftdidevs =
| grep { (($_->{UsbVendor}|| "") eq "0403" && ($_->{UsbProduct}|| "") eq "6010") }
```

Figura 4.2: Sección modificada en motelist.

Para comprobar que los cambios han surtido efecto, una vez conectada nuestra base programadora escribimos desde la terminal lo siguiente:

```
$ motelist
```

si nuestro cambio fue satisfactorio, obtendremos como resultado un mensaje como el que se presenta en la Fig. 4.3.

Reference	Device	Description
XBTMXYSA	/dev/ttyUSB0	Crossbow Crossbow MIB520CA

Figura 4.3: Reconocimiento del dispositivo.

Una vez detectado el dispositivo, es posible comenzar a desarrollar aplicaciones para redes de sensores inalámbricas con TinyOS.

## 4.4. Desarrollo de aplicaciones en TinyOS

El objetivo de esta sección no es profundizar demasiado en los conceptos de nesC, sino presentar de una manera práctica la forma de construir aplicaciones para redes de sensores. En esta sección, se presentan dos tipos de aplicaciones: la primer aplicación, ejecuta un evento de encendido y apagado periódico de leds (diodos emisores de luz), mientras que la segunda aplicación cumple con un objetivo más atractivo, se muestra la forma de obtener información del medio ambiente y transmitirla a una estación base para su procesamiento.

### 4.4.1. Aplicación Blink

Blink es una aplicación sencilla para comenzar a trabajar bajo el paradigma de programación orientada a eventos de TinyOS. En programación orientada a objetos Blink, sería él “Hola Mundo”, es decir, una rutina simple de bienvenida a TinyOS. El objetivo es ejecutar una acción de parpadeo intermitente en los leds del nodo sensor. Físicamente el nodo sensor cuenta con: un led color rojo (led0), un led color verde (led1) y un led color amarillo (led2). Manipular leds es una forma de depurar el código en cualquier aplicación desarrollada en nesC. El diagrama de componentes de Blink incluye:

- **MainC**: Establece el inicio de la aplicación.
- **LedsC**: Es útil para la manipulación de leds en el nodo sensor.
- **TimerMilliC**: Es un temporizador, que recibe como parámetro una cantidad en unidades de milisegundos.
- **BlinkC**: Es una referencia al componente de implementación, como se observa en la Fig. 4.4.

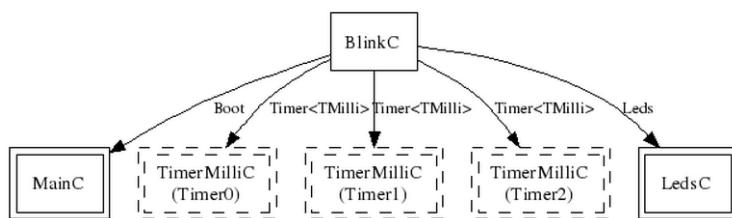


Figura 4.4: Diagrama de componentes de Blink.

Para comenzar a desarrollar la aplicación, debemos crear una carpeta para el almacenamiento del código ejecutando desde la terminal el siguiente comando:

```
$ mkdir Blink
```

el primer componente a crear dentro del directorio es **BlinkAppC.nc**, en el cual por el momento sólo se establece la declaración de los componentes necesarios para cumplir con el objetivo de la aplicación, como se observa en la Fig. 4.5:

```
configuration BlinkAppC { }
implementation
{
    components MainC, BlinkC, LedsC;
    components new TimerMilliC() as Timer0;
    components new TimerMilliC() as Timer1;
    components new TimerMilliC() as Timer2;
}
```

Figura 4.5: Archivo de configuración.

el siguiente paso es crear el archivo **BlinkC.nc**, en el cual se define la implementación de los componentes declarados en BlinkAppC.nc, comenzando por importar la librería “**Timer.h**” (necesaria para implementar un temporizador en nesC) y declararando las interfaces a utilizar con la palabra reservada **use interface**, como se observa en la Fig. 4.6:

```
#include "Timer.h"
module BlinkC
{
    uses interface Timer<TMilli> as Timer0;
    uses interface Timer<TMilli> as Timer1;
    uses interface Timer<TMilli> as Timer2;
    uses interface Leds;
    uses interface Boot;
}
```

Figura 4.6: Declaración de interfaces en el archivo de implementación.

dentro del mismo archivo declaramos una sección **implementation** y definimos un primer evento llamado **Boot booted** (Boot es una interfaz implementada por el componente MainC), dentro de este método se establece un tiempo de duración para cada uno de los temporizadores, como se observa en la Fig. 4.7:

```
implementation
{
    event void Boot.booted()
    {
        call Timer0.startPeriodic( 250 );
        call Timer1.startPeriodic( 500 );
        call Timer2.startPeriodic( 1000 );
    }
}
```

Figura 4.7: Método de arranque de la aplicación Blink.

enseguida declaramos el evento a ejecutar por los temporizadores, en el que se indica el tipo de led a encender y apagar de forma intermitente, como se observa en la Fig. 4.8:

```
implementation
{
    ...
    event void Timer0.fired()
    {
        call Leds.led0Toggle();
    }
    event void Timer1.fired()
    {
        call Leds.led1Toggle();
    }
    event void Timer2.fired()
    {
        call Leds.led2Toggle();
    }
}
```

Figura 4.8: Eventos de los temporizadores.

el evento led0Toggle, por ejemplo, indica que si el led0 esta apagado, entonces debe encenderse, y si esta encendido entonces debe apagarse. Una vez definida la implementación de los componentes, procedemos a realizar el proceso de enlace de componentes (“wiring”), modificando el contenido del archivo BlinkAppC.nc, como se observa en la Fig. 4.9:

```
configuration BlinkAppC
{
}

implementation
{
    components MainC, BlinkC, LedsC;
    components new TimerMilliC() as Timer0;
    components new TimerMilliC() as Timer1;
    components new TimerMilliC() as Timer2;

    BlinkC -> MainC.Boot;
    BlinkC.Timer0 -> Timer0;
    BlinkC.Timer1 -> Timer1;
    BlinkC.Timer2 -> Timer2;
    BlinkC.Leds -> LedsC;
}
```

Figura 4.9: Enlace de componentes.

Por último, es importante crear un archivo llamado **Makefile** (sin extensión alguna) el cual contiene el nombre del componente de configuración y el nombre del directorio donde se establecen las reglas para la compilación de nuestro código. Este directorio esta registrado en nuestras variables de entorno como **MAKERULES**, por lo tanto el contenido de nuestro archivo debe ser similar al que se muestra en la Fig. 4.10:

```
COMPONENT=BlinkAppC
include $(MAKERULES)
```

Figura 4.10: Reglas de compilación.

Para compilar la aplicación debemos ubicarnos a través de la terminal dentro de la carpeta Blink y ejecutar el siguiente comando:

```
$ make plataforma_de.hardware
```

por ejemplo, si nuestra plataforma de hardware es **iris**, el comando a ejecutar es el siguiente:

```
$ make iris
```

Por otra parte, para instalar esta aplicación en el nodo sensor debemos utilizar el siguiente comando:

```
$ make plataforma_de.hardware install dispositivo,puerto
```

por ejemplo, de acuerdo con nuestro caso anterior una manera de instalar esta aplicación sería:

```
$ make iris install mib520,/dev/ttyUSB0
```

#### 4.4.2. Aplicación para la transmisión de datos ambientales

El objetivo es realizar un proceso de comunicación entre un nodo sensor y la estación base, considerando los siguientes puntos:

1. **Aplicación en el nodo sensor:** Utilizando un período de muestreo de 1 segundo, el nodo sensor debe obtener la temperatura del medio ambiente y transmitirla a través de un paquete de manera inalámbrica.
2. **Aplicación en la estación base:** La estación base debe recibir y presentar los datos obtenidos por el nodo sensor en una terminal.

##### 4.4.2.1. Aplicación para el nodo sensor

Esta aplicación ha sido nombrada como Mts400Tester. Los componentes requeridos por la aplicación son:

- **MainC:** Establece el inicio de la aplicación.
- **LedsC:** Es útil para la manipulación de leds en el nodo sensor.

- **ActiveMessageC:** Proporciona mecanismos para la comunicación entre nodos sensores, basada en mensajes.
- **AMSenderC:** Proporciona una interfaz básica para el envío de mensajes.
- **Intersema5534C:** Es un componente de nivel superior para el acceso al sensor Intersema modelo 5534, disponible en algunas plataformas de hardware como por ejemplo iris.
- **Mts400TesterP:** Es una referencia al archivo que contiene la implementación de los componentes, como se observa en la Fig. 4.11.

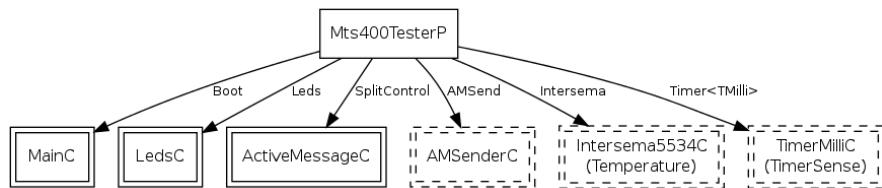


Figura 4.11: Diagrama de componentes de Mts400Tester.

El primer paso, consiste en crear una estructura para el almacenamiento de datos, con el nombre de **Buffer.h**. Básicamente este archivo debe contener un identificador único para el paquete de datos, una cantidad en milisegundos para representar el período de muestreo, una variable para el almacenamiento del identificador del nodo sensor y una variable para el almacenamiento de la temperatura. En nuestro caso, se ha definido a estas variables como **AM\_DATAMSG**, **TIMER\_PERIOD\_MILLI**, **NodeID** y **Temp\_data** respectivamente, como se observa en la Fig. 4.12.

```

#ifndef DATAMSG_H
#define DATAMSG_H

enum{
    AM_DATAMSG=1,
    TIMER_PERIOD_MILLI = 1024
};

typedef nx_struct datamsg{
    nx_uint16_t NodeID;
    nx_int16_t Temp_data;
}datamsg_t;

#endif
  
```

Figura 4.12: Estructura para el almacenamiento de datos.

Nuestro segundo paso consiste en crear el archivo de configuración **Mts400TesterC.nc**, declarando los componentes requeridos e importando nuestra estructura para el paquete de datos, como se observa en la Fig. 4.13.

```
#include "Buffer.h"

configuration Mts400TesterC { }

implementation {
    components MainC;
    components LedsC;
    components ActiveMessageC;
    components new AMSenderC(AM_DATAMSG);
    components new Intersema5534C() as Temperature;
    components new TimerMilliC() as TimerSense;
    components Mts400TesterP as App;
}
```

Figura 4.13: Componente de configuración de Mts400TesterC.nc.

El tercer paso es crear un componente para la implementación llamado **Mts400TesterP.nc**. Este componente debe definir cuales son las interfaces a implementar, como se observa en la Fig. 4.14.

```
#include "Buffer.h"

module Mts400TesterP {
    uses interface Boot;
    uses interface Leds;
    uses interface SplitControl;
    uses interface AMSend;
    uses interface Intersema as Temperature;
    uses interface Timer<TMilli> as TimerSense;
}
```

Figura 4.14: Componente de implementación de Mts400TesterP.nc.

Antes de comenzar a definir la implementación, es importante declarar las variables que utilizaremos en este proceso, como se observa en la Fig. 4.15.

```
module Mts400TesterP {
    ...
}

implementation {
    int16_t Temp_data;
    bool busy = FALSE;
    message_t message;
    datamsg_t* packet;

    ...
}
```

Figura 4.15: Declaración de variables para nuestra implementación de componentes.

Enseguida, debemos escribir la implementación de los componentes comenzando por activar al inicio de la aplicación (esto es dentro del evento Boot.booted), una operación de tipo SplitControl, como se observa en la Fig. 4.16.

```

implementation {
    ...
    event void Boot.booted(){
        call SplitControl.start();
    }
    ...
}

```

Figura 4.16: Evento inicial de la aplicación.

SplitControl es una interfaz implementada por ActiveMessage y es utilizada para conmutar entre los estados de encendido y apagado del componente que lo provee. Una vez activado SplitControl.start, se definen las acciones a ejecutar en el evento SplitControl.statDone, como se observa en la Fig. 4.17.

```

implementation {
    ...
    event void SplitControl.startDone(error_t error){
        if(error==SUCCESS){
            call TimerSense.startPeriodic(TIMER_PERIOD_MILLI);
            call Leds.led0On();
        }else{
            call SplitControl.start();
        }
    }
    event void SplitControl.stopDone(error_t error) { }
    ...
}

```

Figura 4.17: Implementación de los métodos de SplitControl.

En este método activamos el temporizador y realizamos una invocación al evento de led0On, con el objetivo de verificar en el nodo sensor la transición de eventos de nuestra aplicación, sí y sólo sí, el método de SplitControl fue activado correctamente. Una vez disparado el evento del temporizador se debe realizar una lectura de la temperatura ambiental a través del método Temperature.read, como se observa en la Fig. 4.18.

```

implementation {
    ...
    event void TimerSense.fired(){
        call Temperature.read();
    }
    ...
}

```

Figura 4.18: Método para la lectura de la temperatura ambiental.

El circuito Intersema 5534, además de proveer de un sensor de temperatura contiene un sensor de presión barométrica. Una vez invocado al método read desde el componente Inter-

sema5534C, este retornará un arreglo de dos valores: el índice 0 corresponde a la temperatura y el índice 1 corresponde a la presión barométrica. Nuestro siguiente paso es almacenar la información de la temperatura obtenida en la variable Temp\_data, como se observa en la Fig. 4.19:

```
implementation {
    ...
    event void Temperature.readDone(error_t err, int16_t* data){
        Temp_data = data[0];
        ...
    }
}
```

Figura 4.19: Almacenamiento de la temperatura en una variable temporal.

Enseguida debemos construir el paquete; proporcionando además el identificador del nodo, para esto podemos tomar el valor de la variable por defecto **TOS\_NODE\_ID**, como se observa en la Fig. 4.20.

```
implementation {
    ...
    event void Temperature.readDone(error_t err, int16_t* data){
        ...
        packet = (datamsg_t*)(call AMSend.getPayload(&message, sizeof(datamsg_t)));
        packet-> NodeID = TOS_NODE_ID;
        packet-> Temp_data = Temp_data;
        ...
    }
}
```

Figura 4.20: Creación del paquete de datos.

Después, debemos transmitir la información con la ayuda del método **send**, como se observa en la Fig. 4.21:

```
implementation {
    ...
    event void Temperature.readDone(error_t err, int16_t* data){
        ...
        call AMSend.send(AM_BROADCAST_ADDR, &message, sizeof(datamsg_t));
    }
}
```

Figura 4.21: Envío de datos por medio de un mensaje de multi-difusión (Broadcast).

para verificar que el paquete de datos ha sido enviado correctamente, podemos invocar al método led1Toggle(), como se observa en la Fig. 4.22.

```

implementation {
    ...
    event void AMSend.sendDone(message_t *msg, error_t error){
        if(error==SUCCESS){
            call Leds.led1Toggle();
        }
    }
}

```

Figura 4.22: Evento de confirmación del envío de mensajes.

Una vez definida la implementación, procedemos a realizar el proceso de enlace de componentes, modificando el contenido del archivo Mts400TesterC.nc, como se observa en la Fig. 4.23.

```

implementation {
    ...
    App.Boot -> MainC;
    App.Leds -> LedsC;
    App.SplitControl -> ActiveMessageC;
    App.AMSend -> AMSenderC;
    App.Temperature -> Temperature;
    App.TimerSense -> TimerSense;
}

```

Figura 4.23: Enlace de componentes de la aplicación Mts400Tester.

El último paso consiste en crear el archivo **Makefile**. En este archivo debemos indicar el nombre del componente de configuración, el nombre de la plataforma de hardware utilizada, como se observa en la Fig. 4.24.

```

COMPONENT=Mts400TesterC
SENSORBOARD=mts400

```

Figura 4.24: Parámetros inciales en el archivo Makefile.

Debemos incluir también la configuración estándar para trabajar con la herramienta **MIG**(Message Interface Generator, por sus siglas en inglés) [33], como se observa en la Fig. 4.25.

```

BUILD_EXTRA_DEPS = DataMsg.java
CLEAN_EXTRA = $(BUILD_EXTRA_DEPS)

DataMsg.java: Buffer.h
    mig -target=iris -java-classname=DataMsg java Buffer.h datamsg -o $@

```

Figura 4.25: Configuración estándar de MIG.

MIG es una herramienta que provee TinyOS desde su primera versión y es utilizada para generar automáticamente una clase con las propiedades y métodos necesarios para el acceso

a la información de nuestro paquete de datos a través de un lenguaje de programación como por ejemplo C, Java y actualmente también Python.

Es importante mencionar que al final de este archivo debemos incluir el nombre del directorio donde se establecen las reglas para la compilación de nuestro código (MAKERULES). Una vez creado el archivo Makefile podemos compilar e instalar la aplicación en nuestro nodo sensor con las mismas instrucciones que se presentaron en el ejemplo de Blink.

#### 4.4.2.2. Aplicación para la estación base

En esta segunda parte de la aplicación necesitamos preparar a nuestra estación base para recibir datos del exterior, es decir, de todos los nodos sensores que transmiten información dentro de su cobertura. La estructura de directorios de TinyOS contiene como parte de sus aplicaciones de prueba, una aplicación que cumple con este propósito, por lo tanto, debemos instalarla una vez identificado y conectado nuestro dispositivo. Esta aplicación está ubicada en:

```
/opt/tinyos-2.1.2/apps/BaseStation
```

Todos los datos recibidos por la estación base son reenviados a través del puerto de comunicación USB a nuestro servidor. Esto facilita su manipulación por ejemplo para propósitos de monitoreo y análisis.

El objetivo de la herramienta MIG fue construir una clase para tener acceso a los datos obtenidos por el nodo sensor por medio de un lenguaje de programación de alto nivel. En el proceso de compilación de la aplicación desarrollada para el nodo sensor, se generó una clase para trabajar con Java, nombrada como **DataMsg.java**. Sin embargo, para no interferir con el código generado automáticamente, debemos escribir una clase adicional para acceder a DataMsg.java y presentar la información en la terminal (computadora o dispositivo móvil).

Una propuesta de clase principal se presenta a continuación:

```
import static java.lang.System.out;
import net/tinyos.message.*;
import net/tinyos.util.*;
import net/tinyos.packet.*;

class Mts400Tester implements MessageListener{
    private PhoenixSource phoenix;
    private MoteIF mif;

    public Mts400Tester(final String source){
        phoenix=BuildSource.makePhoenix(source,PrintStreamMessenger.err);
        mif = new MoteIF(phoenix);
```

```

mif.registerListener(new DataMsg(),this);
}

public void messageReceived(int dest_addr,Message msg) {
    if(msg instanceof DataMsg) {
        DataMsg results = (DataMsg)msg;
        System.out.println("The measured results are ");
        System.out.println("NodeID " + results.getNodeID() +
                           "\tTemperature: "+(results.getTemp_data()/10));
    }
}

public static void main (String[] args) {
    Mts400Tester hy = new Mts400Tester("serial@/dev/ttyUSB1:iris");
}
}

```

Para realizar la evaluación de estas aplicaciones utilizamos el entorno de desarrollo Eclipse [34]. Además como parte de la configuración de nuestro proyecto, fue necesario importar desde el Buildpath el archivo **tinyos.jar**, como se observa en la Fig. 4.26.

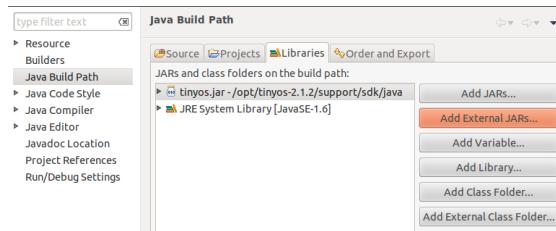


Figura 4.26: Importando **tinyos.jar** desde el Buildpath del proyecto.

este archivo se encuentra ubicado en:

```
tinyos-2.1.2/support/sdk/java/tinyos.jar
```

Una vez configurado nuestro proyecto, se realizó la evaluación de las aplicaciones, el resultado obtenido se observa en la Fig. 4.27.



Figura 4.27: Resultado de la evaluación en el entorno de Eclipse.

# **Capítulo 5**

## **Método propuesto para la detección de incendios forestales**

### **5.1. Motivación**

En la actualidad, los incendios representan una de las principales causas de degradación en los ecosistemas del planeta, principalmente en superficies forestales. Existen diferentes métodos para la detección de incendios, tales como: monitoreo en torres de vigilancia y uso de imágenes satelitales [35, 36]. Desafortunadamente la implementación de estos métodos dificulta el monitoreo en tiempo real y en muchas ocasiones cuando el fenómeno es detectado, su velocidad de propagación ha producido niveles de daño incontrolables.

Una red de sensores inalámbrica [2], es un sistema distribuido compuesto por nodos con capacidad de obtener información acerca de las condiciones ambientales del entorno y transmitirla de manera inalámbrica a un punto de acceso (por ejemplo: una estación base), para su procesamiento o análisis. Los campos de aplicación son distintos: salud, educación, seguridad, entretenimiento entre otros. Entre sus ventajas respecto a los métodos mencionados previamente, destacan el monitoreo en tiempo real, escalabilidad, administración de la energía, autonomía y bajo costo.

En la literatura, se pueden encontrar diferentes propuestas para la detección de incendios forestales utilizando una red de sensores inalámbrica. Algunos sistemas utilizan una gran cantidad de nodos, diferentes tipos de sensores [19–22, 37], arquitecturas muy sofisticadas [27], o bien evalúan su desempeño a través de aplicaciones de simulación [24, 25, 28].

En [37] investigación previa, se proponen dos métodos de detección de incendios forestales. Ambos métodos utilizan técnicas de fusión de información. El primero esta basado en umbrales utilizando los siguientes tipos de sensores: temperatura, humedad relativa e in-

tensidad luminosa, mientras que el segundo método, está basado en la teoría de la evidencia Dempster-Shafer [38] y métodos de interpolación, utilizando los siguientes tipos de sensores: temperatura y humedad relativa. Los resultados obtenidos demuestran que los métodos son capaces de identificar correctamente todos los incendios generados. Sin embargo, ambos métodos reportan falsos positivos debido a la exposición del nodo sensor ante los rayos del sol.

En esta investigación, se propone una arquitectura basada en una red de sensores inalámbrica y un algoritmo de detección de incendios de baja complejidad computacional. La diferencia clave, son las técnicas utilizadas para caracterizar el fenómeno de incendio.

## 5.2. Observaciones

Para conocer las características del ambiente en su estado natural y en una situación de incendio, se recolectaron datos durante varios días. El análisis de los datos, arrojó como resultado las siguientes observaciones:

1. En condiciones normales, la temperatura y la humedad, manifiestan un comportamiento cíclico durante las diferentes etapas del día, como se observa en la Fig. 5.1.

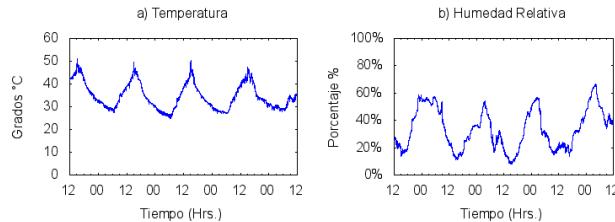


Figura 5.1: Comportamiento en condiciones normales.

2. Tanto en condiciones normales como experimentales la temperatura y la humedad mantienen una relación inversamente proporcional de sus valores, como se observa en la Fig. 5.2.

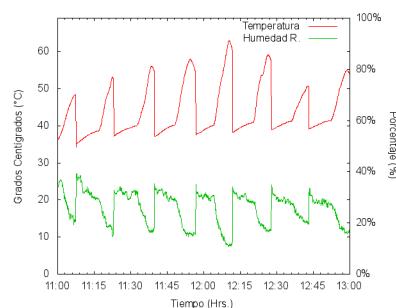


Figura 5.2: Experimentos de fuego controlado.

3. En condiciones normales la temperatura y la humedad relativa, cambian muy lentamente durante el día. Por otro lado, cuando un incendio ocurre la razón de cambio en la temperatura y la humedad es mayor, y se presenta en intervalos menores de tiempo, como se observa en la Fig. 5.3.

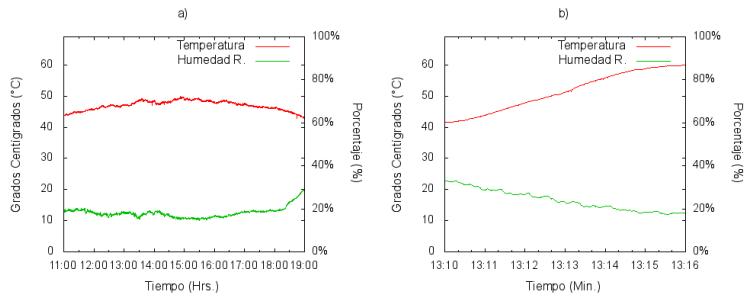


Figura 5.3: Comparación entre: a) Condiciones normales b) Experimento de fuego controlado.

4. Un evento natural que provoca un incremento en la temperatura y por lo tanto un decremento en la humedad, es la exposición del nodo sensor ante los rayos del sol. En la Fig. 5.4, se presenta la relación que existe entre valores de temperatura y humedad relativa cuando: a) el sol afecta directamente al nodo sensor b) se realiza un experimento de fuego controlado.

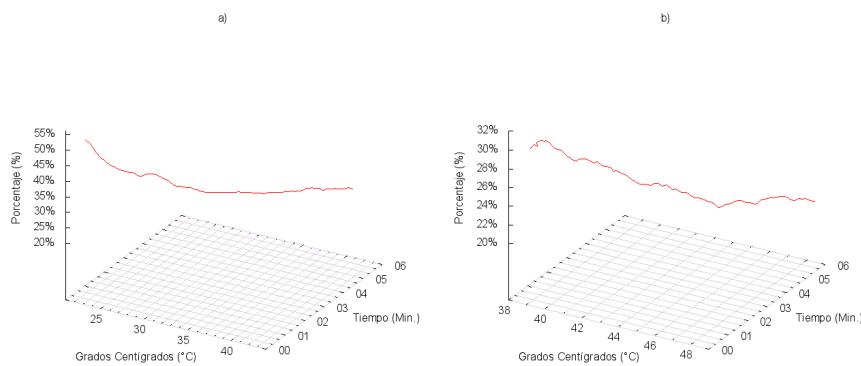


Figura 5.4: La relación entre las unidades manifiesta un comportamiento decreciente para ambos casos.

Con base en estas observaciones, se propone desarrollar un modelo caracterizco del fenómeno de incendio utilizando sólo los sensores de temperatura y humedad relativa, debido a que la relación inversamente proporcional que prevalece en estas unidades permite ampliar nuestro criterio de decisión, manteniendo dos observadores dentro de un mismo evento.

## 5.3. Técnicas utilizadas para el desarrollo del modelo

### 5.3.1. Análisis de regresión

El análisis de regresión, es un proceso estadístico utilizado para estudiar la relación entre una presumiblemente variable dependiente y una o más variables independientes. Cuando trabajamos con una variable dependiente (denominada también “*variable de respuesta*”) y una variable independiente (denominada también “*variable predictora*”), utilizamos el análisis de regresión simple; cuando trabajamos con una variable dependiente y dos o más variables independientes, utilizamos el análisis de regresión múltiple. La relación entre las variables no siempre es perfecta o nula y es muy importante elegir adecuadamente el rol para cada una. El análisis de regresión incluye entre sus objetivos:

- **Descripción de los datos.** Cuando contamos con una serie de datos de algún análisis en particular, es posible representarlos gráficamente en un diagrama de dispersión (nube de puntos), sin embargo, este método es muy poco específico y poco informativo. Una alternativa consiste en representar los datos utilizando una función matemática, como por ejemplo: una línea recta, una función polinomial, una función exponencial, etcétera. Esto proporciona mayor precisión al momento de describir el comportamiento de los datos.
- **Estimación de parámetros.** Al ajustar el comportamiento de los datos con una función matemática, es posible poder estimar valores desconocidos mediante nuestro análisis. Existen diferentes métodos para ajustar una función a un conjunto de datos, uno de los más utilizados es el método de mínimos cuadrados.

### 5.3.2. Método de mínimos cuadrados

El método de mínimos cuadrados, es una manera estandarizada de medir la variación entre los datos. Este método proporciona una función que minimiza la suma de los cuadrados de las diferencias entre la estimación de un modelo de función ideal ( $y_e$ ) y los datos reales ( $y_i$ ), como se observa en la ecuación 5.1.

$$S = \sum_{i=1}^n (y_e - y_i)^2 \quad (5.1)$$

Por ejemplo, si el modelo de función ideal para ajustarse a nuestros datos es una línea recta, estamos hablando de una ecuación de la forma  $y_e = mx_i + b$  sustituyendo sobre la

ecuación 5.1, obtenemos:

$$S = \sum_{i=1}^n (mx_i + b - y_i)^2 \quad (5.2)$$

esta nueva ecuación proporciona dos incógnitas: la pendiente denotada por la letra  $m$  y la constante que corta al eje de las ordenadas denotada por la letra  $b$ . Derivando con respecto a cada una de estas incógnitas, obtenemos las ecuaciones 5.3 y 5.4:

$$m \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i = \sum_{i=1}^n x_i y_i \quad (5.3)$$

$$m \sum_{i=1}^n x_i + bn = \sum_{i=1}^n y_i \quad (5.4)$$

Para resolver este sistema de ecuaciones, podemos utilizar el método de determinantes, obteniendo como resultado:

$$m = \frac{n \sum_{i=1}^n x_i y_i - \left( \sum_{i=1}^n x_i \right) \left( \sum_{i=1}^n y_i \right)}{n \sum_{i=1}^n x_i^2 - \left( \sum_{i=1}^n x_i \right)^2} \quad (5.5)$$

$$b = \frac{\sum_{i=1}^n y_i - m \sum_{i=1}^n x_i}{n} \quad (5.6)$$

Sí los datos se encuentran muy dispersos la bondad de ajuste será baja; en caso contrario si la dispersión es pequeña la bondad de ajuste será alta, a este valor se le llama coeficiente de determinación o  $r^2$ . El coeficiente de determinación es un valor que oscila entre 0 y 1, y se calcula de la siguiente manera:

$$r^2 = 1 - \frac{\text{Suma de cuadrados de los residuos}}{\text{Suma de cuadrados total}} \quad (5.7)$$

sin embargo, una alternativa para obtener el valor del coeficiente de determinación es obtener primero el valor del coeficiente de correlación y después elevar el resultado al cuadrado. El coeficiente de correlación es un índice que mide el grado de covariación que existe entre las variables de un modelo de regresión lineal definido por la ecuación 5.8:

$$r = \frac{n \left( \sum_{i=1}^n x_i f(x_i) \right) - \left( \sum_{i=1}^n x_i \right) \left( \sum_{i=1}^n f(x_i) \right)}{\sqrt{n \left( \sum_{i=1}^n x_i^2 \right) - \left( \sum_{i=1}^n x_i \right)^2} \cdot \sqrt{n \left( \sum_{i=1}^n f(x_i^2) \right) - \left( \sum_{i=1}^n f(x_i) \right)^2}} \quad (5.8)$$

Cuanto más cercano sea el valor de  $r^2$  a la unidad, mayor poder explicativo tendrá el modelo de regresión.

**Ejemplo.** Utilizando los datos de la Tabla 5.1, ajustar una función por el método de mínimos cuadrados y obtener el valor del coeficiente de determinación.

$i$	$x_i$	$y_i$
1	0.56	1.26
2	1.00	1.00
3	1.52	1.60
4	2.16	1.08
5	2.12	2.58
6	2.98	2.26
7	4.14	3.04
8	4.00	4.00
9	5.00	4.00
10	5.22	5.18

Tabla 5.1: Datos para la evaluación del método de mínimos cuadrados.

**Solución.** El primer paso consiste en graficar los datos, como se observa en la Fig. 5.5:

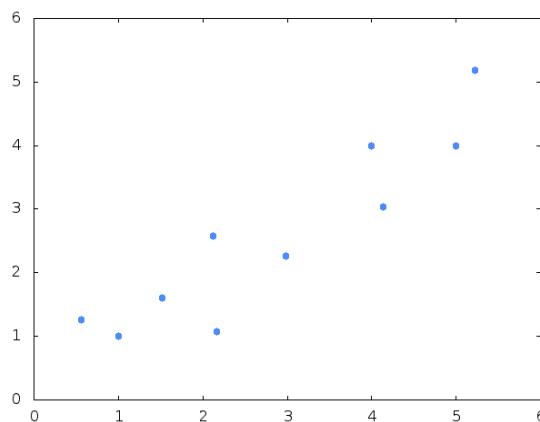


Figura 5.5: Gráfica de los datos de la Tabla 5.1.

para encontrar un modelo de función ideal, necesitamos observar cuidadosamente nuestra nube de puntos. En nuestro caso, si trazamos una línea recta como se observa en la Fig. 5.6, podemos notar que la función ideal pertenece a la familia de funciones de tipo  $y = mx + b$ .

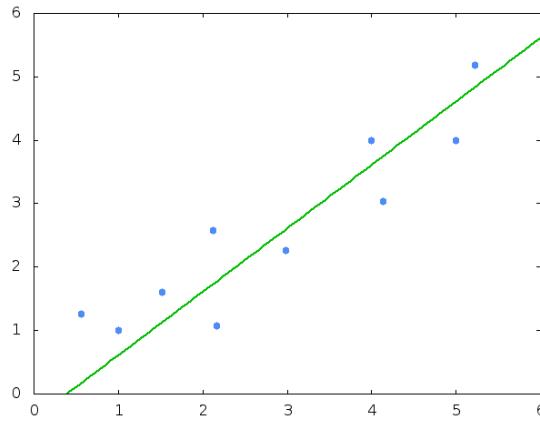


Figura 5.6: Descripción de los datos de la Tabla 5.1.

Para simplificar el cálculo elaboramos la Tabla 5.2, como se observa a continuación:

$i$	$x_i$	$y_i$	$x_i y_i$	$x_i^2$	$y_i^2$
1	0.56	1.26	0.7056	0.3136	1.5876
2	1.00	1.00	1.00	1.00	1.00
3	1.52	1.6	2.432	2.3104	2.56
4	2.16	1.08	2.3328	4.6656	1.1664
5	2.12	2.58	5.4696	4.4944	6.6564
6	2.98	2.26	6.7348	8.8804	5.1076
7	4.14	3.04	12.5856	17.1396	9.2416
8	4.00	4.00	16.00	16.00	16.00
9	5.00	4.00	20.00	25.00	16.00
10	5.22	5.18	27.0396	27.2484	26.8324
$\sum_{i=1}^{10}$	28.7	26.00	94.3	107.0524	86.152

Tabla 5.2: Sumas de apoyo.

al sustituir estos valores sobre las ecuaciones 5.5 y 5.6 obtenemos:

$$m = \frac{(10)(94.3) - (28.7)(26)}{(10)(107.0524) - (28.7)^2} = \frac{943 - 746.2}{1070.524 - 823.69} = 0.7972$$

$$b = \frac{(26) - (0.7972)(28.7)}{10} = \frac{26 - 22.87}{10} = 0.312$$

por lo tanto, nuestra función resultante para ajustarse a los datos de la Tabla 5.1 es:

$$y_e = 0.7972x + 0.312$$

como se observa en la Fig. 5.7:

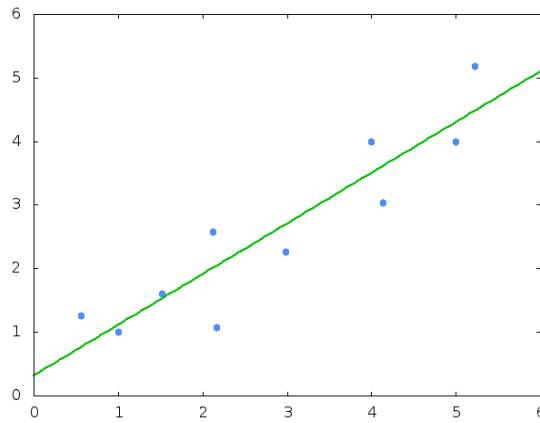


Figura 5.7: Función obtenida por el método de mínimos cuadrados.

Por último, para obtener el coeficiente de determinación utilizamos la ecuación 5.8 y el resultado lo elevamos al cuadrado como se observa a continuación:

$$r^2 = \left( \frac{(10)(94.3) - (28.7)(26)}{\sqrt{(10)(107.0524)} - (28.7)^2 \cdot \sqrt{(10)(86.15) - (26)^2}} \right)^2 = \left( \frac{196.8}{213.98} \right)^2 = 0.8458$$

el resultado obtenido cumple con la siguiente condición  $0 < 0.8458 < 1$ , este valor demuestra un buen comportamiento de los datos con respecto a la función obtenida.

### 5.3.3. Razón de cambio

Suponga que  $y$  es una cantidad que depende de otra cantidad  $x$ . Por lo tanto,  $y$  contiene un valor en función de  $x$  y esto puede escribirse como  $y = f(x)$  [39]. Si  $x$  cambia de  $x_1$  a  $x_2$ , entonces el cambio en  $x$  (incremento) es:

$$\Delta x = x_2 - x_1 \tag{5.9}$$

y el cambio correspondiente en  $y$  es:

$$\Delta y = f(x_2) - f(x_1) \tag{5.10}$$

El cociente de las diferencias entre los puntos es llamado razón promedio de cambio o pendiente. La pendiente de una recta no vertical que pasa por los puntos  $A = (x_1, f(x_1))$  y  $B = (x_2, f(x_2))$  es:

$$m = \frac{\Delta y}{\Delta x} = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \quad (5.11)$$

como se observa en la Fig. 5.8.

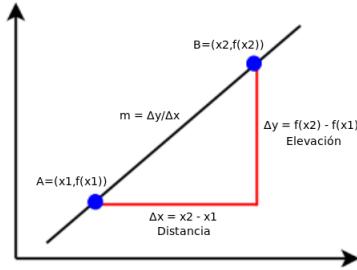


Figura 5.8: Razón de cambio entre dos puntos.

Si  $m = 0$  significa que el comportamiento de la recta es constante, como se observa en la Fig. 5.9.

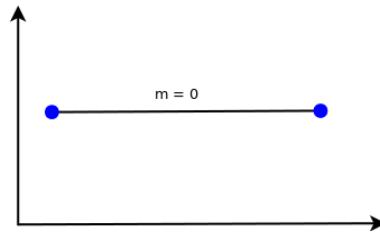


Figura 5.9: Pendiente de la recta constante.

La pendiente de una recta vertical no está definida.

#### 5.3.4. Error cuadrático promedio

Una técnica que proporciona información acerca de tan similares son dos funciones, es el error cuadrático promedio o MSE (Mean Squared Error, por sus siglas en inglés), definido por la ecuación 5.12:

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - X_i)^2 \quad (5.12)$$

donde  $\hat{Y}$  representa una estimación y  $X$  representa un serie de datos (por ejemplo, los datos obtenidos por un sensor).

## 5.4. Arquitectura del modelo propuesto

En términos generales, la arquitectura propuesta funciona de la siguiente manera: los nodos sensores analizan la temperatura del entorno constantemente, al detectar un incremento transmiten la información a la estación base. La estación base, se encarga de almacenar los datos en el servidor, y el servidor realiza una comparación entre los datos recibidos con respecto a un modelo base previamente definido. El modelo base, representa las condiciones de la temperatura y la humedad relativa en función del tiempo, en una situación de incendio. De acuerdo al grado de similitud de los datos, el sistema determina si existe un incendio en la zona de monitoreo o es un evento de condiciones normales. En la Fig. 5.10, se presenta un esquema de la arquitectura propuesta, la cual esta dividida en tres partes.

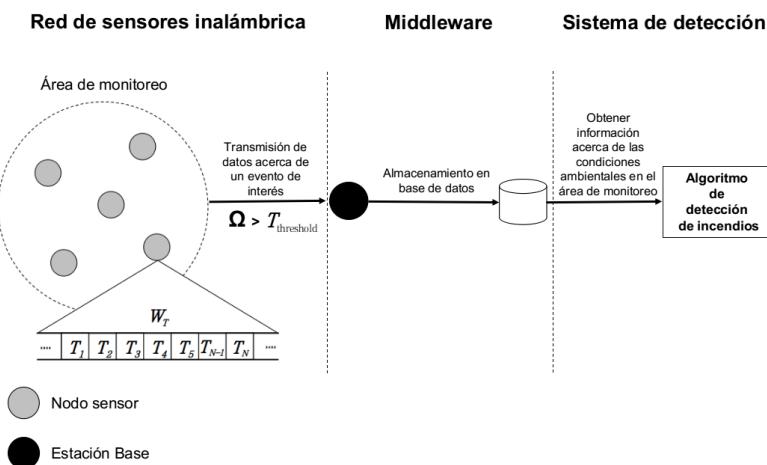


Figura 5.10: Arquitectura propuesta para el sistema de detección de incendios.

### 5.4.1. Red de sensores inalámbrica

En la red, cada nodo sensor tienen la función de realizar lecturas acerca de la temperatura  $T$ , utilizando un período de muestreo constante  $P$ . Internamente las lecturas son almacenadas en una estructura de datos de tipo FIFO (First-In, First-Out) llamada  $W_T$  de tamaño  $N$ . Para verificar si la lectura siguiente, es decir,  $T_{N+1}$  ha manifestado un cambio (incremento o decremento), se evalua la siguiente razón:

$$\Omega = \frac{T_{N+1}}{\mu(W_T)} \quad (5.13)$$

donde  $\mu(W_T)$  corresponde al valor promedio de la temperatura en un momento determinado.

El resultado de  $\Omega$ , será comparado con un umbral de temperatura denominado como  $T_{threshold}$ . En el supuesto caso que  $\Omega$  sea mayor a  $T_{threshold}$ , el nodo sensor debe realizar una lectura de la humedad relativa  $H$  e inmediatamente después, debe transmitir un paquete de datos con ambas muestras, así como, el identificador del nodo sensor y un valor que define su orden en el tiempo denominado como  $t$ . Para el primer paquete el valor de  $t = 0$ , su incremento depende del período de muestreo utilizado. En caso contrario el nodo sensor debe continuar almacenando y evaluando muestras de temperatura, como se describe en el Algoritmo 5.1:

---

**Algoritmo 5.1** Pre-procesamiento en el nodo sensor
 

---

Sea  $P$  = período de muestreo,  $N$  = número de muestras requeridas,  $W_T$  = histórico de datos de temperatura,  $T_{threshold}$  = umbral de temperatura,  $nodeID$  = identificador del nodo sensor,  $T$  = temperatura,  $H$  = humedad relativa,  $t = 0$ ,  $\Omega = 0$ ,  $i$  = índice de datos igual a 1;

```

1. while(1) {
2.   T = Leer T; //leer la temperatura ambiental
3.   if (i ≤ N) {
4.     WT[i] = T;
5.     i = i + 1;
6.   } else {
7.     Ω = (T / promedio de WT);
8.     while(Ω > Tthreshold) {
9.       H = Leer H; //leer la humedad ambiental
10.      Enviar nodeID, T, H y t;
11.      Remover el dato más antiguo del histórico y almacenar T en WT;
12.      t = t + P;
13.      T = Leer T; //leer la temperatura ambiental
14.      Ω = (T / promedio de WT);
15.    } //fin del ciclo while
16.    t = 0;
17.    i = 0;
18.  } //fin de la condición if
19. Esperar hasta el siguiente período;
20.} // fin del ciclo while

```

---

### 5.4.2. Middleware

Cuando el nodo sensor transmite un paquete a la estación base, es importante almacenar la información en una base de datos para su posterior análisis. La función del Middleware, es establecer mecanismos de comunicación entre la estación base y el servidor de base de datos para el almacenamiento y consulta de la información. El Middleware asigna un tiempo de arribo, compuesto por la fecha y hora del servidor a cada paquete recibido por la estación base, de acuerdo con el identificador del nodo y equivalente a su respectivo valor de  $t$ .

### 5.4.3. Algoritmo de detección de incendios

El algoritmo de detección de incendios propuesto, cuenta con la capacidad de identificar la evidencia proporcionada por los nodos sensores y determinar en base a una comparación con un modelo base, la presencia de incendio en el área de monitoreo, para posteriormente enviar una alerta al usuario.

El proceso de comparación entre los datos, se realiza considerando el valor de la razón de cambio de la temperatura y humedad relativa en función del tiempo con respecto a un modelo base compuesto por una función que representa la temperatura denominada  $T(t)$  y una función que representa la humedad relativa denominada  $H(t)$ , en condiciones de incendio.

En esta propuesta utilizamos la razón de cambio de las unidades, debido a que las condiciones climáticas no son un sistema determinístico (mismas entradas proporcionan invariablemente las mismas salidas) que pueda ser representado por magnitudes fijas.

Como toda la información recibida por el servidor es almacenada en base de datos, puede ser representada como un conjunto de datos compuesto por los siguientes elementos: muestra de temperatura  $T$ , muestra de humedad  $H$  y tiempo de arriba  $t$ . Por tanto, cada paquete será representado por los conjuntos  $U$  y  $V$  de la siguiente manera:

$$U = (T_{n-1}, H_{n-1}, t_{n-1}) \quad (5.14)$$

$$V = (T_n, H_n, t_n) \quad (5.15)$$

donde  $n$  es un índice utilizado para representar la información más reciente; mientras que  $n - 1$  es un índice utilizado para representar la información previa. Para denotar esta información utilizamos la siguiente notación:  $V_T$  equivale a la temperatura reciente y  $U_T$  equivale a la temperatura previa.

Por otra parte, debido a que el nodo sensor utiliza un período de muestreo prácticamente muy pequeño (segundos) es probable que tienda a cometer impresiones en la medición realizada. Por ejemplo: puede transmitir la misma información acerca de la temperatura más de una ocasión o enviar una medición errónea (por ejemplo  $V_T = 5$ , mientras que  $U_T = 38$ ), en este último caso la muestra es considerada como espuria (falsa) y es descartada.

Por otro lado, para prevenir que una muestra repetida afecte a nuestro algoritmo, se realiza una comparación de las unidades de temperatura de los respectivos conjuntos y en caso de ser iguales, se realiza una copia de los datos del conjunto  $U$  en un conjunto temporal llamado  $S$ . Esto permite considerar el tiempo transcurrido entre un conjunto y otro, una vez que la temperatura ha manifestado un cambio, como se describe en el Algoritmo 5.2:

---

**Algoritmo 5.2** Identificar un cambio (incremento o decremento) en la temperatura.

Sea  $U = (T_{n-1}, H_{n-1}, t_{n-1})$ ,  $V = (T_n, H_n, t_n)$ ,  $S = (T_{temporal}, H_{temporal}, t_{temporal})$ ;

1. Si  $V_T$  es diferente de  $U_T$  Entonces
  2. Si existe un conjunto temporal  $S$  entonces
  3.     obtenerRazondeCambio( $S, V$ )
  4.     Eliminar los datos de  $S$
  5.     En caso contrario
  6.     obtenerRazondeCambio( $U, V$ )
  7. Fin de la condición
  8. En caso contrario
  9. Si  $S$  no tiene datos, copiar los datos de  $U$  en  $S$
  10. Fin de la condición
  11. Copiar el contenido de  $V$  a  $U$
  12. Obtener un nuevo conjunto de muestras y almacenarlo en  $V$
- 

Según sea el caso (si existe o no la necesidad de almacenar los datos en un conjunto temporal), el siguiente paso consiste en obtener la razón de cambio del conjunto de datos y compararlo con la razón de cambio del modelo base, como se describe en el Algoritmo 5.3:

---

**Algoritmo 5.3** Obtener la razón de cambio de los conjuntos de datos.

Sea  $A$  =conjunto de muestras previo,  $B$  =conjunto de muestras reciente,  $E = 0$ ,  $neval = 0$ ,  $nmax$  =evaluaciones requeridas,  $\Theta$  =tiempo de análisis límite;

1. obtenerRazondeCambio( $A, B$ ) {
  2. Si  $E < \Theta$  Entonces
  3.      $E = E + (B_t - A_t)$  //Acumular el valor del tiempo transcurrido entre  $A$  y  $B$
  4. Fin de la condición
  5.  $R_T$  =Obtener la razón de cambio de la función base  $T(t)$ , cuando  $t = E$
  6.  $R_H$  =Obtener la razón de cambio de la función base  $H(t)$ , cuando  $t = E$
  7.  $m_T$  =Obtener la razón de cambio de los datos de temperatura  $\frac{B_T - A_T}{B_t - B_t}$
  8.  $m_H$  =Obtener la razón de cambio de los datos de humedad  $\frac{B_H - A_H}{B_t - A_t}$
  9. Si  $neval$  es igual a  $nmax$  Entonces
  10. Asignación básica de masa y toma de decisiones
  11.  $neval = 0$  //Reiniciar el número de evaluaciones realizadas
  12. En caso contrario
  13.  $SE_T^2 +=$  Obtener la diferencia cuadrática  $(R_T - m_T)^2$
  14.  $SE_H^2 +=$  Obtener la diferencia cuadrática  $(R_H - m_H)^2$
  15.  $neval = neval + 1$  //Incrementar el número de evaluaciones realizadas
  16. Fin de la condición
  - 17.}//Fin de la función
-

De acuerdo con la condición de la línea 9, es necesario cumplir con un número de evaluaciones ( $nmax$ ), para realizar un proceso de asignación básica de masa y en base a los resultados tomar una decisión. En el Algoritmo 5.4, se describe este procedimiento:

---

**Algoritmo 5.4 Asignación de masa y toma de decisiones.**

---

Sea  $SE_T^2$  = suma de las diferencias cuadráticas de la temperatura,  $SE_H^2$  = suma de las diferencias cuadráticas de la humedad,  $alpha$  = constante para considerar a un valor de masa 'incendio' ;

1. Obtener el promedio  $MSE_T = (SE_T^2/nmax)$
  2. Obtener el promedio  $MSE_H = (SE_H^2/nmax)$
  3.  $mass_T$  = Asignar un valor de masa a  $MSE_T$
  4.  $mass_H$  = Asignar un valor de masa a  $MSE_H$
  5. Obtener la masa total  $mass_{Total} = (mass_T + mass_H)/2$
  6. Si ( $mass_{Total} \geq alpha$ ) Entonces
  7. El sistema emite una alerta de incendio
  8. En caso contrario
  9. El sistema no emite alerta
  10. Fin de la condición
- 

Después de calcular el promedio de las diferencias cuadráticas (línea 1 y 2), se asigna un valor de masa (línea 3 y 4) para cada sensor. En la línea 5, se realiza un promedio de las masas obtenidas por los sensores, y en la línea 6 se compara este valor con respecto a una constante  $alpha$  para determinar si el sistema debe o no, transmitir una alerta de incendio al usuario.

El criterio utilizado para asignar un valor de masa, está relacionado con el objetivo del error cuadrático promedio, es decir, mientras más cercano sea el resultado a cero, mayor es la similitud en la razón de cambio de los datos obtenidos por el nodo sensor con respecto al modelo base, por lo tanto mayor es el número de masa asignada por el algoritmo, como se observa en la Tabla 5.3:

Masa	Resultado de MSE
0.99	0.00 <= x < 0.01
0.90	0.01 <= x < 0.02
0.80	0.02 <= x < 0.03
0.70	0.03 <= x < 0.04
0.60	0.04 <= x < 0.05
0.50	0.05 <= x < 0.06
0.40	0.06 <= x < 0.07
0.30	0.07 <= x < 0.08
0.20	0.08 <= x < 0.09
0.10	0.09 <= x < 0.10
0.01	0.10 <= x < 1.00

Tabla 5.3: Asignación básica de masa del Algoritmo 5.4.

# Capítulo 6

## Evaluación

Durante los meses de Septiembre y Octubre de 2013 se realizó la fase de experimentación. Cada experimento fue realizado entre las 11 am y 1 pm con una duración aproximada de 6 minutos e intervalos entre experimentos de 10 minutos, como se observa en la Tabla 6.1:

Experimento	Horario
1	11:00 - 11:06 am
2	11:16 - 11:22 am
3	11:32 - 11:38 am
4	11:48 - 11:54 am
5	12:04 - 12:10 pm
6	12:20 - 12:26 pm
7	12:36 - 12:42 pm
8	12:52 - 12:58 pm

Tabla 6.1: Horario de los experimentos.

En esta fase, se evaluaron dos tipos de escenarios: a) Recolección de datos en sombra b) Recolección de datos ante los rayos del sol, como se observa en la Fig. 6.1.

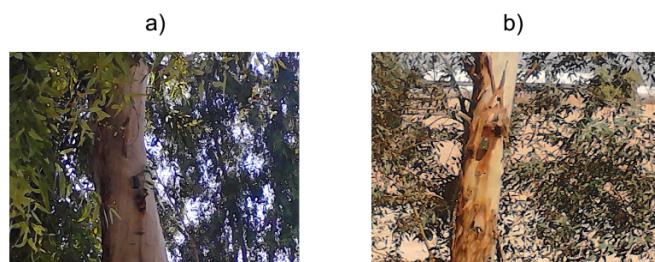


Figura 6.1: Escenarios de recolección de datos.

## 6.1. Plataforma de hardware y software utilizada

En esta investigación, fue utilizado el kit de desarrollo MEMSIC [40] modelo WSN-START2110CA, como se observa en la Fig. 6.2.



Figura 6.2: Kit de desarrollo para WSN.

Este kit incluye una estación base con un procesador ATmega1281, en una base programadora para la adquisición de datos modelo mib520 y dos nodos sensores plataforma de hardware IRIS con placa sensora modelo MTS420/400CC. Esta placa, proporciona cinco tipos de sensores ambientales: humedad relativa, temperatura, presión barométrica, intensidad lumínosa y acelerómetro, con la opción de integrar un módulo GPS (sistema de posicionamiento global). Por otra parte, nuestro servidor fue una computadora portátil marca Dell modelo XPS M1330, procesador Intel Core 2 Duo, modelo T7500 de 2.20GHz y 4 GB de memoria RAM de tipo DDR2.

En cuestiones de software, el sistema operativo base fue la distribución basada en Linux, Ubuntu 12.04 de 64 bits. Sobre esta plataforma, fue implementado el sistema operativo TinyOS [29] en su versión 2.1.2. La aplicación Middleware fue desarrollada en Java [15]. Además toda la información recolectada fue almacenada en base de datos para su análisis. La base de datos, fue desarrollada en MySQL [41], su estructura de tablas se observa en la Fig. 6.3.

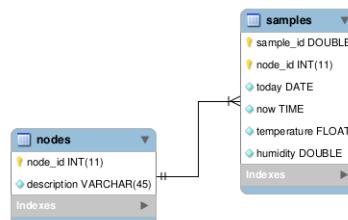


Figura 6.3: Diagrama de la base de datos.

## 6.2. Metodología

La metodología utilizada consistió en colocar el nodo sensor sobre el tronco de un árbol a una altura aproximadamente de tres metros. La recolección de datos fue dividida en dos etapas: Función base y Experimentos de fuego controlado, cada una de ellas se describe a continuación:

1. **Función base:** En esta etapa el nodo sensor fue expuesto a condiciones de fuego artificial utilizando una antorcha para jardín, como se observa en la Fig. 6.4, mientras realizaba lecturas de temperatura y humedad relativa aproximadamente durante 6 minutos, utilizando un período de muestreo de 1 segundo.



Figura 6.4: Exposición del nodo sensor ante fuego artificial.

2. **Experimentos de fuego controlado:** Esta etapa es similar a la etapa de función base, sin embargo, la diferencia entre ambas es la evaluación de diferentes períodos de muestreo: 3, 4, 6 y 8 segundos. Un factor importante en el tiempo de vida de una red de sensores inalámbrica es el ahorro de energía. El objetivo de la evaluación, fue investigar cual es el período de muestreo ideal para la detección de incendio y al mismo tiempo ideal para el ahorro de energía en el nodo sensor.

Todos los datos de acuerdo al tipo de escenario, se obtuvieron en condiciones climáticas similares.

### 6.3. Desarrollo del modelo base

Utilizando como fuente de datos los experimentos de la etapa de función base, como se observa en la Fig. 6.5, se propone un modelo basado en el análisis de regresión para caracterizar el comportamiento de las unidades (temperatura y humedad) ante el fenómeno de incendio.

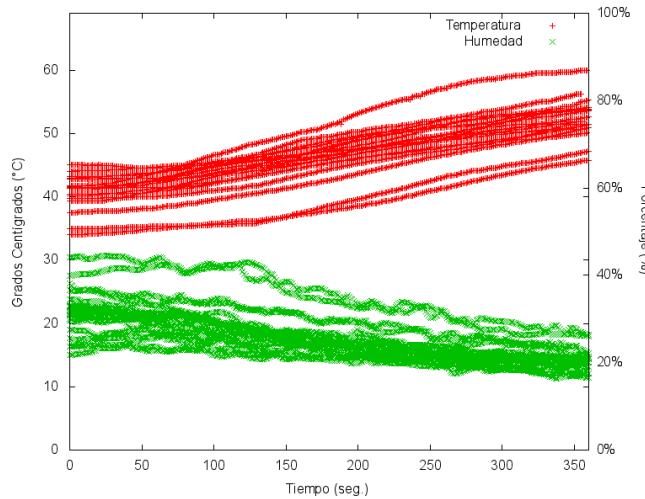


Figura 6.5: Datos de la etapa de función base.

De acuerdo con nuestras observaciones, cada vez que se presenta un incremento en la temperatura se manifiesta un decremento en la humedad tanto en presencia de incendio, así como en un evento natural (rayos del sol). Para modelar esta relación inversamente proporcional, se implementó el ajuste de curva por análisis de regresión simple con la ayuda del software de análisis estadístico SPSS de IBM [42] para los siguientes casos:

1. Temperatura en función del tiempo.
2. Humedad relativa en función del tiempo.

SPSS tiene la cualidad de presentar diferentes tipos de fuciones como resultado del análisis, por ejemplo: cuadrática, cúbica, exponencial, logarítmica, entre otras. Las funciones que mejor se ajustaron a nuestros datos fueron las siguientes:

$$T(t) = -0.000000317t^3 + 0.000185t^2 + 0.006442t + 40.738 \quad (6.1)$$

$$H(t) = 0.0000003578t^3 - 0.00017t^2 - 0.0151t + 31.824 \quad (6.2)$$

Estas funciones tienen un coeficiente de determinación de 0.598 para  $T(t)$  y 0.422 para  $H(t)$ , lo que significa, que la bondad de ajuste es baja.

Una alternativa para mejorar esto, fue descartar datos cuyo comportamiento se aleja de una tendencia, como se observa en la Fig. 6.6.

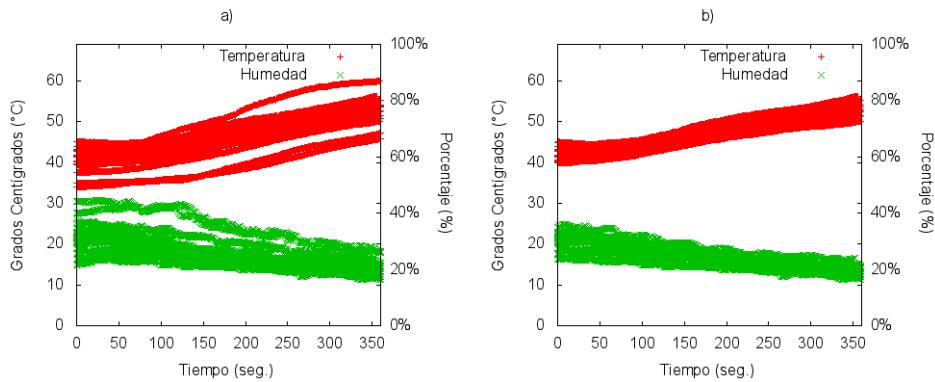


Figura 6.6: a) 24 funciones base, b) 17 funciones base.

Al realizar nuevamente el proceso de ajuste de curva, se obtuvieron como resultado las siguientes funciones:

$$T(t) = -0.0000003124t^3 + 0.000167t^2 + 0.010625t + 41.618 \quad (6.3)$$

$$H(t) = 0.0000001783t^3 - 0.00006119t^2 - 0.030952t + 30.492 \quad (6.4)$$

cuyos coeficientes de determinación son 0.874 para  $T(t)$  y 0.675 para  $H(t)$ , mejorando así la bondad de ajuste. Ambas funciones se observan en la Fig. 6.7.

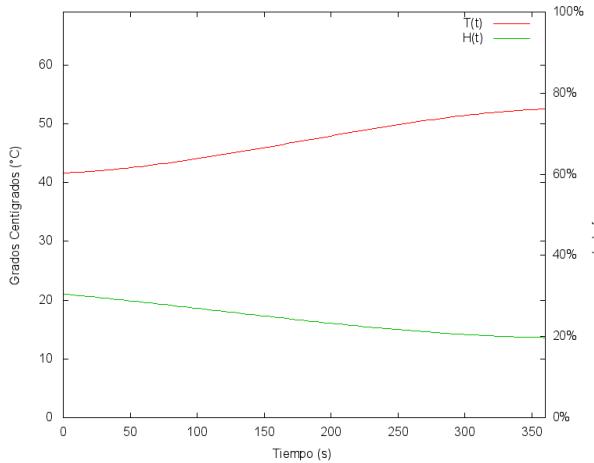


Figura 6.7: Resultado del análisis de regresión.

Por lo que a partir de este resultado, el modelo base es representado por las funciones 6.3 y 6.4.

Una manera de validar este modelo, fue a través de su comparación con diferentes escenarios, como se observa en la Fig. 6.8:

- a) Modelo base con respecto a un evento de condiciones normales en sombra.
- b) Modelo base con respecto a un experimento de incendio en sombra.
- c) Modelo base con respecto a un evento de condiciones normales en sol.
- d) Modelo base con respecto a un experimento de incendio en sol.

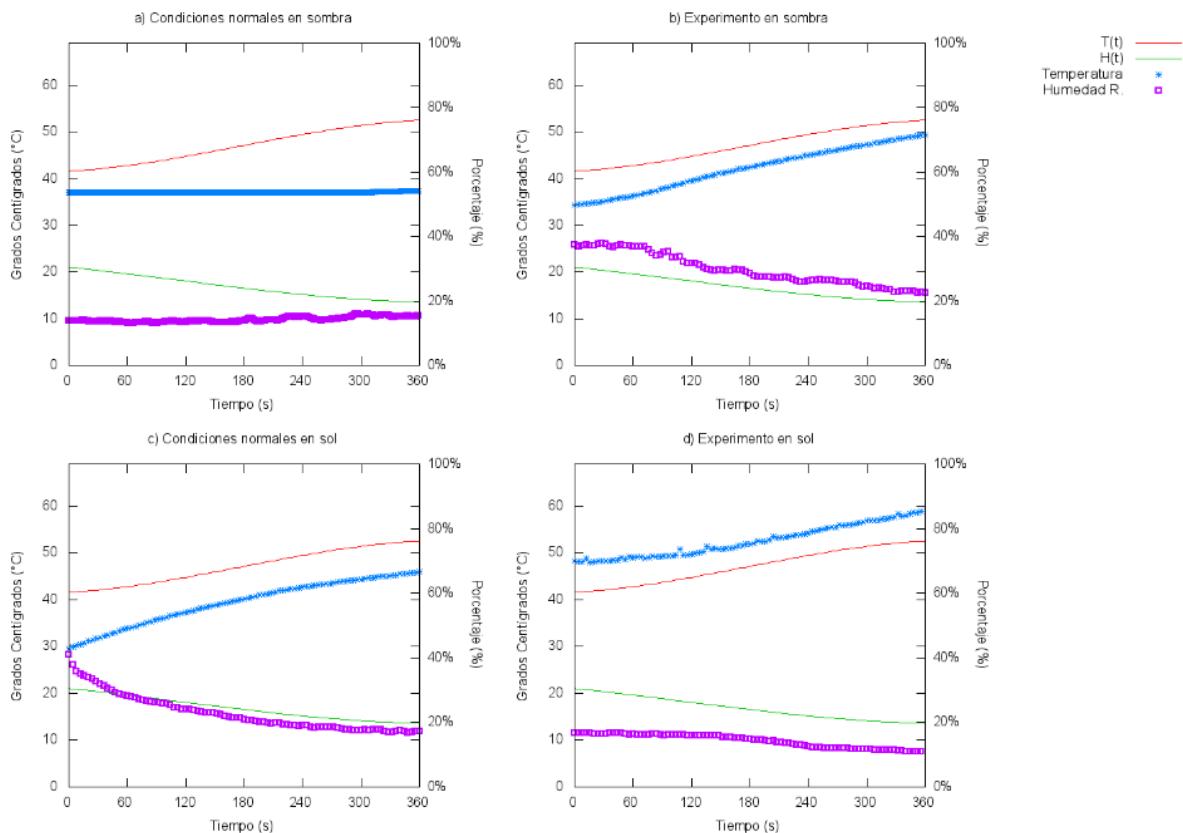


Figura 6.8: Evaluación del modelo base.

Se puede notar, que existe una similitud en la velocidad de crecimiento (razón de cambio) de las unidades de humedad y temperatura para los casos **b**, **c** y **d**, por lo que se concluye que el modelo base se ajusta de manera ideal al comportamiento de los datos en una situación anómala.

Tomando en cuenta que el modelo base es una estimación a las condiciones de un evento de incendio y que los datos reales provienen del nodo sensor, se propone utilizar la técnica del error cuadrático promedio (ecuación 5.12), para realizar una comparación entre el modelo base y la información proporcionada por el nodo sensor para definir si la similitud entre sus razones de cambio representa una amenaza de incendio o un evento de condiciones normales.

Una manera de obtener la razón de cambio del modelo base es utilizando la técnica de la derivada. Conceptualmente la derivada de una función equivale la razón de cambio de dicha función. Derivando las funciones 6.3 y 6.4 se obtienen los siguientes resultados:

$$T'(t) = -0.000000937t^2 + 0.000334t + 0.010625 \quad (6.5)$$

$$H'(t) = 0.000000535t^2 - 0.00012238t - 0.030952 \quad (6.6)$$

Para obtener la razón de cambio de los datos obtenidos por el nodo sensor utilizamos la fórmula de la pendiente 5.11, utilizando la variación de las unidades con respecto al tiempo transcurrido, como se observa en las fórmulas 6.7 y 6.8:

$$m_{temp} = \frac{T_i - T_{i-1}}{t_i - t_{i-1}} \quad (6.7)$$

$$m_{humd} = \frac{H_i - H_{i-1}}{t_i - t_{i-1}} \quad (6.8)$$

sustituyendo adecuadamente en 5.12, se obtienen las siguientes fórmulas:

$$MSE_{temp} = \frac{1}{n} \sum_{i=1}^n (T'(t_i) - m_{temp_i})^2 \quad (6.9)$$

$$MSE_{humd} = \frac{1}{n} \sum_{i=1}^n (H'(t_i) - m_{humd_i})^2 \quad (6.10)$$

El resultado de las fórmulas 6.9 y 6.10, proporciona un valor acerca de la variabilidad que existe entre los datos. Si existe muy poca variabilidad el resultado será muy próximo a cero, esto significa que los datos son muy fiables y por lo tanto existe un alto grado de similitud entre ambas funciones.

## 6.4. Evaluación del algoritmo propuesto

En una sección posterior, se definen los parámetros requeridos por la arquitectura propuesta. Sin embargo, para el desarrollo de este ejemplo definiremos el valor de  $nmax = 5$  y  $alpha = 0.7$ .

**Ejemplo.** Suponga que el servidor recibe los datos que se observan en la Tabla 6.2:

Índice ( $i$ )	Temperatura ( $T$ )	Humedad ( $H$ )	Tiempo de arribo ( $t$ )
0	37.2	37.42	12:04:34
1	37.4	37.81	12:04:40
2	37.6	37.9	12:04:46
3	37.8	37.3	12:04:52
4	38.0	36.98	12:04:58
5	38.2	37.03	12:05:04

Tabla 6.2: Experimento del 9 de septiembre de 2013.

De acuerdo con el Algoritmo 5.2, se necesitan por lo menos dos conjuntos de muestras para comenzar a verificar si existe un cambio en la temperatura. En este caso los valores de  $U$  y  $V$  son:

$$U = \{37.2, 37.42, 12:04:34\}$$

$$V = \{37.4, 37.81, 12:04:40\}$$

equivalentes al registro 0 y 1 de la Tabla 6.2. El siguiente paso, consiste en verificar si  $V_T \neq U_T$ . De acuerdo al ejemplo, el resultado se cumple debido a que  $37.4 \neq 37.2$ . Enseguida se verifica si existen datos almacenados en un conjunto temporal  $S$ , este caso no se presentó para ninguno de los datos obtenidos, debido a que la temperatura mantiene un incremento constante durante todo el experimento, por lo que omitiremos este paso para el resto del ejemplo.

A continuación se debe obtener la razón de cambio de los conjuntos de datos de acuerdo con el Algoritmo 5.3. El primer paso, es obtener la separación en el tiempo de los conjuntos, de acuerdo a la siguiente fórmula  $E = E + (V_t - U_t)$ . El algoritmo establece que mientras el sistema reciba muestras por parte del nodo sensor, el valor de  $E$  acumula las diferencias de tiempo entre los conjuntos de datos mientras que  $E$  sea menor a  $\Theta$  (tiempo de análisis límite). En esta primer iteración, el resultado es  $E = 6$ , lo que significa que probablemente el nodo sensor utiliza un período de muestreo de 6 segundos para cada lectura.

Con el valor previamente obtenido de  $E$ , se calcula la derivada de las funciones  $T(t)$  y  $H(t)$ , cuando  $t = E$ . El resultado obtenido es:

$$T'(6) = 0.0125$$

$$H'(6) = -0.0316$$

Para obtener la razón de cambio de los datos recibidos por el servidor se utilizan las fórmulas 6.7 y 6.8. Sustituyendo adecuadamente se obtienen los siguientes resultados:

$$m_{temp} = \frac{37.4 - 37.2}{12:04:40 - 12:04:34} = \frac{0.2}{6} = 0.033$$

$$m_{humd} = \frac{37.81 - 37.42}{12:04:40 - 12:04:34} = \frac{0.39}{6} = 0.065$$

El siguiente paso es comparar el valor de  $neval$  con  $nmax$ . Sin embargo, debido a que al inicio  $neval$  es menor a  $nmax$ , el siguiente paso es obtener las diferencias de ambas razones de cambio y elevar el resultado al cuadrado:

$$SE_T^2 = (0.0125 - 0.0333)^2 = 0.00043$$

$$SE_H^2 = (-0.0316 - 0.0650)^2 = 0.00933$$

después, incrementamos el valor de la variable  $neval$ , por lo tanto  $neval = 1$ . Para la segunda iteración, el valor de los conjuntos  $U$  y  $V$  es el siguiente:

$$U = \{37.4, 37.81, 12:04:40\}$$

$$V = \{37.6, 37.9, 12:04:46\}$$

Enseguida debemos calcular el valor de  $E$ . En esta ocasión  $E = 6 + (V_t - U_t) = 12$ . Si repetimos el procedimiento del Algoritmo 5.3, hasta que  $neval$  sea igual a  $nmax$ , obtenemos los resultados que se muestran en la Tabla 6.3:

$i$	$T$	$H$	$t$	$neval$	$E$	$T'(t)$	$H'(t)$	$m_{temp}$	$m_{humd}$	$SE_T^2$	$SE_H^2$
0	37.2	37.42	12:04:34	0	0	0	0	0	0	0	0
1	37.4	37.81	12:04:40	1	6	0.0125	-0.0316	0.0333	0.0650	0.00043	0.00933
2	37.6	37.9	12:04:46	2	12	0.0144	-0.0323	0.0333	0.0150	0.00035	0.00223
3	37.8	37.3	12:04:52	3	18	0.0163	-0.0329	0.0333	-0.1000	0.00028	0.00450
4	38.0	36.98	12:04:58	4	24	0.0181	-0.0335	0.0333	-0.0533	0.00023	0.00039
5	38.2	37.03	12:05:04	5	30	0.0198	-0.0341	0.0333	0.0083	0.00018	0.00179

Tabla 6.3: Resumen de las evaluaciones.

De acuerdo con el Algoritmo 5.4, cuando se cumple con el número de evaluaciones  $nmax$ , se obtiene un promedio de la suma de las diferencias cuadráticas para cada tipo de dato. El resultado de este promedio es:

$$MSE_{temp} = \frac{\sum_{i=1}^{nmax} (SE_T^2)}{nmax} = \frac{0.00149214}{5} = 0.000298428$$

$$MSE_{hum} = \frac{\sum_{i=1}^{nmax} (SE_H^2)}{nmax} = \frac{0.01826106}{5} = 0.003652212$$

En base al resultado de  $MSE_{temp}$  y  $MSE_{hum}$ , enseguida se realiza una asignación de masa para cada sensor. De acuerdo con la Tabla 5.3, la masa asignada a los sensores de temperatura y humedad es la siguiente:

$$mass_T = 0.99, \quad mass_H = 0.99$$

El siguiente paso consiste en obtener el valor de la masa total, el cual esta definido como:

$$mass_{Total} = \left( \frac{mass_T + mass_H}{2} \right) \quad (6.11)$$

el resultado obtenido es  $mass_{Total} = 0.99$ . El último paso, consiste en evaluar el valor de la masa total con respecto a  $alpha$ , esta comparación esta definida como:

$$mass_{Total} \geq alpha \quad (6.12)$$

Al principio del ejemplo se declaró el valor de  $alpha = 0.7$ . De acuerdo con el Algoritmo 5.4, la condición  $0.99 \geq 0.7$ , es verdadera y por tanto, el sistema debe emitir una alerta de incendio. En la Fig. 6.9, se presenta la comparación de los datos recibidos por el servidor con respecto al comportamiento del modelo base, correspondiente al ejemplo desarrollado.

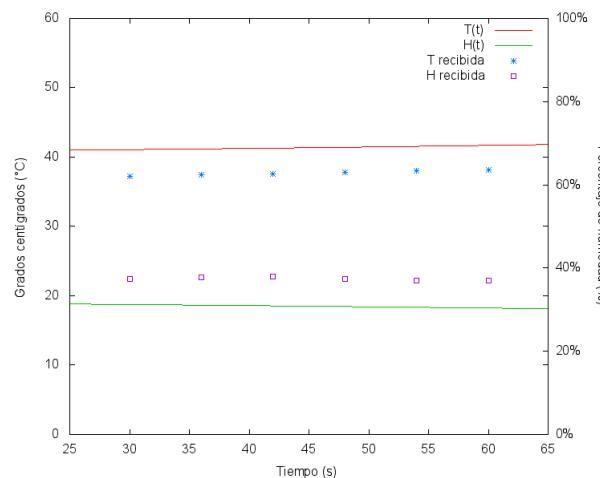


Figura 6.9: Razones de cambio con alto grado de similitud.

## 6.5. Resultados

Para la evaluación de los algoritmos que conforman la arquitectura propuesta, es necesario definir los parámetros que se observan en la Tabla 6.4. Estos valores pueden ajustarse para futuras evaluaciones.

Parámetro	Descripción
$N$	Tamaño de la ventana $W_T$
$T_{threshold}$	Umbral de temperatura
$P$	Período de muestreo del nodo sensor
$\Theta$	Tiempo de análisis límite
$nmax$	No. de evaluaciones de diferencias cuadráticas
$alpha$	Límite para considerar a un valor de masa, incendio

Tabla 6.4: Parámetros requeridos por el sistema de detección.

En nuestro caso, definimos en primer lugar los valores que se observan en la Tabla 6.5:

Parámetro	$T_{threshold}$	$\Theta$	$alpha$
Valor	1.01	360	0.7

Tabla 6.5: Parámetros para la evaluación.

El valor de  $T_{threshold}$  y  $alpha$  se determinaron de manera empírica de acuerdo a las observaciones en la fase de experimentación. Por otra parte, el valor de  $\Theta$  es una referencia a la duración de los experimentos, es decir, 360 segundos equivalentes a 6 minutos. A partir de estos valores, se realizaron evaluaciones para determinar el resto de los parámetros. Sólo se consideraron los experimentos en escenarios de sombra. Los resultados obtenidos, se muestran en las siguientes tablas:

$nmax$	5	10	15	20
Tamaño de $W_T$	Porcentaje de detecciones (%)			
5	58.33	58.33	41.67	41.67
10	87.5	87.5	87.5	87.5
15	100	100	95.83	95.83
20	100	100	100	100
25	100	100	100	100
30	100	100	100	100

Tabla 6.6: Experimentos totales: 24. Período de muestreo: 3 segundos.

$n_{max}$	5	10	15	20
Tamaño de $W_T$	Porcentaje de detecciones (%)			
5	91.67	79.17	66.67	62.5
10	100	100	95.83	95.83
15	100	100	95.83	95.83
20	100	100	95.83	95.83
25	100	100	95.83	95.83
30	100	100	95.83	95.83

Tabla 6.7: Experimentos totales: 24. Período de muestreo: 4 segundos.

$n_{max}$	5	10	15	20
Tamaño de $W_T$	Porcentaje de detecciones (%)			
5	100	100	100	100
10	100	100	100	100
15	100	100	100	100
20	100	100	100	100
25	100	100	100	100
30	100	100	100	100

Tabla 6.8: Experimentos totales: 23. Período de muestreo: 6 segundos.

$n_{max}$	5	10	15	20
Tamaño de $W_T$	Porcentaje de detecciones (%)			
5	100	100	100	100
10	100	100	100	100
15	100	100	100	100
20	100	100	100	95.45
25	100	100	95.45	86.36
30	100	95.45	90.91	31.82

Tabla 6.9: Experimentos totales: 22. Período de muestreo: 8 segundos.

Los resultados demuestran que el período de muestreo que mejor desempeño obtuvo, fue  $P = 6$ , utilizando cualquier tamaño de ventana  $W_T$  y cualquier número de evaluaciones  $n_{max}$ , de las opciones evaluadas.

De acuerdo a los resultados en la definición de parámetros, y como parte de la evaluación de la arquitectura propuesta, durante los meses de Abril y Mayo de 2014, se realizó una nueva serie de experimentos y se generó un nuevo modelo base. Esto debido a que las condiciones climáticas son distintas en esta etapa del año.

Obteniendo como resultado las siguientes funciones:

$$T(t) = -0.0000002477t^3 + 0.000112t^2 + 0.022441t + 28.8861 \quad (6.13)$$

$$H(t) = 0.0000000226t^3 - 0.000017t^2 - 0.027444t + 17.505 \quad (6.14)$$

Ambas funciones se observan en la Fig. 6.10:

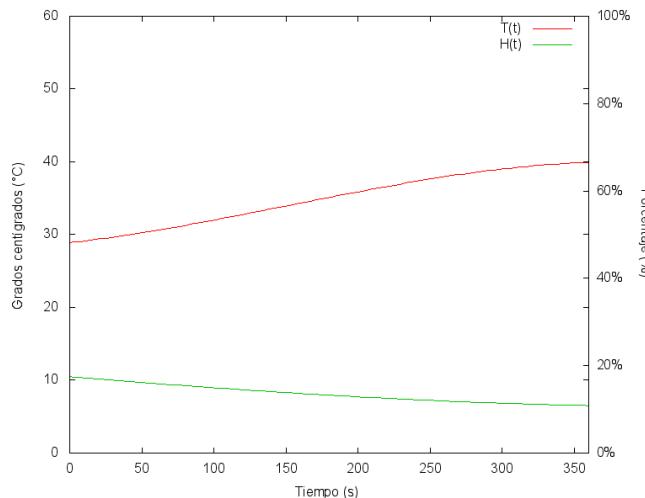


Figura 6.10: Modelo base de 2014.

Para la evaluación del sistema, sólo se realizaron experimentos en sombra. Los valores utilizados, fueron los siguientes:

Parámetro	<i>N</i>	<i>T<sub>threshold</sub></i>	<i>P</i>	$\Theta$	<i>nmax</i>	<i>alpha</i>
Valor	15	1.01	6	360	5	0.7

Tabla 6.10: Evaluación de experimentos realizados en Abril 2014.

Los resultados obtenidos se muestran en la Tabla 6.11:

Experimentos	Detectados
100	100

Tabla 6.11: Resultados de la evaluacion de Abril 2014.

# **Capítulo 7**

## **Conclusiones y Trabajo futuro**

En este trabajo de investigación, se propone una arquitectura para la detección de incendios forestales utilizando una red de sensores inalámbrica y métodos de fusión de información.

Una red de sensores inalámbrica, consiste en un conjunto de pequeños dispositivos con capacidad de comunicación inalámbrica. Se caracterizan por contar con diferentes tipos de sensores, utilizados para medir parámetros relacionados con las condiciones del ambiente. Son una tecnología flexible de fácil instalación y bajo costo, utilizada en diferentes campos de aplicación, tales como: asistencia médica, monitoreo de hábitats de especies animales, detección de eventos, rastreo de objetivos militares, etcétera.

Año con año los incendios forestales representan una de las principales causas de degradación ambiental en los ecosistemas del planeta. Provocan cuantiosas pérdidas materiales y en muchos de los casos también pérdida de vidas humanas. Algunas propuestas para combatir los incendios forestales que implementan una red de sensores inalámbrica utilizan: una gran cantidad de nodos, diferentes tipos de sensores, arquitecturas muy sofisticadas, o bien evalúan su desempeño a través de aplicaciones de simulación.

Para contrarrestar los efectos de este problema, se propone una arquitectura robusta basada en una red de sensores inalámbrica en conjunto con un algoritmo de detección de incendios de baja complejidad computacional. Esta propuesta, sólo implementa dos sensores: temperatura y humedad relativa; no se comparan magnitudes o umbrales, sino la razón de cambio de los datos obtenidos por el nodo sensor con respecto a un modelo base.

El modelo base, representa las condiciones de la temperatura y la humedad relativa en función del tiempo, en una situación de incendio y fue desarrollado utilizando el método de análisis de regresión. La evaluación del sistema propuesto, dio como resultado la detección de todos los eventos de incendio generados. La evaluación contempló, sólo escenarios experimentales en sombra. Uno de los factores que generó falsas alertas de incendio, fue la exposición

del nodo sensor ante los rayos del sol.

Como trabajo futuro, se tiene planeado construir modelos base por temporada del año o por etapa del día (mañana, tarde, noche) e incorporar la resolución de otros nodos sensores respecto al mismo evento.

Para ello se pretende construir grupos (cluster) de nodos sensores e implementar mecanismos de ahorro de energía. Con el objetivo de que sólo un nodo sensor monitoree la zona y en caso de percibir un incremento en la temperatura, transmita un mensaje a los nodos vecinos para ampliar la cobertura de monitoreo. Para determinar un evento de incendio, se tiene planeado implementar la teoría de la evidencia Dempster-Shafer, utilizando como evidencia la información proporcionada por el cluster.

# Bibliografía

- [1] Imrich Chlamtac, Marco Conti, and Jennifer J. N. Liu. Mobile ad hoc networking: imperatives and challenges. *Ad Hoc Networks*, 1(1):13–64, July 2003.
- [2] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.
- [3] CONAFOR. Incendios forestales en México temporada 2013. <http://www.conafor.gob.mx/>. Visitado por última vez Enero de 2014.
- [4] Jeroen Hoobeke, Ingrid Moerman, Bart Dhoedt, and Piet Demeester. An Overview of Mobile Ad Hoc Networks: Applications and Challenges. *Journal of the Communications Network*, 3:60–66, July 2004.
- [5] Benoît Latré, Bart Braem, Ingrid Moerman, Chris Blondia, and Piet Demeester. A survey on wireless body area networks. *Wirel. Netw.*, 17(1):1–18, January 2011.
- [6] S. Park and S. Jayaraman. Enhancing the quality of life through wearable technology. *Engineering in Medicine and Biology Magazine, IEEE*, 22(3):41–48, 2003.
- [7] Paolo Baronti, Prashant Pillai, Vince W. C. Chook, Stefano Chessa, Alberto Gotta, and Y. Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Comput. Commun.*, 30(7):1655–1695, May 2007.
- [8] Eduardo F. Nakamura, Antonio A. F. Loureiro, and Alejandro C. Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Comput. Surv.*, 39(3), September 2007.
- [9] Belur V. Dasarathy. Information fusion—what, where, why, when, and how? *Information Fusion*, 2(2):75–76, 2001.

- [10] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 95–107, New York, NY, USA, 2004. ACM.
- [11] B. V. Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, 1997.
- [12] Junguo Zhang, Wenbin Li, Zhifeng Xia, Guozhu Wang, and Zhen Wan. The implementation of communication for cc2430-based wireless sensor network nodes. In *Proceedings of the 5th International Conference on Wireless communications, networking and mobile computing*, WiCOM'09, pages 3373–3376, Piscataway, NJ, USA, 2009. IEEE Press.
- [13] Yeon sup Lim, Sangsoon Lim, Jaehyuk Choi, Seongho Cho, Chong kwon Kim, and Yong-Woo Lee. A fire detection and rescue support framework with wireless sensor networks. *Convergence Information Technology, International Conference on*, 0:135–138, 2007.
- [14] Agustian Taufiq Asyhari. Application of wireless sensor network monitoring system for fire detection in small scale environment. *Prosiding Seminar Nasional Aplikasi Teknologi Informasi (SNATI)*; 2006, 2009.
- [15] Oracle. <http://www.oracle.com>, 2014. Visitado por última vez Febrero de 2014.
- [16] Kewei Sha, Weisong Shi, and O. Watkins. Using wireless sensor networks for fire rescue applications: Requirements and challenges. In *Electro/information Technology, 2006 IEEE International Conference on*, pages 239–244, May 2006.
- [17] Liyang Yu, Neng Wang, and Xiaoqiao Meng. Real-time forest fire detection with wireless sensor networks. In *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, volume 2, pages 1214–1217. IEEE, 2005.
- [18] S. Madden, R. Szewczyk, M.J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Mobile Computing Systems and Applications, 2002. Proceedings Fourth IEEE Workshop on*, pages 49–58, 2002.
- [19] M. Bahrepour, N. Meratnia, and P.J.M. Havinga. Automatic fire detection: A survey from wireless sensor network perspective, December 2008.

- [20] O.S. da Penha and E.F. Nakamura. Fusing light and temperature data for fire detection. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 107–112, June 2010.
- [21] David M. Doolin and Nicholas Sitar. Wireless sensors for wildfire monitoring, 2005.
- [22] E. Zervas, O. Sekkas, S. Hadjiefthymiades, and C. Anagnostopoulos. Fire detection in the urban rural interface through fusion techniques. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE International Conference on*, pages 1–6, Oct 2007.
- [23] O. Sekkas, S. Hadjiefthymiades, and E. Zervas. A multi-level data fusion approach for early fire detection. In *Intelligent Networking and Collaborative Systems (INCOS), 2010 2nd International Conference on*, pages 479–483, Nov 2010.
- [24] Parul Mohindru and Rajdeep Singh. Multi-sensor based forest fire detection. *International Journal of Soft Computing and Engineering (IJSCE)*, 3:142–145, 2013.
- [25] Yunus Emre Aslan, Ibrahim Korpeoglu, and Özgür Ulusoy. A framework for use of wireless sensor networks in forest fire detection and monitoring. *Computers, Environment and Urban Systems*, 36(6):614–625, 2012.
- [26] Microsoft Visual Studio. <http://www.visualstudio.com/>, 2014. Visitado por última vez Abril de 2014.
- [27] E.M. Garcia, M.A. Serna, A. Bermudez, and R. Casado. Simulating a wsn-based wildfire fighting support system. In *Parallel and Distributed Processing with Applications, 2008. ISPA '08. International Symposium on*, pages 896–902, Dec 2008.
- [28] Shaohua Chen, Hong Bao, Xianyun Zeng, and Yimin Yang. A fire detecting method based on multi-sensor data fusion. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 4, pages 3775–3780 vol.4, Oct 2003.
- [29] Philip Levis and David Gay. *TinyOS Programming*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.
- [30] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesc language: A holistic approach to networked embedded systems. *SIGPLAN Not.*, 38(5):1–11, May 2003.
- [31] TinyOS. Stanford repository. <http://tinyos.stanford.edu/tinyos/dists/ubuntu/>, 2014. Visitado por última vez Febrero de 2014.

- [32] TinyOS. Coding Standard. <http://www.tinyos.net/tinyos-2.x/doc/html/tep3.html>, 2014. Visitado por última vez Febrero de 2014.
- [33] Migtool. <http://www.tinyos.net/tinyos-1.x/docnesc/mig.html>, 2014. Visitado por última vez Febrero de 2014.
- [34] Eclipse ide. <https://www.eclipse.org/>, 2014. Visitado por última vez Febrero de 2014.
- [35] AVHRR. <http://noaasis.noaa.gov/NOAASIS>, 2014. Visitado por última vez Junio de 2014.
- [36] MODIS. <http://modis.gsfc.nasa.gov/>, 2014. Visitado por última vez Junio de 2014.
- [37] Arnoldo Díaz-Ramírez, Luis A. Tafoya, Jorge A. Atempa, and Pedro Mejía-Alvarez. Wireless sensor networks and fusion information methods for forest fire detection. *Procedia Technology*, 3(0):69 – 79, 2012. The 2012 Iberoamerican Conference on Electronics Engineering and Computer Science.
- [38] Tzu-Chao Lin. Improving D-S evidence theory for data fusion system.
- [39] J. Stewart, J.H.R. M, E.F. López, and M.R. Bernal. *CALCULO DE UNA VARIABLE 4/E CONCEPTOS Y CONTEXTOS*. Cálculo: conceptos y contextos. Cengage Learning, 2010.
- [40] MEMSIC. Product datasheet. <http://www.memsic.com/>. Visitado por última vez en Enero de 2014.
- [41] MySQL. <http://www.mysql.com/>, 2014. Visitado por última vez Marzo de 2014.
- [42] IBM. <http://www-01.ibm.com/software/analytics/spss/>, 2014. Visitado por última vez Febrero de 2014.