ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

CARACTERIZACIÓN DE RETARDO DE 6LowPAN EN REDES LINEALES A GRAN ESCALA A PARTIR DE DATOS EXPERIMENTALES OBTENIDOS EN UNA RED DE ESCALA REDUCIDA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

CARLOS EDUARDO VELÁSTEGUI ALMEIDA

DIRECTOR: ING. CARLOS EGAS, MSc.

Quito, Julio 2022

AVAL

Certifico que el presente trabajo fue desarrollado por Carlos Eduardo Velástegui Almeida, bajo mi supervisión.

_____

ING. CARLOS EGAS, MSc
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

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

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

_____

Carlos Eduardo Velástegui Almeida

DEDICATORIA

Este trabajo de titulación que se presenta a continuación está dedicado a mi familia en especial a mi madre, mi padre, mi hermano por el inmenso esfuerzo para que pueda terminar mi carrera universitaria, porque nunca perdieron la fe y confiaron en mí a lo largo de mi vida, a mis hijas que han sido mi motor para salir delante de esta manera poder culminar mi carrera y que con su amor hicieron no me rindiera.

AGRADECIMIENTO

El agradecimiento va dirigido a mi madre, a mi padre y a mi hermano por su apoyo incondicional durante mis años de estudios y siempre apoyándome en todas mis dificultades para poder culminar esta etapa.

Además, un agradecimiento a la Escuela Politécnica Nacional y a mis profesores, por todas las enseñanzas y conocimientos compartidos en especial a mi director del proyecto Ing. Carlos Egas MSc. por el tiempo dedicado para poder culminar el mismo.

ÍNDICE DE CONTENIDO

RESUMEN

El presente proyecto de titulación se encuentra dividido en cuatro capítulos. En el primer capítulo se realiza una explicación breve de lo que es 6LowPAN (IPv6 sobre redes inalámbricas de área personal de baja potencia), adicional de un detalle de los dispositivos y equipos con los que este proyecto fue realizado.

Para el segundo capítulo realizamos la explicación de la red de sensores inalámbricos en topología lineal y los inconvenientes que poseen. Adicional se detallarán los diferentes programas necesarios para el desarrollo de las pruebas en los escenarios planteados y que nos ayuden a resolver este proyecto.

En el capítulo tres vamos a detallar todas las pruebas que vamos a realizar con los nodos para que se entienda de mejor manera los datos obtenidos, y el respectivo análisis que se debe analizar en el proyecto. Con los datos obtenidos se realiza la caracterización de una fórmula, con la cual se evalúe los retardos y permita proyectarse a una red de gran escala, adicional a eso se explica las correcciones que se hayan realizado a la fórmula luego de implementar las pruebas necesarias.

En el capítulo cuatro podemos apreciar las conclusiones y recomendaciones que se obtuvieron durante la elaboración y la caracterización de la fórmula como resultado del presente proyecto de titulación.

En los anexos encontramos manuales de instalación de los nodos, y el Sniffer como requisitos previos para la realización de las pruebas del proyecto.

PALABRAS CLAVE: WSN,6LOWPAN, PROCESAMIENTO, TIEMPO, NODO, RETARDO

ABSTRACT

In this project we have four chapters. The objective of this project is Characterize the delay times in the nodes and convergence times of the 6LowPAN protocol, in large-scale linear sensor wireless networks from experimental data obtained in a reduced scale network.

In the first chapter a brief explanation is made of the 6LowPAN additional detail of devices and equipment

For the second chapter we explain the wireless sensor network in linear topology and the problems they have. In addition, the different programs necessary for the development of the tests in the proposed scenarios and that help us solve this project will be detailed.

In chapter three I am going to detail all the tests that we are going to carry out with the nodes so that the data obtained is better understood, and the respective analysis that must be analyzed in the project. With the data obtained, the characterization of a formula is carried out, with which the delays are evaluated and allows it to be projected to a large-scale network, in addition to that, the corrections that have been made to the formula after implementing the necessary tests are explained.

The last chapter we can see the conclusions and recommendations that have been obtained during the preparation and characterization of the formula because of this degree project.

KEYWORDS: WSN, 6LOWPAN, PROCESSING, TIME, DELAY, NODE

1 INTRODUCCIÓN

El vertiginoso crecimiento de las redes de sensores inalámbricas WSN y los distintos usos que se le puede dar a las mismas, ha generado gran interés en estudiar más a fondo el comportamiento de estas redes de sensores, ya que al ir avanzando la tecnología adicional al avance que ha tenido el internet y tomando en cuenta el agotamiento de las direcciones en IPV4, nos hemos visto en la necesidad de buscar nuevas alternativas y es ahí donde nace la idea de utilizar IPV6, por esa razón entraremos a analizar de manera más profunda 6LowPAN.

En el presente proyecto se presentan los fundamentos teóricos requeridos para caracterizar los retardos. Se analiza de manera breve las redes WSN, IPV6, sus elementos, y protocolos. Se analiza la arquitectura 6LowPAN, esto nos permitirá visualizar de mejor manera las redes de sensores inalámbricos y las características de los sensores escogidos para la realización del presente proyecto.

Luego se presentan las pruebas realizadas para obtener la caracterización del modelo que permita ser utilizado en la evaluación del retardo en redes a gran escala.

1.1 OBJETIVOS

El objetivo general de este Proyecto Técnico es caracterizar los tiempos de retardo en los nodos y tiempos de convergencia del protocolo 6LowPAN, en redes inalámbricas de sensores lineales a gran escala.

Los objetivos específicos del Proyecto Técnico son:

- Evaluar el tiempo de convergencia del protocolo 6LowPAN en redes con topología lineal.
- Evaluar los tiempos de retardo por procesamiento en los nodos que operan con el protocolo 6LowPAN.
- Proponer un modelo matemático para evaluar teóricamente los tiempos de retardo en una red sensor inalámbrica con topología lineal a gran escala que opere con protocolo 6LowPAN

1.2 ALCANCE

La temática se enfoca en un escenario de redes de sensores inalámbricos con topologías lineales a gran escala con miles de saltos, tales como las que se utilizan en oleoductos o

ductos de agua, donde el objetivo principal censar eventos a lo largo de la infraestructura de red.

En el presente proyecto de titulación, a partir de una fórmula básica para el cálculo de retardos y tiempo de procesamiento, se pretende usar los resultados obtenidos en la experimentación de los sensores inalámbricos en redes de topología lineal, para crear un modelo matemático descriptivo en base a las variables, parámetros y restricciones que se evidencien durante la experimentación, el modelo obtenido permite realizar una cálculo de los retardos que se ajuste a la realidad; este modelo permite explorar, estudiar y proyectar las redes de sensores inalámbricos 6LowPAN en topología a gran escala; sin proyecto final demostrable partiendo de una red de sensores inalámbricos a escala reducida.

Para obtener los retardos de manera experimental que nos permitan evaluar la fórmula matemática, se utilizará un prototipo de red de sensores inalámbricos 6LowPAN, el mismo que será implementado utilizando el Kit de desarrollo de WSN1 para IPv6 desarrollado por Libelium e IBM, denominado Moterunner y un Sniffer Atmel, que se encargará de realizar la captura de paquetes para la adquisición de datos que nos ayuden a generar la fórmula.

El prototipo de red de sensores inalámbricos 6LowPAN a implementarse, está ilustrado en la Figura 1.1, el mismo que se encuentra conformado por 4 nodos inalámbricos y el nodo Gateway; el nodo Gateway tiene dos interfaces: la interfaz inalámbrica que le permite conectarse la red 6LowPAN y la interfaz cableada que le permitirá la conexión a la PC y un capturador de paquetes (sniffer)

La conexión entre el nodo Gateway y la PC, ilustrado en la Figura 1.1, únicamente se puede configurar con IPv4, por lo que es necesario implementar un túnel IPv6/IPv4, para que los datos IPv6 de la red sensores inalámbricos lleguen a la PC, para ser procesados.

Se desarrollará un programa para transmitir un dato y otro modificando el payload para que se ajuste a las características necesarias para obtener los datos y crear la expresión matemática.

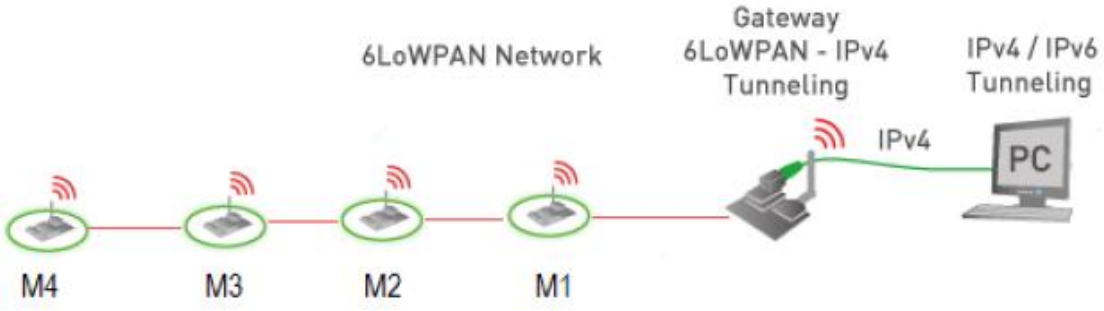

Figura 1.1 Red Física (topología Lineal) en el prototipo de red 6LowPAN.

Las pruebas se realizan en un ambiente externo, considerando un prototipo de cinco nodos fijos incluido el Gateway. La distancia entre los nodos contiguos se manejó de tal manera que esta topología cumpla con las características requeridas para evaluar los tiempos de retardo y de convergencia ante nodos y enlaces fallidos, para luego realizar proyecciones con cientos de saltos, utilizando modelos matemáticos. Los parámetros que se tomarán en cuenta son, el tiempo de demora en asignar los identificadores a los nodos por medio de 6LowPAN, el tiempo de convergencia en la recuperación de la red, el tiempo de transmisión, el tiempo de procesamiento el cual se medirá mediante la captura de paquetes cuando llega el paquete al nodo, es procesado y es finamente transmitido para ello se realizará un programa para configurar los nodos de tal manera que nos permita obtener los resultados propuestos.

1.3 MARCO TEÓRICO

1.3.1 RED INALAMBRICA DE SENSORES (WSN) [1] [2]

Las redes inalámbricas de sensores (WSN), son un conjunto de dispositivos autónomos e interconectados entre sí y distribuidos espacialmente en un área, que tienen la capacidad de tomar datos del medio, procesar el nodo y transmitir hacia los demás nodos de la red, muchos de estos sensores no solamente son capaces de captar la información, sino que actúan en el medio, es decir se comportan también como actuadores, a estos dispositivos se los denomina nodos.

Los sensores inalámbricos por lo general están formados por: un módulo con protocolo de comunicación integrado, los sensores que forman parte del nodo, un microcontrolador que se encarga del procesamiento de los datos, una batería, una antena que puede estar integrada o puede ser montable, en caso de necesitar nodos más complejos se debería a que el procesamiento de los datos es mayor.

Estas redes aparecieron inicialmente como redes militares o de uso militar, sin embargo, con el paso del tiempo se llegaron a dar cuenta que son de gran ayuda para producir beneficios a los seres humanos.

Las redes de sensores inalámbricos tienen diversas aplicaciones en la actualidad, ya que poseen componentes de hardware y de software adaptable para cada aplicación, además de que han ido avanzando en capacidad de adquisición y procesamiento de datos, así también la autonomía para su funcionamiento ha ido avanzando.

De esta manera a esta red de sensores se la considera como la mejor forma para poder crear una comunicación entre todos los objetos alrededor, es decir una opción para poder implementar el internet de las cosas IOT. Cabe destacar que estamos frente a las limitaciones que tienen estas redes de sensores con respecto a IOT, puesto que el mismo utiliza protocolo IP.

Por esta razón el IETF (Internet Engineering Task Force) el grupo de trabajo de Ingeniería de internet, primer organismo de normas de internet ha creado el estándar para hacer frente a estas limitaciones permitiendo la integración de IPV6 en las redes de sensores inalámbricos llamado 6LowPAN, cuyo objetivo principal es mantener que el consumo de energía sea bajo en la red de sensores.

1.3.2 PARÁMETROS PARA CONSIDERAR EN LA IMPLEMENTACION DE UNA RED WSN [3] [4] [5]

Al momento de implementar una red WSN, existen algunos parámetros que se deben tomar en cuenta, dentro de estos están los siguientes.

Escalabilidad: Permite que la red continue su crecimiento sin ningún inconveniente, de tal manera que podemos agregar más nodos con ayuda de los protocolos.

Cobertura: Se entiende como las zonas que se encuentran cubiertas por la señal de los sensores. Va en conjunto con la escalabilidad de la red WSN, y así se pueda interactuar entre todos los nodos de la red.

Tolerancia a fallos: Conocido también como confiabilidad de la red; el objetivo como tal es generar una red robusta que garantice la comunicación, considerando los diferentes casos como son una posible destrucción física del nodo, las condiciones ambientales en las que trabajen, el agotamiento de las baterías, y fallas en la configuración de los dispositivos, una manera de poder solventar este tipo de situaciones es a través de la redundancia.

Energía: Se podría considerar que es uno de los parámetros más importantes, la energía consumida por el sensor depende directamente del nivel de potencia en el que vamos a trabajar, hay que considerar que los sensores van a trabajar con baterías y las mismas van a tener un tiempo de duración después de ello deberían ser reemplazadas para que tengan un correcto funcionamiento. Hay que acotar que se debe realizar un mantenimiento correcto de los sensores para evitar que la red falle, además hay que considerar el consumo de energía en la transmisión de información de tal manera que sea eficiente, tratando de evitar la retransmisión de datos y que de esa manera se ahorre energía en algunos procesos.

Redundancia: Se puede entender que la redundancia es la colocación de nodos estratégicamente para que la red sea lo más confiable, de tal manera que evite fallos en la transmisión de información, reenviando paquetes incluso cuando exista la perdida de comunicación con otro nodo, mediante la reconfiguración de ruta de envío hasta que la información alcance su destino.

1.3.3 TOPOLOGÍAS DE UNA RED INALÁMBRICA DE SENSORES [6] [4]

En una red de sensores inalámbricos existen diferentes tipos de topologías que conectan un nodo con otro nodo, dependiendo si existe un nodo coordinador o no, el cual se encarga del ordenamiento de los nodos y la recolección de los datos. Dentro de las topologías que podemos encontrar se detallan las siguientes:

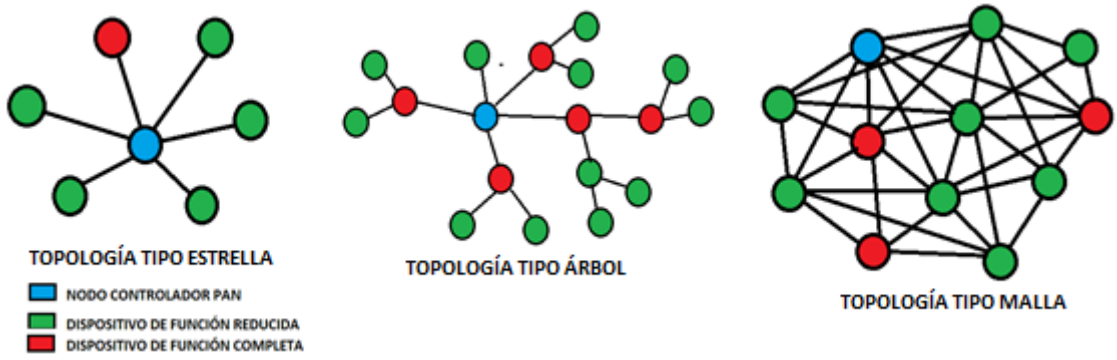

Figura 1.2. Topologías de una WSN [7]

- Topología Estrella: Esta topología se caracteriza por tener un solo nodo coordinador y todos los sensores o nodos se van a comunicar solamente con el nodo central, así todos los datos de la red deben pasar por el nodo principal o central, es la puerta de enlace con otros dispositivos de la red. Para poder comunicarse un nodo con otro se deben enlazar primeramente con el nodo coordinador, por esta razón es

una red que no alcanza grandes distancias, sin embargo, la tasa de transmisión de los datos es alta, por esta razón la confiabilidad de la red también es alta, ya que la tasa de existencia de fallos es baja, aunque si el nodo coordinador tiene algún fallo, simplemente la red se pierde, puesto que no hay otra ruta alternativa.

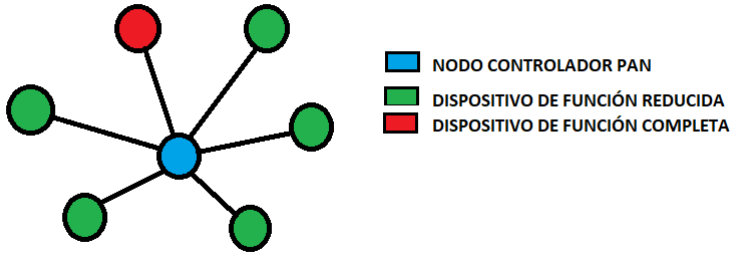

Figura 1.3. Topología tipo estrella [6]

- Topología Malla: Su característica principal es que todos los nodos pueden comunicarse entre sí. Además, cada nodo puede enviar y recibir mensajes sin la necesidad de una puerta de enlace, también es más tolerante a fallos, ya que busca caminos alternativos para hacer que llegue el mensaje, este tipo de red se autoconfigura cuando existe una pérdida del nodo.

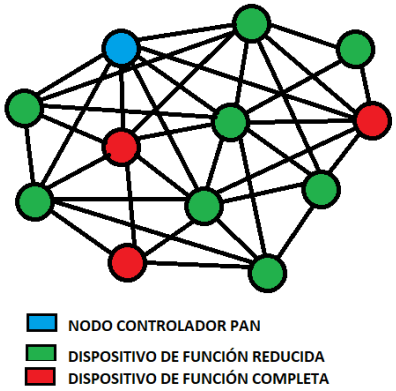

Figura 1.4. Topología tipo malla [6]

- Topología de Árbol: Se caracteriza por ser una topología que tiene jerarquía, parte desde un nodo coordinador que a su vez envía la información a un nodo hijo, que puede ser de funciones completas o reducidas, dependiendo para que se van a utilizar las redes, tiene mayor cobertura y alcance que las topologías anteriores, considerando que su latencia también aumenta.

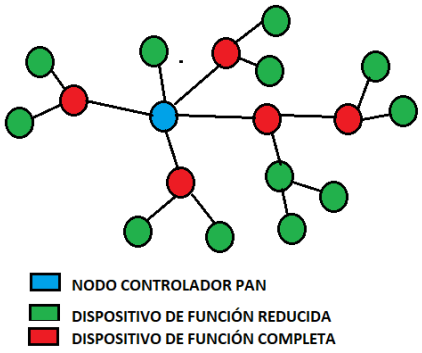

Figura 1.5. Topología tipo árbol [6]

- Topología Lineal: Es un tipo de red con topología tipo árbol de una sola rama, se caracteriza por tener sensores colocados en línea recta de esta manera tiene escalabilidad, se colocan sensores con cierta distancia para que la red se autoconfigure, el alcance de los sensores que se van a utilizar en espacio abierto es de 60 a 70 metros , de manera que la eficiencia de la red es mayor, adicional a eso todos los nodos son transmisores y receptores de información, y su alcance es mayor que los otros tipos de redes ya que puede llegar a cientos de kilómetros.

Figura 1.6. Topología lineal [6]

1.3.4 ELEMENTOS DE UNA RED INALÁMBRICA DE SENSORES WSN [3]

Como se mencionó anteriormente, las redes inalámbricas de sensores se encuentran constituidas en base a nodo sensor, Gateway, sensores, estación base y red.

A continuación, se va a detallar los elementos que componen la red de sensores.

- Gateway: Se encarga de captar la información de WSN y enviarla hacia la estación, este nodo suele estar conectado siempre al computador o a la red internet.
- Nodo Sensor: Son dispositivos pequeños cuyo objetivo es obtener una variable física, tales como humedad, temperatura, movimiento, entre otras variables, de tal manera que los datos se transforman en impulsos eléctricos para que procese en el nodo sensor.

- Estación Base: Nos indica la información de cada nodo sensor de la red a través de una interfaz amigable con el usuario, puede ser local o remota, esta estación base puede tener gran capacidad de procesamiento porque en muchos casos maneja protocolos TCP/IP.
- Red inalámbrica: Por lo general basada en los estándares IEEE 802.15.4 y Zigbee sin embargo, puede tener protocolos propietarios.

1.3.5 Estándar 802.15.4

El estándar de telecomunicaciones IEEE802.15.4, se lo puede definir como una solución para redes inalámbricas de área personal PAN con baja tasa de datos (LR-WPAN), la versión del estándar fue liberada en 2006. [8]

Una LR-WPAN es una red de bajo costo, que permite conectividad en aplicaciones que utiliza dispositivos con consumo de potencia limitada, y una tasa menor de 250 kbps. Dentro de las facilidades de LR-WPAN, tenemos la instalación, la confiabilidad en la transferencia de datos mediante el uso de mecanismos, como acceso al medio por CSMA-CA, mensajes de acuse de recibo ACK y la retransmisión de datos, bajos costos de operación, duración razonable de batería, y manteniendo un protocolo simple y flexible.

En el estándar IEEE 802.15.4, se definen dos tipos de dispositivos los cuales son los de funcionalidad completa FFD, que puede trabajar como coordinador PAN, con capacidad de crear y controlar una red PAN, coordinador que realiza sincronización de los dispositivos cuando emplea el modo ranurado,

El otro tipo de dispositivo es el RFD, que es un dispositivo de funciones reducidas que tienen una mínima implementación, no tiene funciones de coordinador PAN; es también llamado como dispositivo final, puede asociarse a un solo FFD, no necesita transmitir grandes cantidades de datos.

Las topologías que se pueden emplear fueron detalladas en el punto 1.3.3.

1.3.6 Estándar IEEE 802.15.4 [8]

El estándar IEEE 802.15.4 se enfoca a redes con bajo consumo de energía, tasas bajas de transmisión, y una arquitectura de red sencilla; para esto, se definen dos capas, la capa física y la capa MAC o de acceso, y control al medio. Estas se encuentran basadas en el modelo OSI como podemos observar en la figura 1.7, cada capa es responsable de ofrecer servicios a la capa superior gracias a la subcapa de enlace lógico LLC que ya viene definida en el estándar 802.2

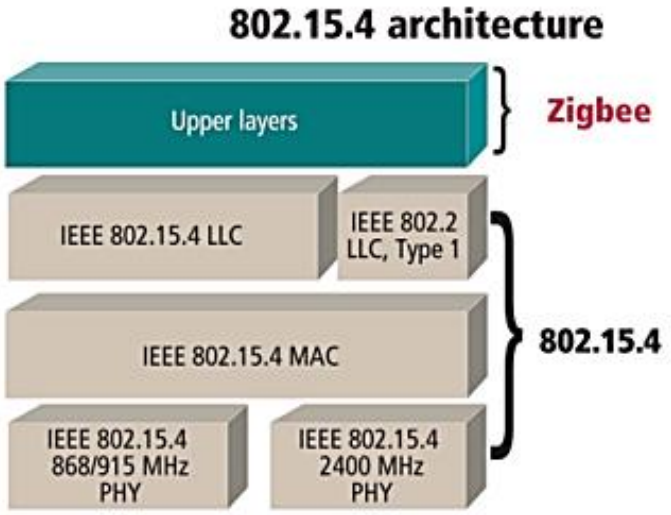

Figura 1.7. Arquitectura 802.15.4 [9]

1.3.6.1 Capa Física PHY [6] [10]

Su función es transmitir y recibir datos, para ello ocupa un canal de radio definido previamente, además para la modulación ocupa una técnica específica.

El estándar IEEE 802.15.4 funciona en tres bandas de frecuencia 2.4GHz, 915MHz y 868 MHz. Dentro de las bandas de frecuencia se tienen varios canales, los cuales podemos encontrar un canal en la banda de 868 y 868.6 MHz, 10 canales en la banda de 902 y 928 MHz, y 16 canales en la banda de 2.4GHz a 2.4835GHz. Las técnicas de modulación que ocupa dependen de la banda de frecuencia en la que este trabajando, para los 868 MHz y 915 MHz utiliza BPSK que es modulación por desplazamiento de fase binaria, para la banda de los 2.4 GHz utiliza la modulación O-QPSK que es modulación por desplazamiento de fase en cuadratura y tasas de bit a nivel físico que a continuación se de detallan.

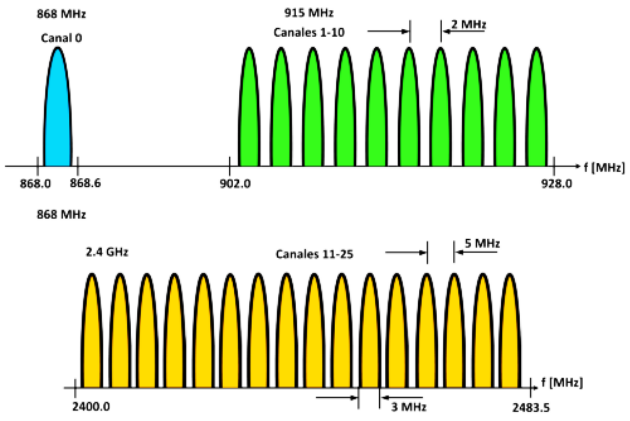

Figura 1.8. Frecuencias centrales de los canales IEEE 802.15.4 [8]

El Estándar IEEE802.15.4., para realizar las transmisiones utiliza DSSS (Direct Sequence Spread Spectrum) es decir Secuencia directa de Espectro ensanchado, este método facilita la filtración de interferencias, con la desventaja de que se tiene más ineficiencia por el ancho de banda.

Tabla 1.1. Características del estándar IEEE802.15.4 – 2003 [10]

Banda de Frecuencia	Número de Canales	Técnica de Ensanchamiento (Spreading)	Modulación	Velocidad de símbolo por canal(kbaud)	Tasa de bits por canal (kbps)
868 MHz	1	Binary DSSS	BPSK	20	20
915 MHz	10	Binary DSSS	BPSK	40	40
2.4 GHz	16	16-array DSSS	O-QPSK	62.5	250

El estándar IEEE802.15.4 - 2006, utiliza técnicas de modulación, que nos ayudan a ensanchar el espectro como PSSS y DSSS (Secuencia Paralela de Espectro Ensanchado y Secuencia Directa de Espectro Ensanchado), a continuación, las características de IEEE 802.15,4 - 2006 en la siguiente tabla.

Tabla 1.2. Características de IEEE 802.15,4 – 2006 [10]

Banda de Frecuencia	Número de Canales	Técnica de Ensanchamiento (Spreading)	Modulación	Velocidad de símbolo por canal(kbaud)	Tasa de bits por canal(kbps)
868 MHz	1	20-bit PSSS	ASK	12.5	250
915 MHz	10	5-bit PSSS	ASK	50	250
868 MHz	1	16-array DSSS	O-QPSK	25	100
915 MHz	10	16-array DSSS	O-QPSK	62.5	250

1.3.6.1.1 Activación y Desactivación del Transceptor [6]

El modo de operación del Transceptor es Half duplex, es decir que puede transmitir, pero no recibir a la vez o viceversa. Una ventaja es que el transceptor puede operar en un modo dormido, por lo que la subcapa MAC puede manejar los tres estados de operación como son el dormido, transmisión, y recepción.

1.3.6.1.2 Detección de Energía (ED) [11]

La Detección de Energía (ED) es una estimación del nivel de potencia de energía de las señales recibidas dentro del canal, para lo cual, el dispositivo si desea realizar la detección de energía debe encender el dispositivo y sintoniza el receptor. La sensibilidad del receptor

es el nivel más bajo de energía que el receptor puede detectar, y demodular con una tasa de error de paquete de menos de 1%. [8]

1.3.6.1.3 Evaluación de Canal Libre (CCA) [6]

Indica si el canal esta libre o se encuentra ocupado. La evaluación del canal libre (CCA) se lo puede realizar en tres modos que son los siguientes.

- Modo Detección de Portadora (CS), controla que los participantes de la red comprueben que el medio se encuentre en realidad libre solo así se inicia la transmisión de los datos.
- Modo Detección de Energía (ED), en este caso vamos a tener un valor umbral de la ED, si el valor detectado es superior al del umbral consideramos que el canal está ocupado.
- Modo Detección de Portadora con Detección de Energía (EC y DS), en este caso se considera que el canal está ocupado, cuando cumple con los dos casos anteriores.

El estándar define el valor de CCA en 8 periodos por símbolo el cual se define como la velocidad del símbolo como se aclara en el punto 2,4.

1.3.6.1.4 Primitiva de Servicio [11] [12]

Las primitivas son los servicios que brinda una capa superior a una inferior, cada capa tiene funciones diferentes y puede ofrecer servicios diferentes a sus capas adyacentes. La capa superior emplea un punto SAP (Service Access Point), punto de acceso al servicio) para solicitar servicios a la capa inferior, existen cuatro tipos de servicio que son:

- Petición (Request), sirve para pedir un servicio
- Indicación (Indication), Se usa para mostrar un evento importante sea este una solicitud de servicio o un evento interno
- Respuesta (Response), en este enviamos lo solicitado
- Confirmación (Confirm), se genera en la capa inferior después de que le enviaron lo solicitado y que el servicio fue completado

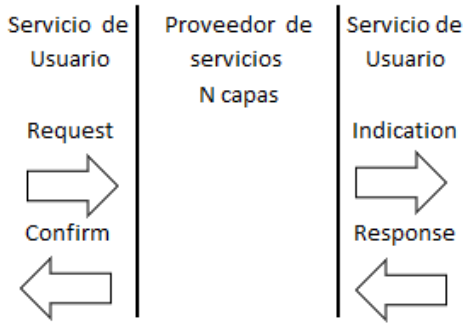

Figura 1.9. Servicio de primitivas según el estándar 802.15.4 [12].

1.3.6.1.5 Formato de la Trama PPDU de capa Física [13]

Si se va a transmitir datos dentro de una red PAN a un dispositivo, la subcapa MAC es la que genera un pedido de servicio, que su capa inferior la capa física trata de completar. Este paquete generado por la subcapa MAC, es el MPDU MAC, es decir la unidad de datos del protocolo MAC al pasar a la capa física pasa como (PSDU, unidad de servicio de datos físico) esta PSDU se une con la cabecera de sincronización (SHR), y la cabecera física (PHR, todos en conjunto forman la unidad de datos de protocolo físico (PPDU PHY) la longitud máxima es de 133 bytes.

PPDU de Capa Física				
Secuencia de preámbulo	Delimitador de inicio de trama (SFD)	Longitud de la trama 7 bits	Reservado 1 bit	Carga Útil de Capa Física (PHY)
Cabecera de Sincronización SHR		Cabecera PHR		PSDU

Figura 1.10. Formato de trama PPDU [13]

La cabecera de sincronización (SHR) tiene como finalidad principal sincronizar el dispositivo y fijar la cantidad del flujo de bits, está dividida en dos partes que son la secuencia de preámbulos y el delimitador de inicio de trama. La secuencia de preámbulo va a permitir que el dispositivo alcance a sincronizarse por medio de los símbolos con el mensaje entrante. El delimitador de inicio de trama (SFD), indica que termina el encabezado, y que va a iniciar el paquete de datos. El SFD tiene un tamaño de 8 bits la mayoría de las veces muestra la secuencia de bits 11100101. La cabecera PHR contiene la longitud de la trama, es decir la cantidad de bytes del payload. La PSDU contiene la carga útil la cual proviene de la subcapa MAC, este va a tener un tamaño variable entre 0 y 127 bytes.

1.3.6.2 CAPA DE CONTROL DE ACCESO AL MEDIO MAC [6]

La subcapa Mac es la que brinda el interfaz para comunicar a la capa física con las capas superiores, esto lo realiza mediante dos interfaces, MAC Management Service, servicio de administración de la subcapa MAC o MAC Layer Management Entity (MLME).

En base a la figura 1.11 podemos mostrar como la subcapa MAC relaciona la capa Física con la capa de red NW. Dentro de la subcapa MAC tenemos la MAC-PIB que es una base de datos que tiene la capa MAC, al igual que la PHY donde están almacenados los valores definidos por el estándar para el correcto funcionamiento. La MCPS-SAP se encarga de enviar los datos a las capas superiores mientras que la MLME-SAP y PLME-SAP son las que permiten comunicarse con las capas vecinas.

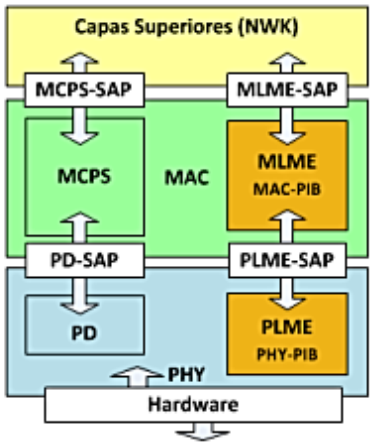

Figura 1.11. Interfaz entre la subcapa MAC y sus capas vecinas [8]

1.3.6.2.1 Modos de Operación [6]

El protocolo MAC tiene la facilidad de trabajar en dos modos, los cuales son el Beacon Enable y Non Beacon Enable o modo ranurado y no ranurado; vamos a ver los modos de operación, y las formas de acceso al medio CSMA-CA que tiene IEEE802.15.4 en la subcapa MAC.

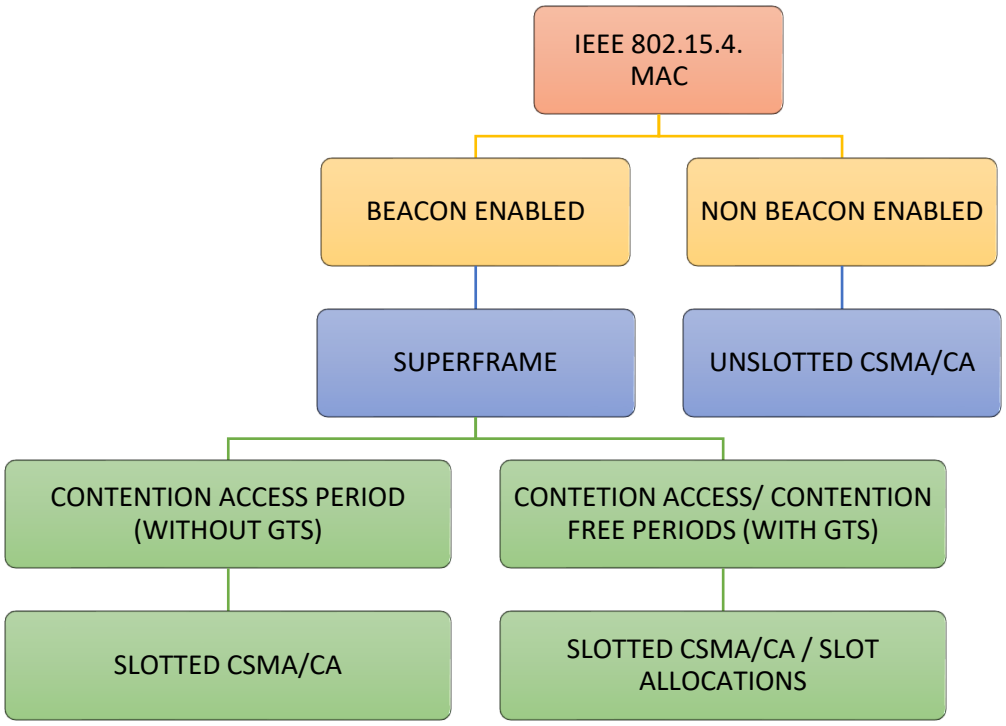

Figura 1.12. Modos de operación de IEEE 802.15.4.[6]

1.3.6.2.1.1 Modo Beacon Activado o modo ranurado [6]

Este modo, utiliza la estructura de supertrama para controlar las comunicaciones entre dispositivos de una misma red PAN, el nodo coordinador es el encargado de generar esta supertrama y enviarla periódicamente a los otros dispositivos de la red a través de una trama beacon.

Esta supertrama se encuentra dividida en 16 ranuras de igual tamaño, que van seguidas de un tiempo inactivo el cual se encuentra preestablecido, la supertrama se encuentra limitada por dos tramas beacon consecutivas, y va a estar formada por un periodo de acceso a la contención (CAP), y también puede incluir un periodo libre de contención (CFP).

Vamos a describir los dos tipos de supertrama que son con GTS, y sin GTS (GTS Ranuras de tiempo garantizado).

- Si un dispositivo desea comunicarse, y el periodo de acceso a la contención está restringido, el dispositivo debe competir con demás dispositivos de la red PA usando CSMA-CA, de esta manera seria una supertrama sin GTS.

Figura 1.13. Estructura de una supertrama sin GTS. [6]

- Si deseamos tener Calidad de Servicio (QoS), se debe definir un periodo de libre contingencia (CFP), el nodo coordinador de la PAN es encargado de activar la CFP y consiste en asignar tiempos de intervalos garantizados, este CFP es parte de una supertrama y empieza inmediatamente después del CAP, podemos asignar hasta siete GTS y cada uno de ellos puede usar más de un intervalo de tiempo. Si ocupamos este modo de configuración de la supertrama, todo tipo de comunicación basada en contención debe estar concluido antes de poder iniciar el CPF; en cambio el nodo debe confirmar que la transmisión de datos se haya completado antes del siguiente GTS, este modo de comunicación solo se emplea entre coordinador PAN y un dispositivo.

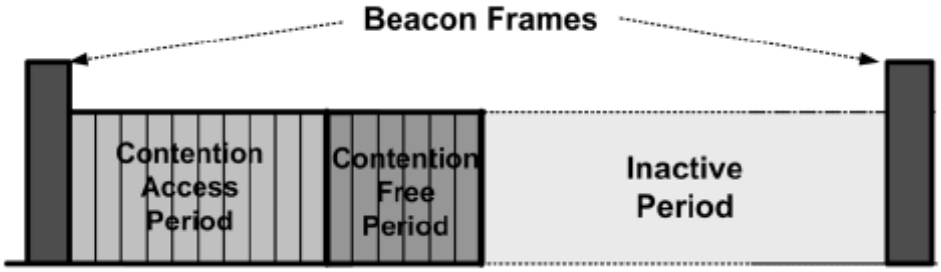

Figura 1.14. Estructura de una supertrama con GTS. [6]

1.3.6.2.1.2Estructura de la supertrama [6] [8]

La supertrama está dividida en 16 ranuras de tiempo iguales, empieza y termina con el inicio de otro beacon y se caracteriza por tener un periodo activo e inactivo.

Para delimitar el inicio de una trama se usa el primer beacon, el cual es transmitido en el primer slot, además de que nos ayuda a sincronizar los dispositivos enlazados y permite agregar nuevos dispositivos a la PAN. Se puede emplear intervalos de tiempo garantizados

o GTS, los cuales son asignados por el nodo coordinador en los casos de que se necesite un ancho de banda específico o se requiere de una red con baja latencia.

Los intervalos GTS conforman el CFP dentro de un CFP, pueden existir 7 GTS, estos GTS pueden durar uno o más slots. Así podemos notar la estructura de la supertrama.

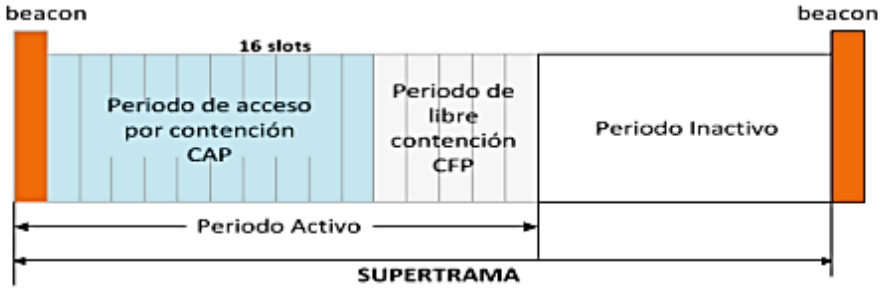

Figura 1.15. Estructura de una supertrama. [8]

1.3.6.2.1.3Modo Non Beacon Enable o modo no ranurado [6]

Cuando el nodo coordinador de la red PAN trabaja en este modo, no se emplea la supertrama ni beacon. El mecanismo CSMA/CA no ranurado proporciona el control de acceso al medio.

1.3.6.2.2 Mecanismos de acceso al medio CSMA-CA [6]

El estándar IEEE 802.15.4 nos da dos versiones de mecanismo de acceso al medio CSMA-CA, estas son, la versión ranurada que emplea el modo beacon (CSMA-CA-ranurado), y la versión no ranurada que emplea el modo no beacon (CSMA-CA no ranurado), cualquiera de estas dos versiones utiliza el algoritmo de backoff, donde un periodo de backoff es igual a 20 símbolos, este valor es definido por el protocolo MAC, considerada como la unidad básica de tiempo.

En una red que opera en modo ranurado, el nodo coordinador es el que determina el periodo de backoff, a comparación de una red no ranurada donde cada nodo con su reloj interno determina el periodo de backoff.

Para configurar el acceso al medio CSMA-CA emplea tres variables las cuales son:

- Número de Backoff (NB), es la cantidad de intentos que requiere el algoritmo para intentar acceder el canal, el número de intentos permitidos macMaxCSMABackoffs es (0,5), si este se pasa de ese valor ocasiona un error y el número de backoff se inicializa a cero antes de cada nuevo intento de transmisión.

- Ventana de contención (CW), se utiliza en CSMA-CA ranurado solamente es la cantidad de periodos de backoff para que el canal se encuentre disponible antes de que pueda iniciar una transmisión.
- Exponente de Backoff (BE), es el tiempo que debe esperar un dispositivo antes de censar y determinar si el canal esta libre o no. Los valores que puede tomar van a depender del tipo de modulación y la frecuencia, estos valores están comprendidos entre un valor mínimo de 0 y el máximo que se encuentra entre 3 y 8.

1.3.6.2.2.1 Mecanismo CSMA-CA ranurado (slotted CSMA-CA) [6] [8]

CSMA ranurado utilizado en redes con ranuras de tiempo, al momento que el coordinador inicia una trama beacon, está se iniciacon el primer periodo de backoff, los límites de la supertrama están alineados con los periodos de backoff de todos los dispositivos que se encuentran en la red PAN.

El algoritmo inicia con el Numero de backoff, la ventana de contención y el exponente de backoff, el NB inicia en cero y CW en dos, como se ha mencionado anteriormente, si se tiene activado el modo de extensión de la batería activada, el BE toma el valor de 2, es el valor de una red no ranurada, después de esto el algoritmo genera retardos aleatorios para evitar colisiones donde el retardo va a depender de los periodos de backoff, si este periodo no termina antes de que termine el periodo de acceso por contención se detiene e inicia el siguiente periodo de acceso por contención.

Si el periodo de backoff culmina en los límites del CAP se vuelve a censar el canal para ver si este se encuentra disponible, y al mismo tiempo la ventana de contención disminuye su valor en uno, si el valor de la ventana de contención es diferente de cero el algoritmo empieza otro CCA.

La CCA se ejecuta por dos veces antes de iniciar la transmisión debido a que el algoritmo inicia en 2 la ventana de contención, si el canal está ocupado la ventana se reestablece en el valor de 2 y el valor de NB Y BE se incrementa en una unidad. Si el valor de NB es máximo, el algoritmo indica que ocurre un error durante la transmisión. Caso contrario si NB no toma su valor permitido, empieza un nuevo CCA, después de ejecutar el algoritmo de backoff, en el siguiente diagrama de flujo podemos visualizar el mecanismo CSMA-CA ranurado.

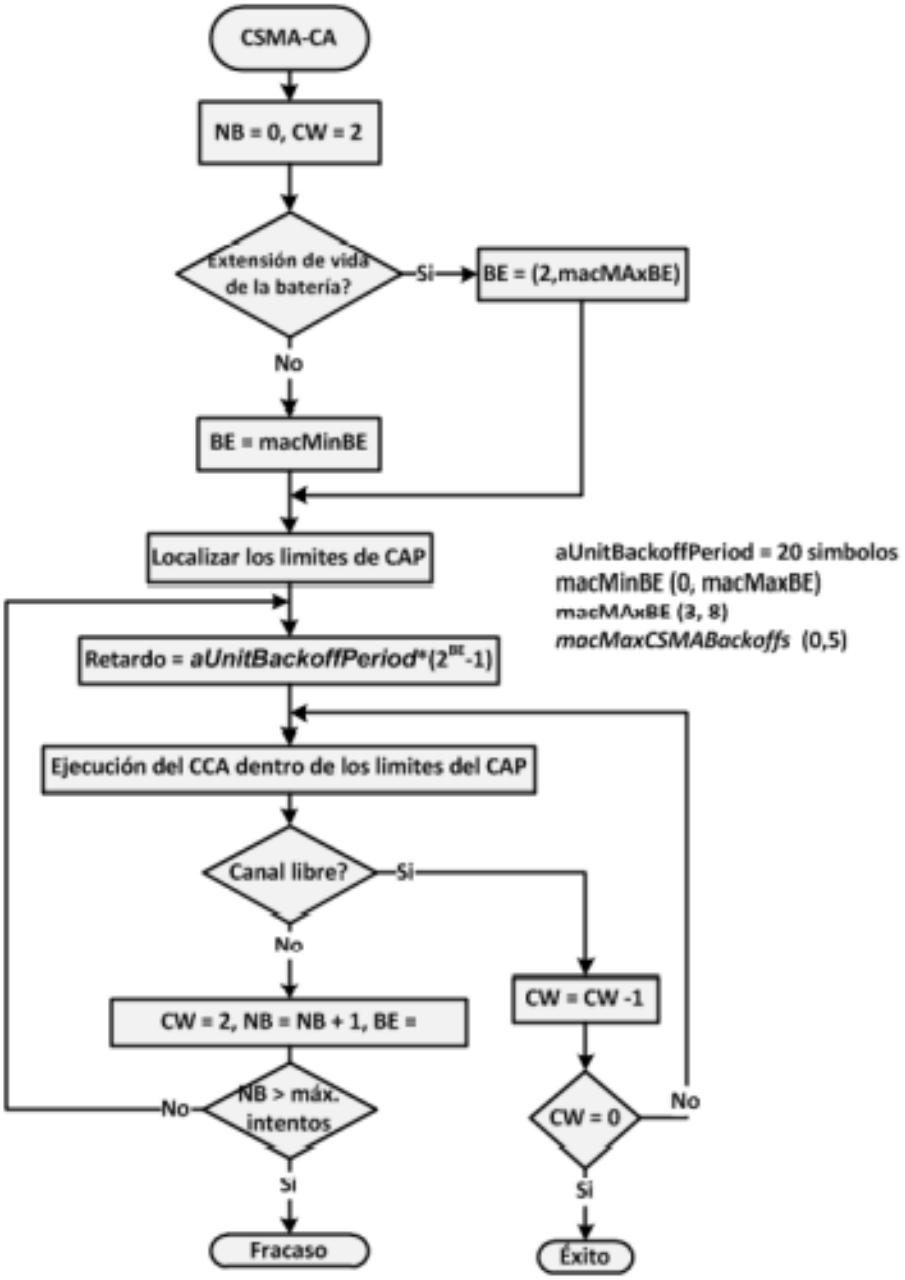

Figura 1.16. Algoritmo de backoff en una red beacon. [8]

1.3.6.2.2.2Mecanismo CSMA-CA no ranurado (Unslotted CSMA-CA)

Es empleado por las redes no beacon o no ranurado es el CSMA-CA, antes de que cualquier dispositivo desee transmitir, primeramente, se ejecuta el algoritmo de backoff, después de que culmina este tiempo de espera, el dispositivo transmite, caso contrario se ejecuta nuevamente el algoritmo, las tramas ACK se envían sin emplear el mecanismo CSMA-CA.

Cuando inicia el algoritmo de backoff las variables NB y BE, toman los valores de cero el NB y el BE = *manMinBE*, estas variables dan como resultado el periodo de backoff, después de ello el dispositivo realiza un CCA para censar el canal, si el canal está disponible inicia la transmisión, caso contrario el canal ocupado de las variables NB y BE, produce un incremento de esta manera el algoritmo se repite hasta que el número de intentos de transmisión fallidos sea menor al número de intentos permitidos, si no es así la transmisión no se lleva a cabo, a continuación se detalla a través de un diagrama de flujo el mecanismo de funcionamiento del mismo.

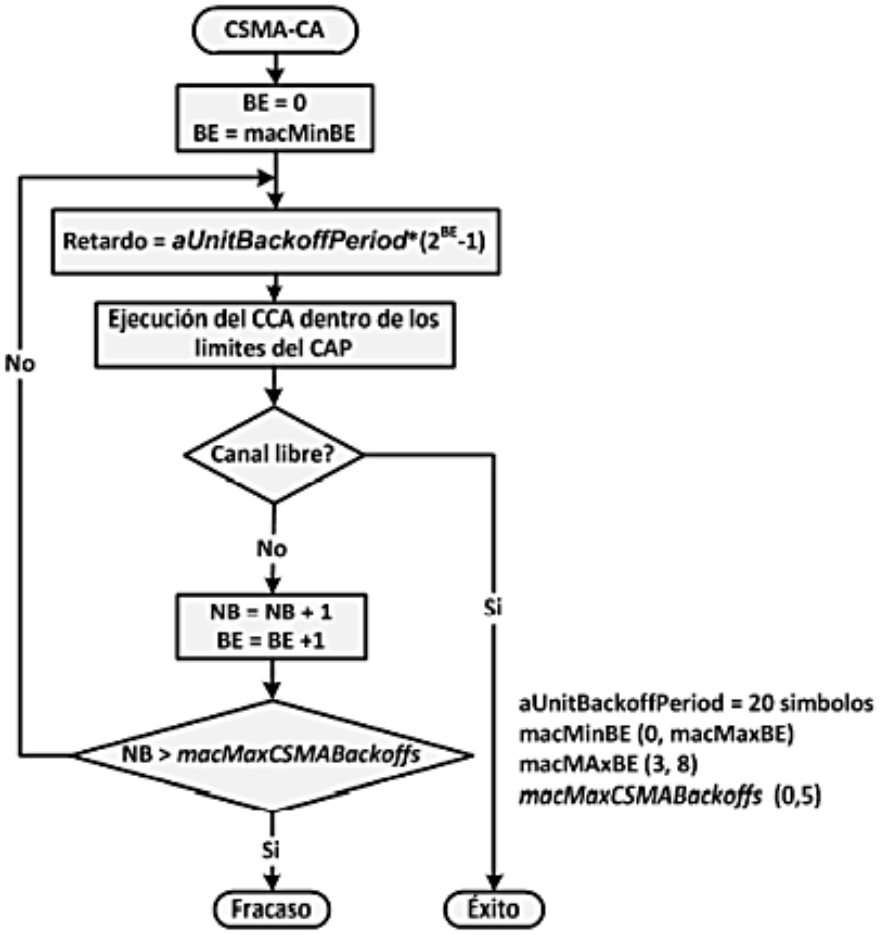

Figura 1.17. Algoritmo de backoff en redes no beacon. [8]

1.3.6.2.3 Formas de transferencia de datos [13]

Podemos tener tres tipos de transferencia de datos en una red bajo el estándar IEEE 802.15.4 entre los cuales son:

- Dispositivo a Coordinador.

- Coordinador al dispositivo.
- Transferencias punto a punto.

1.3.6.2.3.1 Transferencia de datos hacia un coordinador [13]

Cuando vamos a transmitir de esta forma se tiene dos opciones, transmitir con el modo llamado modo ranurado o utilizar el modo no ranurado, para el modo ranurado es importante el uso de la supertrama para la sincronización de los nodos, si el canal está libre el dispositivo transmite los datos al coordinador, aquí el coordinador puede o no utilizar el ACK o trama de acuse recibo.

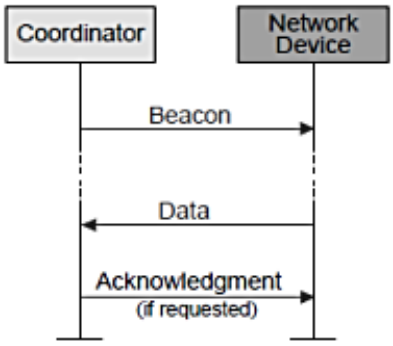

Figura 1.18. Comunicación hacia el coordinador en una red beacon. [14]

También transmitir con el mecanismo CSMA-CA no ranurado, en este modo sí el dispositivo desea transmitir el nodo envía la información de la misma manera al coordinador, puede emplear el ACK para confirmar la recepción de los datos.

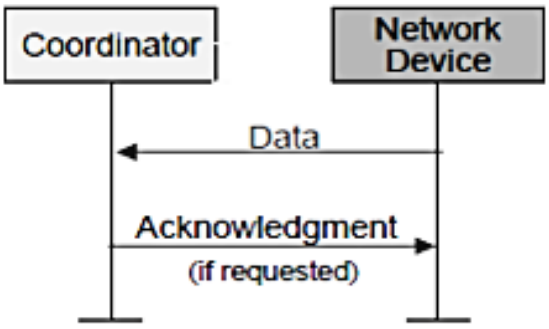

Figura 1.19. Comunicación hacia el coordinador en una red no beacon. [14]

1.3.6.2.3.2 Transferencia de datos desde un coordinador [13]

Si el coordinador se encuentra en el modo CSMA-CA ranurado el coordinador informa que tiene datos en cola por enviar a los nodos de la red que se encuentran escuchando y censando el canal periódicamente, si alguno de los nodos reconoce que los datos son para ellos, envía un comando MAC solicitando que se transmita la trama de datos utilizando CSMA-CA ranurado.

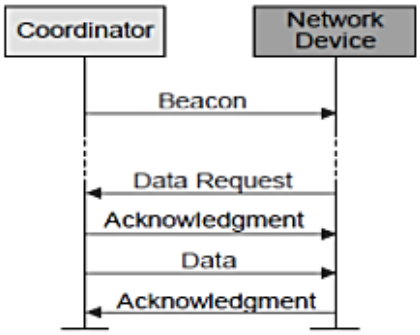

Figura 1.20. Comunicación desde un coordinador en una red beacon [14]

Si el coordinador emplea el mecanismo CSMA-CA no ranurado, el coordinador guarda los datos hasta que un dispositivo que pertenezca a la misma red envie una trama MAC solicitando que le transfieran los datos, el coordinador envía al dispositivo una trama ACK confirmando la aceptación de la solicitud del dispositivo. Una vez que se han enviado los datos desde el coordinador hacia el dispositivo, de igual manera se puede enviar una trama ACK.

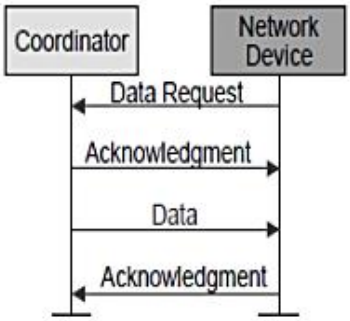

Figura 1.21. Comunicación desde un coordinador en una red no beacon [14]

1.3.6.2.3.3 Transferencia de datos de una conexión punto a punto [8]

Los dispositivos que se encuentran dentro de una red punto a punto, pueden comunicarse con todos los dispositivos que se encuentren dentro del rango de cobertura. Estos

dispositivos para obtener una transferencia de datos confiable deben mantener constantemente el intercambio de información, o deben sincronizarse entre ellos o pueden transmitir en CSMA-CA no ranurado.

1.3.6.2.4 Espacio de tiempo entre tramas [8] [11]

El (IFS) espacio entre tramas, es un tiempo de espera que el dispositivo transmisor realiza, así el receptor tiene el tiempo requerido para analizar la trama antes de enviar la próxima trama. Hay que analizar la diferencia en los espacios entre las tramas cuando usamos ACK y cuando no usamos ACK,

Al variar el tamaño del MPDU se puede tener un SIFS, o short Interframe Spacing, cuando las tramas de datos son cortos y un LIFS o Long Interframe Spacing para tramas de datos grandes.

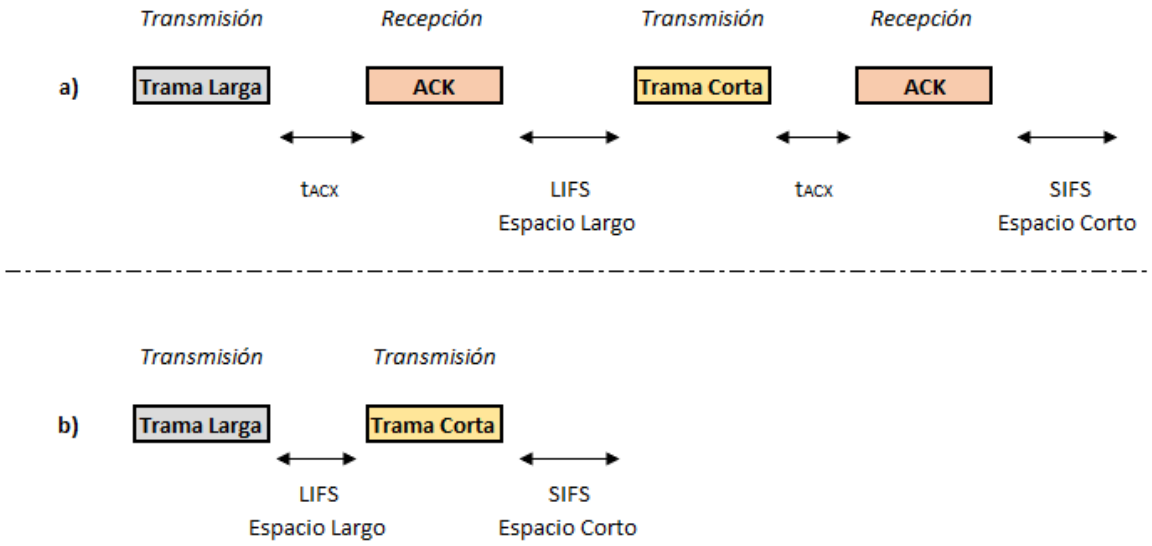

Figura 1.22. Espacio entre tramas: (a) transmisión con ACK, (b) transmisión sin ACK. [8]

1.3.6.2.5 Estructura de las tramas MAC [6]

La estructura general de una trama MAC para el estandar IEEE 802.15.4 que tiene una longitud máxima de 127 bytes, y esta conformada por tres partes importantes que son el encabezado MAC MHR, la unidad de datos de servicio MAC, y el fin de trama o MFR.

El encabezado MAC o cabecera MAC dentro de su estructura contiene los FCF, o Frame Control Field, el número de trama, la información concerniente a las direcciones, y un encabezado auxiliar que nos ayuda con la información acerca de la seguridad.

La unidad de datos de servicio MAC o MAC Service Data Unit, este campo tiene la carga útil, es de longitud variable y lleva información dependiendo del tipo de trama.

El fin de la trama o MFR, esta contiene la secuencia de chequeo de trama FCS, que nos ayuda con el control de la trama.

El estandar 802.15.4 tiene 4 tipos de tramas estas pueden ser.

1.3.6.2.5.1 Trama de datos [13]

Esta trama contiene los datos que se envian a la subcapa MAC por medio de la MSDU (MAC Service Data Unit) que vienen de las capas superiores.

A continuación, podemos observar la composición de la trama de datos.

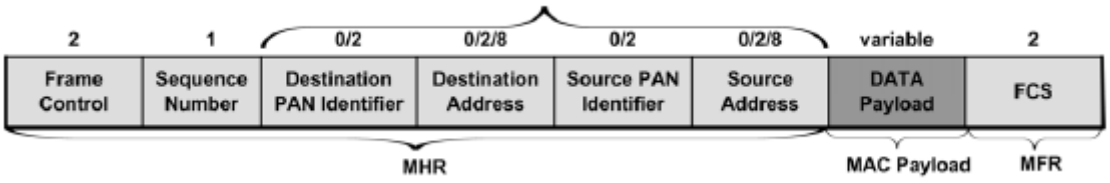

Figura 1.23. Formato de trama de datos [6]

1.3.6.2.5.2 Trama de acuse de recibo ACK [13]

Esta trama se encarga de dar la confirmación de la recepción de una trama enviada; su longitud es de 5 bytes, no contiene carga útil e información de direccionamiento.

A continuación, se ilustra la composición de una trama ACK.

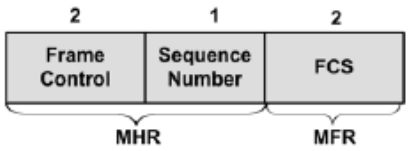

Figura 1.24. Trama de acuse de recibo ACK [6]

1.3.6.2.5.3 Trama de Comandos

Esta conformado por dos campos, el primero de ellos es identificadar la trama de comando, su longitud es de 1 byte y tiene el tipo de comando a emplear, el segundo de los dos campos de esta trama es el de carga útil de comando, este campo puede tener una longitud variable y contiene la información del comando.

A continuación, podemos detallar gráficamente la estructura de la trama de comandos.

Bytes	2	1	6-8-10-12-14-20	0-5-6-10-14	1	Variable (n)	2
	Control de Trama	Número de Secuencia	Información de direcciones	Encabezado de seguridad auxiliar	Tipo de comando	Carga útil	FCS
	MHR				MSDU		MFR

Figura 1.25. Trama de comandos [9]

1.3.6.2.5.4 Trama Beacon

Esta trama es la encargada de sincronizar los dispositivos que se encuentren en la red, va a ser enviada por el coordinador de la PAN para informar a los dispositivos de la red, acerca de la existencia de datos pendientes para transmitir. Así lo vamos a detallar en la siguinte figura para indicar los campos que va a tener esta trama beacon.

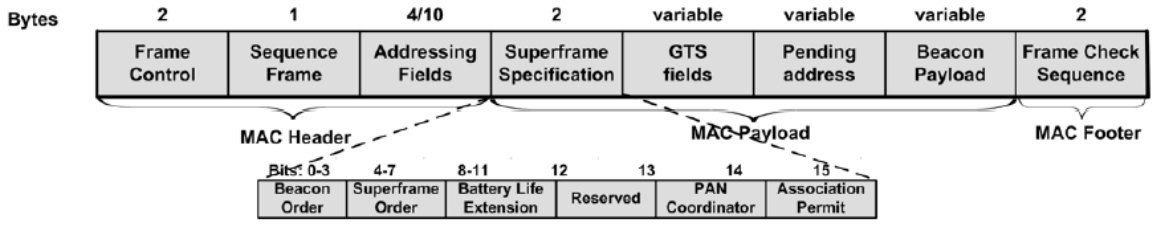

Figura 1.26. Formato de trama Beacon [6]

1.3.6.2.6 Direccionamiento en IEEE802.15.4

En el estandar IEEE802.15.4., se maneja un direccionamiento diverso, en ocasiones no podemos encontrar explicitamente la dirección puesto que se puede encontrar implicitamente, esto depende del tipo de mensaje, es más se puede emplear una versión reducida para optimizar espacio en el encabezado (MHR) de la trama MAC.

Para esto se puede emplear dos tipos de direccionamiento para IEEE802.15.4 como son: el direccionamiento extendido, y el direccionamiento corto, (extended addressing and short addressing), dentro del encabezado de la trama MAC Header (MHR) se muestra el tipo de direccionamiento, que se está ocupando en la dirección de origen y en la dirección de destino.

1.3.6.2.6.1 Dirección Corta

El tamaño de una dirección corta es de 16 bits, esta dirección hace que cada nodo o dispositivo de la red pueda identificarse dentro de la red PAN.

1.3.6.2.6.2 Dirección Extendida

El tamaño de esta dirección extendida es 64 bits y es similar a un número de serie que tiene cada dispositivo para identificarse de los demas dispositivos, es decir es único..

1.3.6.2.6.3 Identificador de red (PAN ID)

Este identificador nos permite ver desde que red proviene un mensaje y a donde va el mismo, para que no existan repeticiones el uso de una bandera la intra-pan, que es la encargada de identificar el tráfico de mensajes dentro de una misma red y solo ocupa la PAN ID de destino..

1.3.6.2.6.4 Broadcast

Se usa cuando queremos comunicarnos con todos los dispositivos de la red y que reciban una misma información todos, el broadcast usa direccionamiento corto y la dirección de destino que ocupa es 0xFFFF.

1.3.7 Protocolo 6LowPAN [15]

Existen muchos protocolos basados en el estándar 802.15.4 que no son compatibles con TCP/IP, por esta razón no permiten realizar una comunicación con dispositivos que usen esta tecnología, en esta coyuntura, es desarrollado 6LowPAN, IPv6 sobre redes de área personal inalámbricas de baja potencia, cuyo objetivo es hacer posible la comunicación de dispositivos como los nodos con otros dispositivos IP, ya que define paquetes IPv6 sobre la norma IEEE 802.15.4, y producto de esto garantizar la conexión extremo a extremo.

A continuación, se detallarán varias características:

- Presenta topologías malla, estrella o árbol.
- Gran número de dispositivos en la red.
- Puede transmitir hasta 250 kbps
- Basado en el estándar 802.15.4 ocupa su mismo direccionamiento MAC
- Ahorro de energía con su modo (modo dormido), de esta manera presenta periodos largos de letargos.
- Es un estándar abierto a varias aplicaciones.

- Posee una compresión de cabecera eficiente.

1.3.8 ARQUITECTURA DE WSN CON EL PROTOCOLO 6LowPAN [6] [16]

Todo sistema de comunicación necesita unas reglas para poder funcionar correctamente, de esta manera nos permite dar forma a los datos y controlar el intercambio de estos, para esto se usan los stacks de protocolos los cuales separan en distintas capas los protocolos de comunicación que dispone el sistema para ponerse en comunicación con otros dispositivos.

Estas capas están divididas de tal manera que comienzan con las de más bajo nivel a través de una interacción física mediante el hardware, mientras vamos avanzando las capas son más específicas e incorporan mejores características que la capa anterior. Por esta razón se va a realizar un análisis de lo que es la estructura de 6LowPAN definido por cinco capas del modelo ISO/OSI que a continuación detallaremos.

- Capa Física (PHY)
- Capa de Enlace (MAC)
- Capa de Red (IP)
- Capa de Transporte
- Capa de Aplicación

El stack de protocolos IPV6 con 6LowPAN a comparación con el Stack de protocolos IP y con sus correspondientes capas dentro de la arquitectura TCP/IP, podemos notar algunas diferencias, la primera es que 6LowPAN solo soporta IPV6 mediante una capa de adaptación llamada LowPAN, la cual fue definida para usarse sobre IEEE 802.15.4.

En la capa de transporte ya no se ocupa TCP, puesto que degrada el desempeño y la eficiencia de la red por razones de complejidad, solamente se usan UDP e ICMP. Su capa de red implementa IPV6 con LowPAN, que básicamente es una compresión de 40 bytes a 4 bytes, lo que permite eliminar campos que se repiten, y finalmente en su capa de aplicación tiene IPV6 y radio 802.15.4 para que trabajen de forma conjunta e implementar seguridades los cuales se lo hace mediante cifrado de paquetes AES-128.

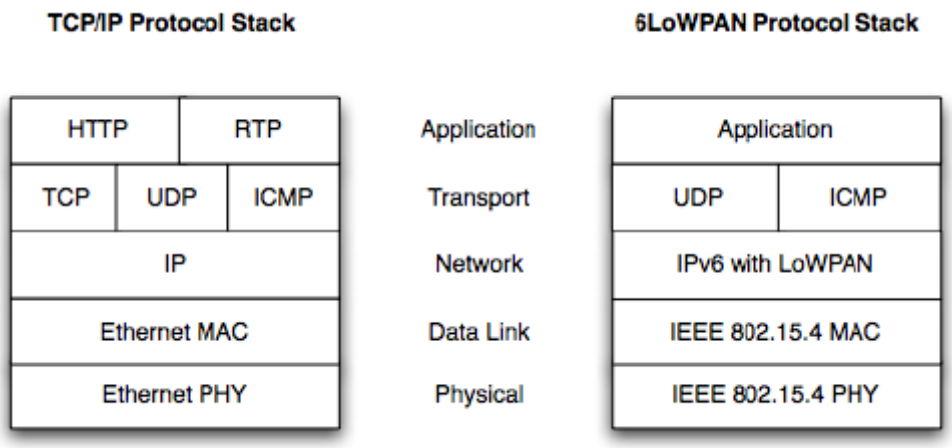

Figura 1.27. comparación Arquitectura 6LowPAN e IP [15]

Esta arquitectura permite el encapsulamiento de paquetes 6LowPAN sobre la trama de enlace de datos IEEE 802.15.4 con un tamaño máximo del paquete de 127 bytes.

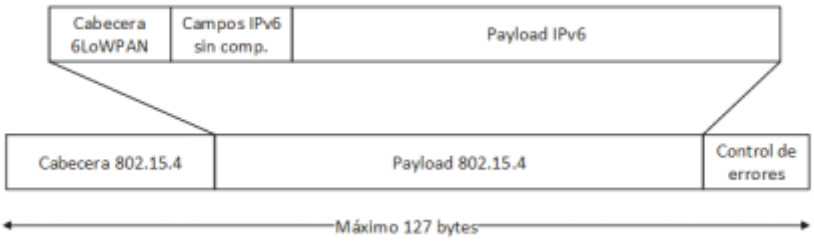

Figura 1.28. Formato de paquetes 6LowPAN [17]

En el nodo Gateway se tiene los dos stack, tanto 6LowPAN como de IP, para este tipo de redes se han definido mecanismos para la conexión a nivel de enlace ocupando el estándar IEEE 802.15.4.

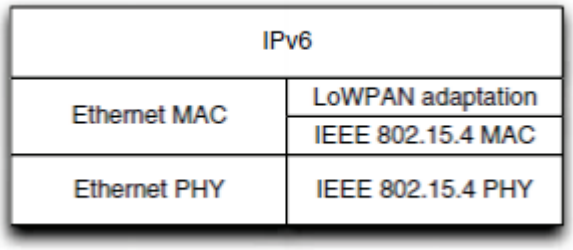

Figura 1.29. Protocolos en el nodo Gateway [15]

6LowPAN añade una capa conocida como capa de adaptación que se introduce entre la capa de enlace y la capa de red como se muestra en la figura 1.27 y figura 1.29, esta capa

nos ayuda a resolver una serie de problemas que enfrenta el estándar dentro de los cuales tenemos:

Facilita el enrutamiento de datos, la capa de adaptación decide a que nodo se tiene que dirigir los datos a nivel de capa de enlace para seguir hasta el destino final marcado por la dirección IP, es decir que permite un avance a nivel de capa de enlace y de red.

La capa de adaptación incorpora Stateless auto-configuration que genera de forma automática la dirección IP a cada miembro de una red 6LowPAN, también ayuda a resolver el problema de duplicate address detection (DAD)

Esta capa permite enviar distintos tipos de paquetes, además de distinguir varios tipos de encapsulamiento, puede adaptarse a la MTU exigida por IP 1280 bytes solucionando el conflicto que se tiene en IEEE 802.15.4 que solamente tiene paquetes de 127 bytes, así como la fragmentación, reconstrucción, y ensamblaje de paquetes, la capa de adaptación incluye una compresión de los headers, evitando enviar información repetida y reduciendo el tamaño de los headers y así la redundancia.

La capa de adaptación 6LowPAN tiene tres elementos que son la fragmentación, comprensión y el direccionamiento.

1.3.9 FRAGMENTACIÓN DE LOS DATOS [18] [17]

Los paquetes de IPV6 pueden tener distintos tamaños, los más pequeños solo contienen 40 bytes correspondientes a las cabeceras, es decir que no tienen carga por ende su utilidad es mínima. Por otro lado, el tamaño máximo puede llegar a tener un paquete IPV6 es de 65575 bytes en total con una carga de 65535 (2^16-1) bytes, ahora muy pocas redes pueden transmitir paquetes tan grandes sin tener pérdidas, puesto que la mayoría tiene definidos MTU menores al valor máximo de bytes definidos por el protocolo. Todas aquellas redes que tengan un MTU inferior al tamaño del paquete que se quiere enviar utilizan fragmentación.

El estándar IPV6 dispone de una MTU de 1280 bytes mientras que IEEE802.15.4 tiene una MTU (unidad máxima de transferencia) de 127 bytes para el caso de 6LowPAN que su objetivo era aunar los dos estándares, se decidió que el MTU sea de 1280 bytes, se recomienda que sea de 1500 bytes que es el que maneja ethernet puesto que la mayoría de las redes de internet está basada en este protocolo y es probable que exista una transferencia de datos entre este tipo de redes.

Route Over: La fragmentación y la reconstrucción se puede realizar en cada salto, puesto que si no llega la información completa al nodo este devuelve todos los fragmentos al nodo anterior, para que se los envíe correctamente y así se reenvíen los fragmentos correctamente y armar el mensaje original, esto aumenta el tiempo de transmisión a comparación que la otra configuración mesh under, aunque el beneficio es que si se extravía un paquete solo se lo debe notificar como en la figura 1.30.

Mesh-under: En este tipo de enrutamiento al contrario del modo anterior, la fragmentación se realiza solamente en el nodo fuente desde donde sale el paquete y la reconstrucción se la realiza en el nodo destino, y los nodos intermedios solamente envían partes sin reconstruirlos así el tiempo de transmisión es más rápida en cada nodo, sin embargo, si se pierde un fragmento de paquete debe ser enviado nuevamente desde el nodo transmisor como se muestra en la figura 1.30

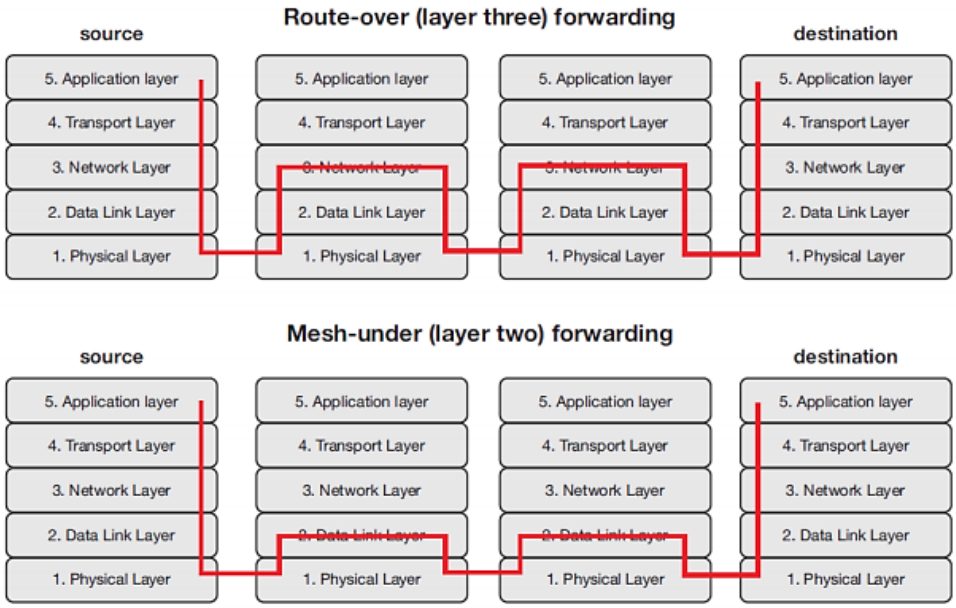

Figura 1.30. Tipos de enrutamiento de paquetes en 6LowPAN [17]

De acuerdo con el estándar 6LowPAN establece que los paquetes solamente pueden ser fragmentados y recompuestos en el nodo fuente, y el nodo destino por esa razón se ocupa el modo mesh under. El encargado de decidir si el paquete se fragmenta o no, es el nodo emisor puesto que no está permitida la fragmentación en nodos intermedios, esta decisión se toma en base de la ruta que va a tomar el paquete

Para esto el estándar 6LowPAN tiene implementado el PMTUD que es un mecanismo de IPV6, nos indica el mínimo valor de MTU entre todos los nodos que participan de la transmisión así busca la ruta que evite la fragmentación.

1.3.10 FORMATO DE LA CABECERA [15]

A pesar de que el payload sea pequeño como para que no necesite realizar la fragmentación y de esta manera tener una reducción de consumo energético; la utilización limitada de los canales IEEE 802.15.4 causado por los encabezamientos de IPV6, era una razón por la cual antes se evitaba implementar tecnología IP en los sensores. Ahora para poder utilizarla es importante entender que la compresión sea fundamental para 6LowPAN, para poder resolver este inconveniente el estándar plantea dos soluciones que son la compresión sin estado y la compresión basada en el contexto. La cabecera de LowPAN se compone de varios bytes, el primero toma un valor asignado que identifica el tipo de encabezado, seguido por un byte de compresión de cabecera IPV6 que indica los campos que se van a comprimir y luego cualquier campo IPV6 en línea.

La compresión stateless es usada en redes locales ya que el entorno es el mismo para esto realiza una simplificación de los header esta consiste en no informar al nodo que va a recibir, acerca del estado o contexto en el cual se produce la comunicación, ya que el entorno será el mismo, no se necesita definirlo al momento de enviar los paquetes. La compresión IPV6 incluye el propio encabezado y puede incluir la compresión del protocolo UDP, siempre que se encuentre incluido en IPV6. El HC1 realiza las simplificaciones en la cabecera de IPv6 mientras que HC2 aporta con la compresión de la cabecera UDP.

Cuando realizamos la compresión de la cabecera IPv6 la parte que esta comprimida tiene banderas que indican las opciones de compresión HC1 y la parte de la cabecera que no ha sido comprimida.

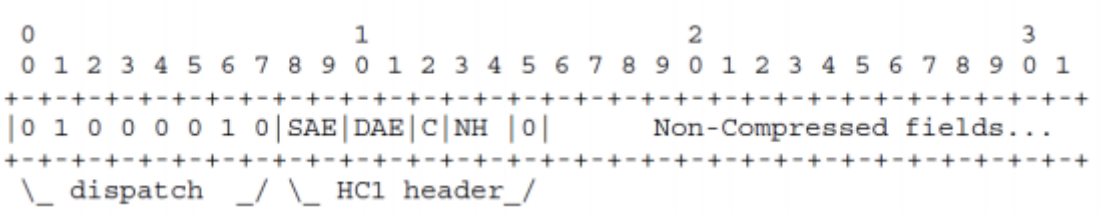

Figura 1.31. Cabecera de compresión HC1 de un paquete IPv6 [17]

Ahora al hablar de la cabecera de opciones UDP, su presencia va a depender del bit menos significativo de la cabecera HC1 si se encuentra un 1 se incluye el HC2 en la cabecera, de

no estar el 1 no se lo incluye. A pesar de que este bit habilite la presencia de la cabecera de opciones UDP solo es útil siempre y cuando el valor del campo NH sea de 17 es decir si el siguiente header es el de udp como se indica en la siguiente figura, así podemos comprender de mejor manera como trabajan estas compresiones de cabecera simultaneas.

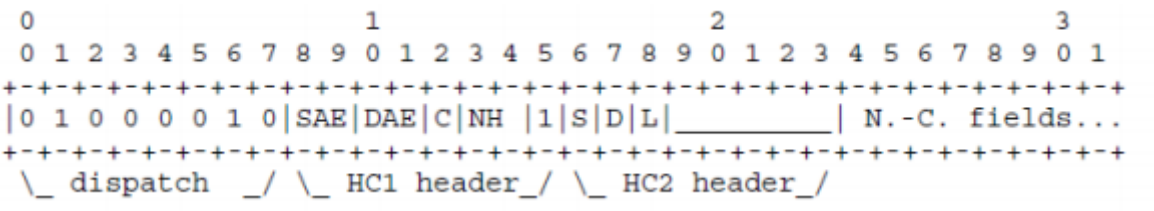

Figura 1.32. Cabecera de compresión con HC2 habilitado de un paquete IPv6 [17]

Si tenemos compresión de HC2 aparecen 3 bits (S, D, L) los cuales son los encargados de la compresión del puerto emisor, del receptor, y también de la longitud. Si hablamos de la longitud del paquete esta es inferida según el payload ya que el tamaño de los otros parámetros que interfieren ya es conocido, de esta forma el valor de la longitud se hace fácil de comprimir.

Los campos que no se comprimen de UDP seguirán a continuación de los campos de IPv6 no comprimidos siguiendo el mismo orden de UDP de manera que esté puerto del nodo de origen, puerto del nodo destino, longitud del paquete y su respectivo checksum

La compresión context-based está orientada a redes globales de comunicación donde cada nodo incluye en los paquetes que envía, su contexto actualizado, de esta manera el factor más importante es el de la sincronización de contextos, entre compresor y descompresor para que la cabecera pueda ser procesada de manera correcta. Por este motivo el nodo emisor del paquete (compresor), debe tener los suficientes indicios que el nodo receptor va a poder descomprimir el paquete para que de esta manera tenga la información correcta, tomando en cuenta que se intenta no enviar valores en ranuras que recientemente se hayan modificado ya que existe el riesgo de que el valor modificado no se encuentre en todos los nodos, ya que existe una dependencia de alcance y de presencia de nodos en reposo, otra opción es dejar los valores de esa ranura durante un tiempo antes de introducir nuevos para no generar errores.

De la misma forma que ocurriría en el caso de compresión sin estado se incluye la posibilidad de simplificar los paquetes UDP desde un encabezado IPv6, así podemos dividir el header en IPHC que es la cabecera de simplificación IPv6 y NHC que es la cabecera de

simplificación de UDP. En el IPHC se coloca al inicio del paquete y tiene una serie de banderas y bits de selección que muestran el grado de simplificación de la cabecera IPv6

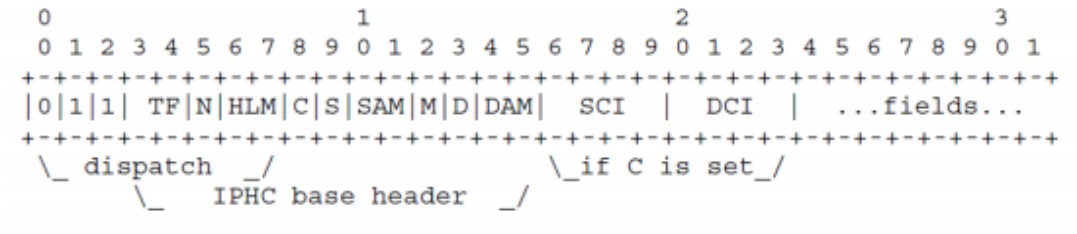

Figura 1.33. Formato de cabecera IPHC en un paquete IPv6 [17]

Podemos ver un ejemplo de compresión 6LowPAN en la que en la figura 1.34 en el paquete superior se tiene un valor de 1 byte de despacho se incluye para indicar IPV6 completo sobre IEEE802.15.4. Al contrario de la figura 1.35, donde se muestra un ejemplo de 6LowPAN/UDP en su forma más simple con un valor de despacho y de compresión de cabecera IPV6 comprimidas seguidas de un byte de cabecera de UDP con compresión de puertos origen y destino y la suma de comprobación UDP (4 bytes)

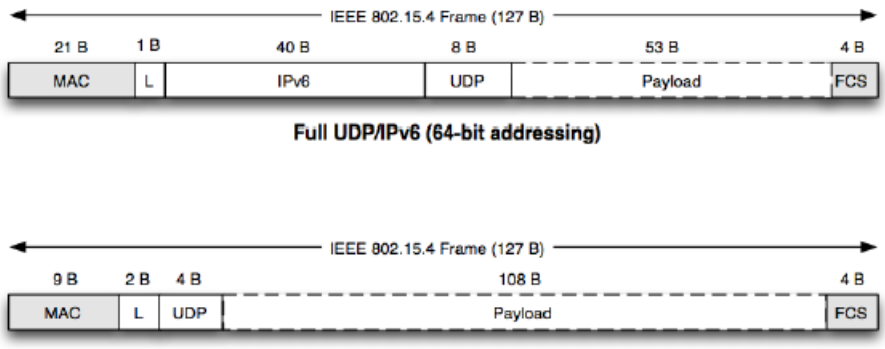

Figura 1.34. Ejemplo de compresión de cabecera 6 LowPAN [15]

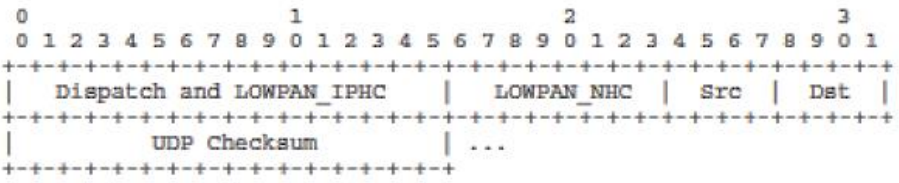

Figura 1.35. Compresión de cabeceras 6LowPAN/UDP [15]

Al realizar una comparación de las cabeceras podemos notar que la compresión de 6 LowPAN será de solo 6 bytes a comparación de la otra cabecera que es de 48 bytes de longitud como se muestra en la figura 1.36. Por lo que se considera que en el peor de los

casos IEEE802.15.4 tiene solamente 72 bytes de carga útil disponible después de las cabeceras de la capa de enlace y así podemos concluir que la compresión es importante.

Figura 1.36. Cabeceras estándar IPv6/UDP (48 bytes)[15]

1.3.11 DIRECCIONAMIENTO [15]

El estándar 6LowPAN tiene un mecanismo de generación automática. Este mecanismo es conocido como stateless auto-configuration y asigna las direcciones a los nuevos nodos a partir de la información estática que se encuentra en los nodos y routers que están a cargo de gestionar la red LOWPAN. Así se elimina la asignación manual de direcciones, o el uso de protocolos de asignación automática, la creación de direcciones varía, esto dependiendo de si la red es local donde no ocupe routers o es una red global como puede ser el internet.

Para direcciones locales se utiliza EUI-64, la cual es una asignación dada por el fabricante de los nodos, asociada a la dirección MAC de la capa de enlace, es decir que los fabricantes aseguran que sus direcciones EUI-64 son únicas en sus productos, de esta forma sabemos que cada dirección generada por ellos es única, y diferente a las demás.

Para asignación de nodos de redes globales, la dirección del nodo también debe tener una dirección única por esta razón ya no se utiliza solamente la dirección del nodo física sino también se necesita la información global aportada por el router de frontera. El proceso para incorporar un dispositivo se gestiona por medio de enviar un mensaje solicitando información al router, el cual responde con otro mensaje de información donde se da el prefijo de la red global a la que se va a conectar, que junto al dirección EUI-64 crea la dirección global.

Tenemos que considerar además, la posibilidad de una asignación de dirección manual de ocurrir esto al momento de tener una red global o local, tendríamos nodos repetidos y eso no se puede permitir, por esa razón se utiliza el DAD que es someter a los nodos a un proceso de detección de direcciones duplicadas en la red, este es un proceso obligatorio para todos los nodos que se intentan acoplar a una red, este proceso es automático y a partir de eso podríamos detallarlo como la generación de direcciones IPv6 únicas.

2 METODOLOGÍA

2.1 TOPOLOGÍAS LINEALES CON WSN

2.1.1 CONCEPTOS Y CARACTERÍSTICAS DE TOPOLOGÍAS LINEALES

Se llaman redes de topología lineal a las redes de sensores que se colocan en línea recta, son consideradas como subdivisión de una red tipo árbol de una sola rama, estas redes constan varios nodos sensores inalámbricos y un gateway el cual permite la transmisión de información fuera de las redes de sensores inalámbricos, además de que ayuda con la configuración e iniciación de la red de sensores como se indica en la figura 1.1 del capítulo 1. Para que exista un buen funcionamiento de una red de sensores que trabaje en topología lineal donde se encuentren cientos de nodos hay que considerar ciertos aspectos como los siguientes:

Que en este tipo de topología existe un incremento en la eficiencia al momento de realizar el enrutamiento. Al disminuir los tiempos de operación de los nodos y se reduce el consumo de energía que existe en la red. La frecuencia de trabajo a la que vamos a trabajar es 2,4 GHz en los sensores.

Otro parámetro positivo que se obtiene al trabajar con redes en topología lineal es la escalabilidad que proporciona una mejor capacidad para trabajar adecuadamente cuando va creciendo o tiene que adaptarse a las diferentes demandas que tiene la red.

La topología lineal tiene el siguiente funcionamiento, partiendo de la premisa de que cada nodo tiene una información que enviar, así que el nodo genera un paquete de información que va a ser enviado a los nodos intermedios de la topología hasta llegar al gateway, por lo que podemos entender que todos los nodos procesan información y esto influye de manera directa en la duración de las baterías.

Uno de los principales inconvenientes que tenemos con estas redes de topología lineal es el flujo de datos que existen en ellas, ya que al tener que pasar la información por todos los nodos intermedios se necesita que tengan un mayor procesamiento de la información y requiere que la red tenga una mayor confiabilidad, puesto que si un nodo falla se perderá la conectividad entre algunos nodos intermedios y de esta manera se perdería la completa funcionalidad de la red. Es por eso por lo que en esta topología se propone que los nodos sensores intermedios se encuentren a distancia media del alcance, los nodos que vamos a utilizar tienen un alcance de unos 60 a 70 m en ese caso se escojeria una distancia menor o igual a la mitad del alcance que tiene el sensor, así si por algún motivo un nodo se

inhabilite por cualquier razón, las redes busquen otra ruta siguiendo la topología lineal para poder cumplir con la transmisión de datos.

2.2 DESCRIPCIÓN DEL FUNCIONAMIENTO DEL PROTOTIPO

El prototipo funciona de la siguiente manera, al encender cada uno de los nodos separados una distancia que está relacionada con la zona de cobertura como se lo ha mencionado anteriormente. Los nodos se enlazan entre si utilizando los protocolos necesarios para ir colocando un nodo a continuación de otro siguiendo una topología lineal.

Lo que hace cada uno de los nodos es enlazarse con sus vecinos a espera que se envíen los datos reconociendo cuál es su nodo vecino. Si por algún motivo el nodo vecino no se encuentra disponible, el nodo busca una nueva ruta con el siguiente nodo cercano que se encuentra en la ruta según el protocolo y se envía la información al nodo para así mantener la comunicación entre cada uno de los nodos y de esta manera mejorar la confiabilidad de la red.

2.2.1 ASOCIACIÓN DE LOS NODOS EN UNA RED DE TOPOLOGÍA LINEAL

Al momento de encender los nodos que van a trabajar en nuestra WSN, los nodos primero se configuran a la menor potencia de trabajo para que las distancias sean las mínimas y se ubican en las posiciones requeridas que mencionamos anteriormente es decir a diferentes distancias un nodo de otro, tomando en cuenta que dos nodos deben estar dentro del mismo radio de cobertura de un nodo sensor para que de esta manera si algún nodo llegará a fallar, el sistema se reconfigure y no se caiga la red que hemos planteado.

Para esto la asignación de la numeración de los nodos se hace, censando los niveles de potencia que existen entre unos y otros, así se configura de acuerdo con la potencia que se perciba de un nodo sensor con respecto de otro, de esta manera el sensor que al evaluar la potencia de los nodos contiguos se enlazará con el nodo que esté más cerca de él porque su potencia será mayor a la potencia de los otros que el nodo sensor puede percibir. Es decir que, al tener la misma potencia, pero al estar alejados, al censar el canal los niveles de potencia de cada uno de los sensores serán menor, de esta manera en la WSN en base a las potencias se van enumerando los sensores desde el Gateway hasta el sensor más lejano. Además, se colocó en los nodos previamente configurados un limitante del número de hijos para evitar que exista una asignación diferente de dos nodos por fluctuaciones de potencia en los nodos por consumo de batería.

De esta manera podemos tener escalabilidad es decir podemos ir agregando más sensores de requerirlo recordando que para estas pruebas de una red de WSN ayudan a ubicar a los nodos más cercanos.

El proceso de asignación de los nodos puede verse afectado por las siguientes causas:

2.2.2 NODO FALLIDO

La caída de un nodo puede tener varios motivos cuando hablamos de sensores, uno de los principales es el agotamiento de la batería o que exista un eventual daño físico especialmente cuando estos nodos están a la intemperie. En esos casos el nodo encargado de transmitir una información dejaría de trabajar así el nodo anterior intentaría enviar la información sin tener respuesta de confirmación o ACK, al estar deshabilitado este nodo sensor no recibirá, procesará ni trasmitirá información, y al ser una red de topología lineal no podría ser reenviada al siguiente nodo de la secuencia, al suceder eso el nodo anterior procede a realizar varios intentos para llegar al nodo destino, que en base a la configuración que se disponga en el nodo sensor pueden ser de 3 hasta 10 veces, en nuestro caso 3 intentos de alcanzar el nodo destino son suficientes para declararlo como nodo caído, en ese caso el protocolo busca una ruta para enviar el mensaje a su destino.

Al mencionar una nueva ruta entra en funcionamiento la red de topología lineal, como lo habíamos mencionado con anterioridad al tener dos nodos dentro del rango de cobertura del nodo emisor, si el nodo caído es el nodo más cercano existirá un nodo un poco más alejado, pero dentro de la misma zona de cobertura que pueda recibir la información y alcanzar su destino, de esta manera la confiabilidad de la red no disminuye, solamente cambia de ruta, lo declara como nodo caído y comienza a enviar la información por la nueva ruta.

Al ser el protocolo el que realiza la búsqueda de la nueva ruta nos facilita la utilización de este tipo de redes WSN

2.2.3 FALLA DE ENLACE

Similar al nodo caído ,la falla de enlace en una red de topología lineal sucede cuando las tramas que se transmitieron no fueron receptadas por el nodo destino, y al no recibirlas no se tiene el acuse de recibo, este tipo de fallas en el enlace pueden suceder por algunos motivos entre ellos puede ser la interferencia, por esta razón cada nodo en su configuración también puede determinar el número de veces que se vuelve a enviar la información si es que no se tiene respuesta del nodo destino, similar al nodo caído en los nodos vamos a tener tres intentos antes de declararlo una falla de enlace este número de veces igualmente

puede ser hasta 10 veces, hay que destacar que a veces la interferencia puede ser al primer o segundo envío y luego de varios intentos podemos llegar al nodo destino.

2.2.4 BÚSQUEDA DE NUEVAS RUTAS

Sea por falla de enlace o por nodo caído, la búsqueda de una nueva ruta para alcanzar el destino se convierte en algo fundamental en una red WSN. Sin embargo, hay que recordar que tanto la falla de enlace o el daño en el nodo puede ser momentáneo y luego de ello puede el nodo a reestablecerse por esta razón el cambio de ruta puede ser considerado como temporal, ya que siempre el protocolo estará censando los nodos que se encuentren en la zona de cobertura de esta manera al volverlo a encontrar o revisando que ya se encuentre el nodo la ruta será nuevamente reconfigurada para que todos los nodos trabajen.

Considerando que el nodo puede estar caído o puede existir fallo de enlace de manera temporal ya se por las baterías o por un apagón momentáneo y las interferencias, pero dejando abierta la posibilidad de que le nodo no está dañado la red WSN se enlaza con su copia de seguridad dejando claro que no es de manera permanente puesto que si el nodo se reestableciera vuelve a su configuración de la WSN inicial.

2.3 HERRAMIENTAS PARA IMPLEMENTAR EL PROTOTIPO

Para realizar las pruebas, como primer punto hay que implementar las redes WSN, una vez teniendo claro el funcionamiento de éstas, procedemos a ver los requerimientos que va a necesitar la red, para esto se tuvo que realizar la instalación de software, el cual nos permita trabajar tanto con los nodos de IBM Mote Runner y de MEMSIC. Es decir, debemos cumplir con ciertos pasos y requerimientos fundamentales que necesitan los sensores, esto para generar un correcto funcionamiento.

A continuación, se va a detallar de manera breve lo que se necesita para su uso, así como su funcionamiento.

Comenzamos con los nodos de IBM, para poder implementar una red de sensores IBM Mote Runner se necesita lo siguiente:

- Kit de desarrollo Waspmote Pro V1.2
- MonoDevelop
- Ubuntu 12.04LTS
- Mote Runner SDK

- Mote Runner Compiler
- Mote Runner Shell
- MRV6
- AVRdude 5.11.1-1

2.3.1 Kit de desarrollo Waspmote Pro V1.2 [19]

Libelium en conjunto con IBM, nos brindan una plataforma de desarrollo IPv6 para redes de sensores e IoT (Internet de las cosas), a través de la combinación de IBM Mote Runner SDK que se puede entender como un paquete de herramientas que sirve para el desarrollo de software, y ayudan al programador a generar nuevas aplicaciones, adicionalmente se cuenta con un conjunto de nodos sensores Libelium Waspmote los cuales puede ser configurados de acuerdo a las necesidades y de esta manera nos brindan una herramienta para el desarrollo de aplicaciones relacionadas con la conectividad de 6LoWPAN para IPv6.En la figura 2.9 se presenta el kit.

2.3.2 MonoDevelop [20]

Es un IDE GNOME de utilización gratuita, el cual fue diseñado para la utilización de C# y otros lenguajes .NET, de esta manera es una plataforma extensible, donde cualquier herramienta puede construir o desarrollar todo tipo de programa.

2.3.3 Ubuntu 12.04LTS [21] [22]

A pesar de existir nuevas versiones de Linux se trabajó con esta versión, ya que, la librería que se utilizó solo se permitía ejecutar en esta versión y no existía compatibilidad con nuevas versiones.

Ubuntu es un sistema operativo basado en GNU/LINUX, más conocido como software libre, de esta manera al no requerir de licencias nos permite tener un fácil acceso. Una de las ventajas es que posee LTS (Long Term Support) significa que tiene soporte prolongado.

2.3.4 Mote Runner SDK [19]

IBM Mote Runner es el sistema operativo, utilizado en los sensores inalámbricos, además de ser un entorno de desarrollo de las WSN (redes de sensores inalámbricos).

Una de las dificultades que tiene el Mote Runner es que no es capaz de soportar todas las características de lenguaje. Se trata de una estrategia común para definir un subconjunto del lenguaje Java o C# adecuado para sistemas pequeños, se puede listar las restricciones y omisiones más importantes.

Restricciones Mote Runner:

- Tipos de datos float y double
- 64 bits o más números enteros
- Arreglos multidimensionales
- Reflexión
- Plantillas
- Hilos y primitivas de sincronización
- Arreglos booleanos
- Boxing
- API de tiempo de ejecución estándar
- Enumeraciones

Omisiones Mote Runner:

- Clases internas en C# (se admiten clases internas en Java).
- Delegados del multicast (solo soporta delegados unicast).
- Strings (excepto en situaciones especiales) y arreglos de Strings.
- Cualquier información de tipo en tiempo de ejecución como son getClass, nombres de clases excepto para los operadores de del lenguaje como instanceof, is y as.

Moter Runer define sus propias API del sistema, ya que las API de tiempo de ejecución utilizadas para la compatibilidad y las bibliotecas estándar de Java y C#, son demasiado extensas para que puedan ser contenidas en pequeños sistemas. Esto significa que la clase raíz Object viene sin ningún campo, ni cuenta con un método, puesto que, no hay reflexión y no hay información del tipo simbólica en el tiempo de ejecución.

Debido a que almacena bits en paquetes, requiere extenderse en el código las matrices booleanas no son compatibles, ya que, son matrices de bytes y al ser tan extensas resulta ineficiente utilizarlas.

2.3.5 Mote Runner Compiler [19]

Para poder compilar el programa que se va a cargar en los nodos, se utiliza el Compilador Mote Runner (MRC) que básicamente consiste en convertir el código fuente en un lenguaje ensamblador, para luego poderle cargar en la plataforma Mote Runner. El MRC dependiendo de lo que se necesite, organiza el llamado de varias herramientas para la ejecución, y así logra la transformación requerida.

2.3.6 Mote Runner Shell [19]

Es una interfaz de línea de comandos para el entorno de programación, y gestión basada en Javascript para Mote Runner

2.3.7 MRV6 [19]

Mote Runner dispone de un protocolo basado en 6LoWPAN, llamado MRv6, el cual puede ser instalado en cada uno de los nodos para poderles proporcionar IPv6.

Se puede entender que el MRv6 es una red basada en multisalto TDMA, es decir, que tiene acceso múltiple por división de tiempo, y que permite una comunicación fundamentada en IPv6 entre los nodos de la red inalámbrica y los hosts en Internet.

Se debe señalar que MRv6 no está implementado como un componente de Mote Runner, aunque se encuentra implementado en C#, empaquetado en asemblys y se instala de manera predeterminada con los mecanismos que tiene Mote Runner para realizarlo.

La red MRv6 es administrada totalmente por el nodo de borde, en los nodos de implementación también cuenta con una función de enrutamiento limitada para cumplir con la red de topología lineal, esta red es muy útil para aplicaciones de recolección de datos en donde los nodos sensores envían información periódicamente a un host remoto, cabe mencionar que con poca frecuencia necesitan actualizaciones o peticiones de hosts remotos.

Es por eso que, el nodo de borde tiene gran importancia, ya que él decide sobre las asociaciones, asigna los tiempos y determina las rutas de la red. En los nodos que pertenecen a la red MRv6 se pueden cargar aplicaciones cuyo funcionamiento es diferente al manejo de la red, puesto que esté conectada la red se puede consultar la red, comunicarse entre los nodos de la red, y recuperar datos de la misma.

El esquema que utiliza de paquetes forma parte de las especificaciones que tiene 6LowPAN que básicamente implica una compresión del estándar IPv6.

Además, MRv6 nos permite realizar un túnel para nodos conectados con el nodo de borde, que nos facilita la conversión de paquetes IPv6 a 6LowPAN, o viceversa, y los envía a los destinos requeridos, hay que acotar que el túnel es necesario solamente cuando la comunicación es con hosts internos, y que MRv6 soporta un Shell y un interfaz de programación que nos permite administrar, consultar, depurar, y comunicarse con una red inalámbrica, sea esta física o virtual.

2.3.8 AVRdude 5.11.1-1 [23] [24]

Utilizado para microcontroladores Atmel AVR, es un programa que permite la descarga de código y datos con la ventaja que son de software libre, por esta razón está disponible para Linux, este programa nos ayuda a la configuración del hardware de los nodos inalámbricos.

2.4 HERRAMIENTAS PARA IMPLEMENTAR EL PROTOTIPO CON MEMSIC

Luego de analizar los requerimientos necesarios para el desarrollo de las pruebas con los nodos sensores IBM, se va a detallar los requerimientos que tienen los sensores Memsic, como ya se había mencionado antes para poder realizar una comparación en dos esquemas diferentes como son el modo ranurado o TDMA de los nodos de IBM, se va a emplear otros nodos que trabajen en IPv6, de esta manera se escogió los nodos sensores Memsic.

A continuación, se va detalla los requerimientos que necesitan los sensores Memsic.

Para la implementación de la red WSN se partirá de 3 sensores IRIS, el mismo que mantendrá conexión con su nodo Gateway que estará enlazado a un computador, esto para generar la red, estos nodos estarán programados con la potencia mínima para cumplir con el esquema de poder medir a través del Sniffer los paquetes de datos de la red, lo que se utilizó para implementar fue lo siguiente:

- Kit de desarrollo MEMSIC que contiene los tres nodos sensores iris XM2110CB y el nodo Gateway MIB520CB
- Sistema Operativo Linux Ubuntu 14.04 LTS 32 bits
- Paquete TinyOS 2.1.2 para Linux
- Software Java-Eclipse Luna Sr2 4.4.2
- Liberia Yeti-2 para-Eclipse.

2.4.1 Kit de desarrollo MEMSIC

Esta tecnología o kit de desarrollo tiene múltiples aplicaciones, es ideal para sistemas que necesiten varios sensores a la vez, este kit cuenta con un nodo gateway y nodos sensores que nos permiten realizar varias aplicaciones.

Se escogió este kit para el análisis comparativo porque también trabaja con un sistema basado en IPv6; en los anexos se detallará lo que contiene en su totalidad el kit. En la figura 2.11 se presenta el nodo memsic.

2.4.2 Ubuntu 14.04 LTS para 32 bits

Ubuntu es un sistema operativo basado en GNU/LINUX y que se conoce como software libre, de esta manera al no requerir de licencias nos permite tener un fácil acceso.

Una de las ventajas es que posee LTS (Long Term Support) significa que tiene soporte prolongado, gracias a eso podemos tener más libertades al momento de trabajar con estos sensores en lo que tiene que ver a desarrollo. Adicional a eso los sensores Memsic necesitan el sistema operativo TinyOs, sistema que se acopla sin inconvenientes a esta versión de Linux y las versiones de 32 bits

2.4.3 TinyOS 2.1.2 para Linux

Es el sistema operativo con el que trabajan estos sensores inalámbricos, al ser de libre distribución facilita mucho para el desarrollo de proyectos de investigación, está diseñado para trabajar con las restricciones que manejan los microcontroladores de los sensores, posee bastante amplitud en el mercado, este está basado en una programación modular que hace que su manejo sea más sencillo.

Para poder ocupar todas sus herramientas es recomendable utilizar un gestor como lo es Java o C#.

2.4.4 Java Eclipse Luna

Plataforma de código abierto, al principio fue desarrollada por IBM, ahora es distribuida por Eclipse una fundación; la plataforma tiene varias herramientas de desarrollo integrado lo que hace que sea compatible con los sistemas operativos que comúnmente se utilizan en el mercado, dispone de un editor que nos permite corregir errores de forma inmediata, de esta manera facilitando el desarrollo de programas para luego ejecutarlos.

2.4.5 Librería Yeti-2

Es un repositorio Java, el mismo que cuenta con las herramientas necesarias para crear aplicaciones con el sistema operativo TinyOS; hay que mencionar que cuenta con su propio

lenguaje, el cual es el NesC, este repositorio nos permite crear aplicaciones con este lenguaje.

Para poder realizar la instalación de esta librería, hay que ir a su página oficial e instalarlo como nuevo repositorio en el entorno Java-Eclipse, mientras se esté realizando la instalación debemos cambiar la perspectiva del programa Java-Eclipse, debido a que al programa se lo va a utilizar con el sistema operativo TinyOS, cabe señalar que el entorno es similar al normal, con la diferencia que se tendrán las herramientas necesarias para trabajar con archivos de aplicaciones de formato NesC.

2.4.6 WIRESHARK [25]

Es un programa que permite analizar los paquetes de red que captura, y muestra a detalle el tipo de paquete que es. Antes este tipo de herramientas requerían de una licencia, pero en la actualidad este programa es considerado como uno de los principales analizadores de paquetes y es gratuito.

Los usos más frecuentes que le dan a este programa, es para implementar seguridades en las redes y también depurar la red.

A continuación, se va a detallar las características que ofrece Wireshark, y las ventajas que se pueden obtener al utilizarlo:

- Se encuentra disponible para Windows y UNIX
- Captura paquetes directamente de una interfaz
- Muestra de forma detallada los protocolos de la red
- Almacena los paquetes capturados
- Podemos filtrar paquetes o buscarlos
- Nos muestra la información de manera ordena, y amigable con el usuario.

El motivo por el cual se trabajó con Wireshark es debido a que el Sniffer al momento de su instalación nos redireccionó hacia Wireshark, es decir, que el Sniffer trabaja con el programa de Wireshark en conjunto.

Gracias a eso podemos aprovechar las siguientes aportaciones:

- Es compatible con Linux Ubuntu.
- Realiza captura de paquetes IPv6 que se encuentran en la red

- Entrega información detallada de los tipos de paquetes que se encuentran en la red, además de la dirección de origen y destino del paquete, como también el tiempo de llegada.
- La facilidad de manejar, buscar y filtrar información usando diferentes filtros.

Una vez detallado las herramientas que se va a necesitar para proceder con las pruebas, debemos detallar el programa a ejecutar, entendiendo que lo más importante es la medida de los tiempos que vamos a tener para luego poderlos plasmar en una fórmula.

La idea es enviar un dato por cada nodo con las redes planteadas, este dato va a ser reenviado hasta llegar al nodo final de la red, en ese afán es importante señalar que se establece dos esquemas, el uno en el que existe procesamiento, es decir se toma el dato en el nodo se lo extrae, se toma la información y se envía otro dato; y la otra que es solo el reenvío del dato sin procesarlo como se detalla anteriormente.

2.5 PROPUESTA DEL MODELO MATEMATICO PARA EL RETARDO POR PROCESAMIENTO

Para proponer un modelo matemático para evaluar teóricamente los tiempos de retardo en una red sensor inalámbrica con topología lineal a gran escala que opere con protocolo 6LowPAN, es necesario identificar las variables que influyen en el tiempo de retardo que de una red inalámbrica de sensores con topología lineal.

El tiempo de retardo en la transmisión de una trama desde un nodo v1 a otro nodo v2 se compone del tiempo de retardo por procesamiento en el nodo, del tiempo de retardo por propagación para que la señal viaje desde el nodo v1 al nodo v2, y el tiempo que el nodo requiere para transmitir la trama.

La velocidad utilizada es de 250 kbps, por lo tanto 1 símbolo se compone de 4 bytes y tiene un período Ts = 16μs

El protocolo IEEE 802.15.4 define el tiempo TIFS (Interframe Space time) como el tiempo que requiere el nodo para procesar la trama, y está definido por el período mínimo de separación entre tramas (IFS). La duración del período IFS depende del tamaño de la trama. La trama IEEE 802.15.4 con una longitud de hasta 18 bytes debe ir seguida de un período SIFS de al menos 12 períodos de símbolos. Si la trama tiene una longitud mayor a 18 bytes deben ir seguidas de un LIFS de al menos 40 períodos de símbolos.

El tiempo requerido para la transmisión de una trama con una carga útil de x byte, Tfra(x) se la puede calcular de la siguiente manera:

$$T_{fra}(x) = 8\left(L_{phy} + L_{MACHDR}\, L_{address} + x + L_{MACFTR}\right)/\, R_{data} \quad (2.1)$$

En esta expresión x representa el número de bytes que deben ser encapsulados en la trama IEEE 802.15.4, L_{phy} es la longitud de la cabecera PHY de sincronización en bytes (6 bytes), LMACHDR es la longitud de la cabecera MAC en bytes (3 bytes), $L_{address}$ es la longitud del campo de direcciones MAC, L_{MACFTR} es la longitud del campo de fin de trama MAC (2 bytes), R_{data} es la velocidad de transmisión (250 kbps).

El tiempo de transmisión de una rama ACK puede ser calculado de la siguiente manera

$$T_{ACK} = \left(L_{phy} + L_{MACHDR} + L_{MACFTR}\right)/\, R_{data} \quad (2.2)$$

Debido a que la distancia entre los nodos es mínima, el tiempo de propagación se considera insignificante. Por lo tanto, el tiempo que necesita Tnrr el nodo v1 para transmitir al nodo v2, tomando en cuenta la figura 1.22 se lo puede calcular de la siguiente manera

$$T_{nrr}(x) = T_{CCA} + T_{BO} + T_{fra}(x) + T_{TA} + T_{ACK} + T_{IFS} + \zeta \quad (2.3)$$

TBO es período de espera (backoff), y se calcula como el producto entre el número de ranuras de backoff y el tiempo de cada ranura (20 símbolos). El número de ranuras que forman parte del tiempo de backoff es un número aleatorio que se obtiene de la siguiente expresión (0, 2BE-1), con un valor mínimo de BE igual a 3. Todos los nodos inician el cálculo de este periodo de espera con el valor igual a 3

- TCCA representa el tiempo que el nodo requiere para determinar si el canal está libre
- TTA es el tiempo que requiere el nodo para pasar del modo de TX a modo RX y viceversa
- TACK es tiempo de transmisión para una trama ACK
- ζ el tiempo de retardo por propagación

Si el modo de operación es el modo no ranurado, tomando como referencia la figura 1.15 para que el nodo transmita la siguiente trama se debe incluir el tiempo de inactividad Ti y el tiempo de duración de la super trama TSD. En este caso el tiempo de retardo Tnrr es

$$T_{nrr}(x) = T_{CCA}+T_{BO} + T_{fra}(x)+T_{TA}+T_{ACK}+T_{IFS}+T_{SD}+T_i + \zeta \quad (2.4)$$

Esta expresión considera que el número de nodos que existen en la zona de cobertura es pequeño, y que cada nodo puede transmitir en cada una de las 15 ranuras del periodo de actividad, por lo que los nodos no van a sufrir retardos por competir por el uso de una ranura, esta aseveración es cierta considerando las características de estructuras lineales a gran escala, por ejemplo, utilizadas en el monitoreo de oleoductos en donde la ubicación de los nodos sensores define una separación de nodos cada 15 metros. En el peor de los casos

2.6 IMPLEMENTACIÓN DEL PROTOTIPO

En esta parte se va a detallar cada una de las pruebas y las condiciones en que se realizaron cada una de ellas.

Para esto hay que considerar que todas las pruebas se realizaron con los nodos en topología lineal, la idea principal es que estos nodos se encuentren en un espacio abierto y trabajando a la menor potencia para que puedan ubicarse de tal manera que el Sniffer Atmel, realice la captura de paquetes de todos los nodos participantes de la red.

Gracias al Sniffer se pudo analizar los datos producidos por estos sensores, hay que acotar que los nodos fueron configurados reduciendo la potencia al mínimo en el caso de los sensores mote runner IBM, al reducir la potencia mínima nos dio un rango de cobertura de 20 metros. En este caso los sensores fueron colocados a una distancia menor a ella, debido a que el propósito es que dos sensores estén en la zona de cobertura del primer sensor transmisor, facilitando la detección si en un caso el sensor tenga un desperfecto o este nodo pierda conectividad. La red en base a los protocolos que maneja puede autoconfigurarse y así puede seguir transmitiendo los datos, sin la necesidad de obligatoriamente tener que esperar a que el nodo se reestablezca, gracias a esto se gana eficiencia en la red y confiabilidad.

Hay que acotar que para poder tener un mejor análisis dentro del proyecto debido a que los sensores escogidos solo trabajaban en modo ranurado, se decidió realizar pruebas con otros tipos de sensores como es de la tecnología MEMSIC, estos nodos se manejan de manera similar a los nodos de IBM. Se tomó en cuenta los mismos parámetros, como la configuración a menor potencia, estos nodos manejan un protocolo basado en IPv6 para el enrutamiento y levantamiento de la red, es por eso que adicional decidimos utilizarlos

para realizar las pruebas. Cabe mencionar que antes para armar la topología debemos considerar el alcance de los sensores.

Es por eso que al armar la topología, se debe considerar las distancias máximas, al realizar las pruebas se pudo observar que la frontera o el área de cobertura era de 20m a la redonda, pero al querer que nuestra topología lineal funcione; en caso de generar algún inconveniente consideramos colocar a menor distancia, como ya se mencionó anteriormente el alcance con la reducción de potencias es de 20m, recomendamos por esta razón colocar cada nodo sensor a siete metros y medio, esto al realizar la pruebas con los nodos IBM; por otro lado en el caso de las MEMSIC al regular la potencia se determinó la medición con un rango aproximado de 5m de distancia antes de llegar a la frontera del área de cobertura, basándonos en la misma idea de la topología lineal se colocó a los nodos a los 2,5 metros para poder cumplir con las condiciones de la topología.

En la figura 2.1 que se muestra a continuación se establece las áreas de cobertura de cada uno de los nodos, y la disposición en topología lineal para que los nodos puedan converger en caso de nodo fallido. De esta manera, determinamos un modelo en el cual se ha considerado la posibilidad de que un nodo fallido, sea reemplazado por otro sensor el cual brinde cobertura de área fallida, con esto manteniendo una red más eficiente y confiable.

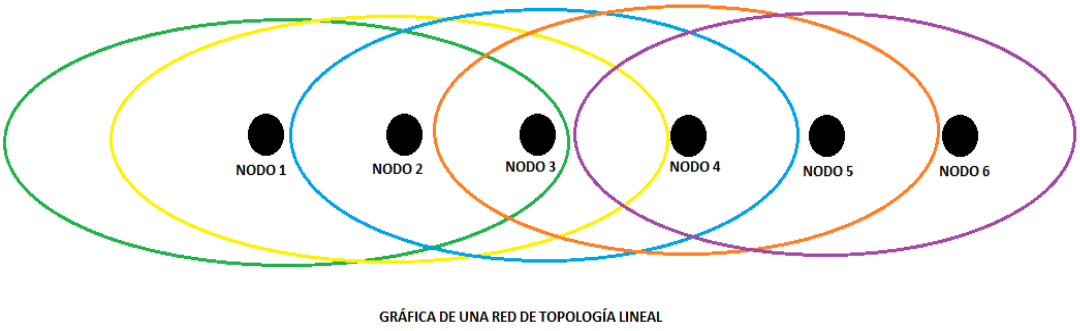

Figura 2.1. Gráfica de una red de topología lineal

Ya que hemos definido la topología de la red y las ventajas de utilizar la misma anteriormente**,** vamos a proceder a detallar las diferentes pruebas que se realizaron, de esta manera se podrá comprender de mejor manera cada una de ellas, vale acotar que el objetivo principal de nuestro proyecto es sacar una caracterización del comportamiento de este tipo de red a gran escala partiendo de estas pruebas a escala reducida, es por esta razón que la primera prueba que se va a realizar es el tiempo de convergencia de la red.

2.7 TIEMPO DE CONVERGENCIA DE LA RED

El tiempo de convergencia, es una de las medidas más importantes al momento de evaluar la eficacia de la red puesto qué el tiempo de convergencia lo podríamos definir como el tiempo que tarda entre que el primer nodo se enciende, hasta que el último nodo se enciende y se enlaza a la red que hemos formado, el tiempo de convergencia se ha medido hasta que se envía el primer dato en la red conformada.

Podemos decir que ese sería el tiempo de convergencia desde que se enciende la red hasta que llegar al último nodo, y así perteneciendo en la red enlazada, ya que en esta prueba el objetivo es observar cuanto es el tiempo que tarda la red en conectarse con todos los nodos sensores, uno con el otro y que estén listos para poder comunicarse entre sí, consideramos algunos parámetros que nos ayudan de alguna manera a cuantificar este dato tan importante de la red de sensores en topología lineal. Para poder medir este tiempo se estableció que se medirá el tiempo desde que se enciende el primero de los nodos hasta llegar al último de los nodos sensores, en este caso al querer que sea el tiempo lo más real posible se procedió a encender cada uno de los nodos al mismo tiempo y en las posiciones asignadas con anterioridad, con esto observar como el protocolo encargado establece las rutas en base al censado de la potencia de cada uno de los sensores, y así determinando cual es el sensor más cercano para realizar la comunicación.

De esta manera se pudo constatar que la red de topología lineal con los diferentes equipos trabaja de la misma manera, realizando la conexión entre los nodos para que cumplan con la condición de topología lineal. Ahora ya colocados en sus posiciones se procede a encender cada uno de los nodos para que en base al protocolo se configure la red, de esta manera al estar encendidos los nodos conforman la topología que estamos configurando, para medir el tiempo de convergencia de la red a través de un Sniffer realizamos la captura de paquetes, y a través de eso medimos los tiempos que tardan en enviar un paquete desde el primer nodo para crear la red, y el paquete final lo reciba el último nodo de la topología, así podemos revisar los tiempos entre el primer y último nodo, logrando obtener el tiempo de convergencia de la red. Esta fue la primera prueba que se realizó.

2.8 TIEMPO DE CONVERGENCIA ANTE UN NODO FALLIDO

Adicional al tiempo de convergencia de la red, uno de los datos que se necesitan conocer es el tiempo de restablecimiento de la red, este tiempo se considera el tiempo que demora la red en encontrar una nueva ruta para el envío de los paquetes, debido a que el nodo al que se iba direccionado cuenta con un desperfecto. Obviamente que al hablar de múltiples

nodos se debe considerar la posibilidad de que uno de ellos pueda llegar a estar defectuoso, ya sea por causa de su batería o por algún otro tipo de desperfecto. Al considerar esos detalles y si queremos tener confiabilidad en la red de sensores, medir el tiempo de restablecimiento de la red frente a un fallo, este tiempo nos puede ayudar con la confiabilidad de la red frente algún daño.

Por esta razón, al utilizar esta topología lineal lo que se intenta es que sí por alguna razón llega a fallar un nodo, la red no quede inoperable, sino que encuentre otra ruta o un nodo dentro de su área de cobertura con el cual pueda entablar comunicación, y así realizar la transmisión de los datos y de esta manera continuar con la transmisión garantizando la confiabilidad en la red.

Es por eso la importancia de delimitar las zonas de cobertura de los nodos y que se encuentren dos nodos dentro de la zona, así lo que intentamos es que si un nodo pierde la red se mantenga la comunicación por medio de un sustituto, en virtud de lo expuesto anteriormente la prueba a realizarse consistirá en apagar un nodo de la red, y desde el momento en que se apaga, en el nodo no existirá comunicación impidiendo continuar con el envío de paquete de información, una vez apagado se medirá el tiempo que tarda el nodo en buscar otra ruta y enviar el paquete de información, es decir suponiendo que queremos enviar información por cada uno de los nodos de la red de topología lineal.

En el caso de que, el paquete se dirige del nodo 1, al nodo 2, y del nodo 2 al 3, y así sucesivamente hasta llegar al destino, cuando el nodo fallido es el nodo 2, el nodo 1 al no poder comunicarse con nodo 2, envía un mensaje para encontrar una nueva ruta, y de esta manera continuar con el envío de la información sin tener que pasar por el nodo fallido, ya que al estar dañado o fuera de servicio imposibilitaría la comunicación de la red.

En la gráfica que se muestra a continuación se puede observar cómo los nodos que están dentro del área de cobertura pueden realizar la comunicación si un nodo vecino esta fuera de servicio.

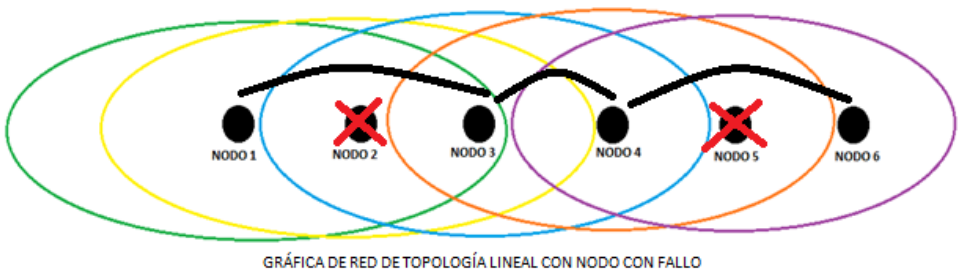

Figura 2.2. Gráfica de una red de topología lineal con nodo con fallo

Para realizar la prueba se apaga uno de los nodos sensores para simular un nodo fallido, y de esta manera poder medir el tiempo que el nodo tarda en encontrar una nueva ruta cuando se tiene un nodo fallido en la ruta. Se pretende medir el tiempo que tarda un paquete en llegar al nodo siguiente luego de apagar el nodo intermedio, es así como a través del sniffer realizamos la medida de tiempo desde que el nodo envía el dato, al nodo fallido y este al no recibir un mensaje de confirmación pues el protocolo ejecuta la búsqueda de otro nodo que este cerca de la zona de cobertura, y se pueda transmitir su mensaje sin la necesidad de pasar por el nodo fallido, a este tiempo se lo conoce como el tiempo de restablecimiento de la red.

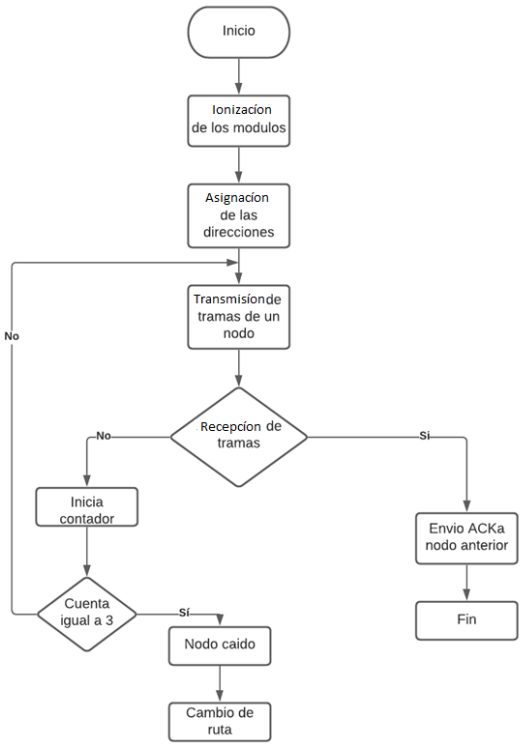

Figura 2.3. Algoritmo para Minimizar el Efecto de un Nodo Fallido

2.9 TIEMPO DE RETARDO EN EL NODO POR PROCESAMIENTO

Vamos a mencionar que dentro de esta prueba de tiempo de retardo por procesamiento se realizaron dos formas posibles, la primera que consiste en el envío del tiempo del nodo con encapsulamiento del payload es decir que en este proceso debemos des encapsular, procesar y encapsular nuevamente y la otra prueba donde no vamos a realizar este proceso, sino solo recibir la información y retransmitirla hacia los otros nodos, así evaluaremos los dos tiempos de procesamiento.

Este es otro de los tiempos que a través del Sniffer podemos medir lo podríamos catalogar como el tiempo de procesamiento en el nodo, como nosotros sabemos la idea es que la información generada debe ser enviada y transmitida por la red, pero cada nodo puede incrementar la información o ir modificando su cabecera, y de esa manera ir revisando la ruta que la información toma, así conocer de los daños que existen en la red, así se puede decir que es un procesamiento ya que el nodo recibe la información la des encapsula para luego nuevamente encapsular, cambiando la dirección de destino y lógicamente la dirección origen, puesto que al pasar de un nodo a otro nodo, lo que va cambiando es la dirección de destino que cuando llega se convierte en dirección origen, entonces si al medir los tiempos revisamos cuando llega el paquete al nodo, y luego por el Sniffer vemos lo que ese paquete es enviado con otras direcciones pero con la misma información, a este procedimiento le podemos medir y nombrarlo como el tiempo de procesamiento, de esta manera podemos obtener el tiempo que se va demorar des encapsulando el paquete y encapsulándolo nuevamente, generando las nuevas direcciones.

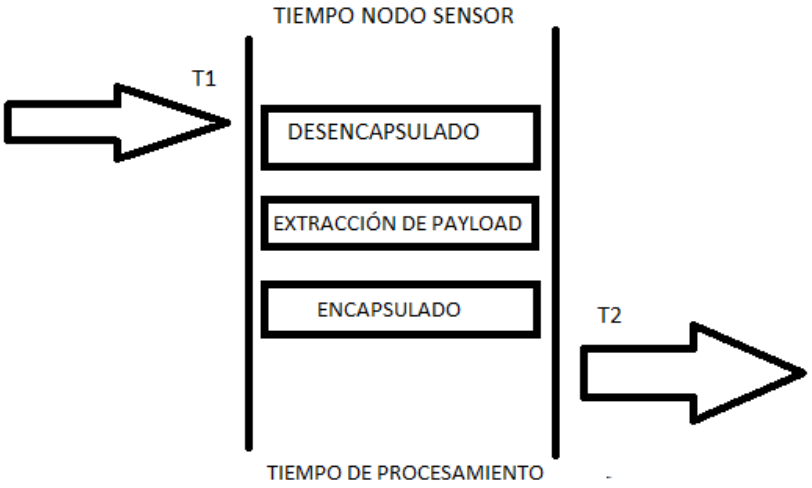

Figura 2.4. TIEMPO DE PROCESAMIENTO

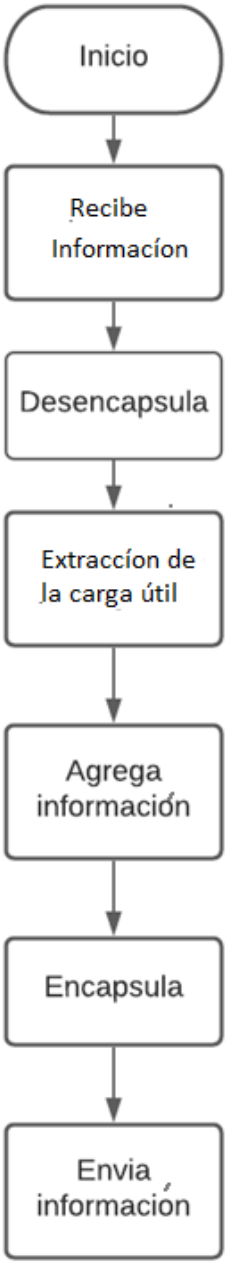

Figura 2.5. Diagrama de flujo del tiempo de procesamiento

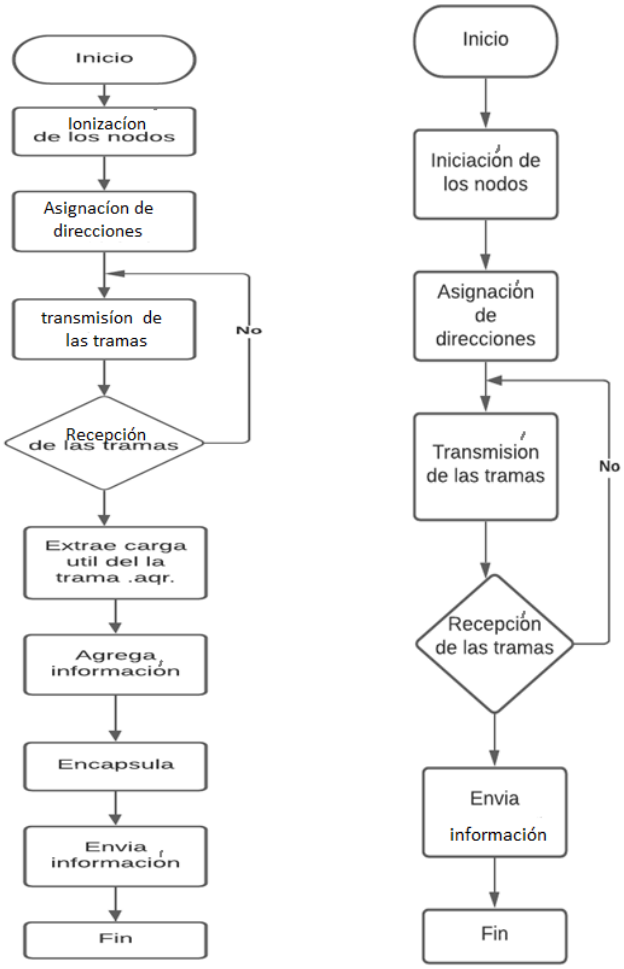

Figura 2.6. Diferencia en los diagramas de flujo en retransmisión con y sin procesamiento de la información en un nodo

2.10 TIEMPO DE RECUPERACIÓN POR UN ENLACE FALLIDO

Un fallo de enlace se considera cuando, de un nodo se va a transmitir la información a otro nodo, pero al momento de recibir no tiene respuesta de la recepción del paquete de información, para esto se considera el número de intentos que realiza el nodo para hacer llegar la información. Cada nodo tiene un contador el cual establece el número de transmisiones fallidas, en el caso de los nodos con los cuales se va a trabajar, el número de retransmisiones fueron dos, al tercer intento de no recibir confirmación en la entrega del paquete de información, automáticamente el nodo busca una nueva ruta para enviar el paquete de información, y declara al nodo anterior como nodo caído, esto generando para próximas entregas cambio de ruta de envío de información.

Podríamos decir qué, para evaluar este tiempo de fallo de enlace, se debe observar el número de intentos que realiza el envío del mismo paquete, y los tiempos entre el primer envío del paquete y el último intento de envío de información, antes de cambiar la ruta del paquete.

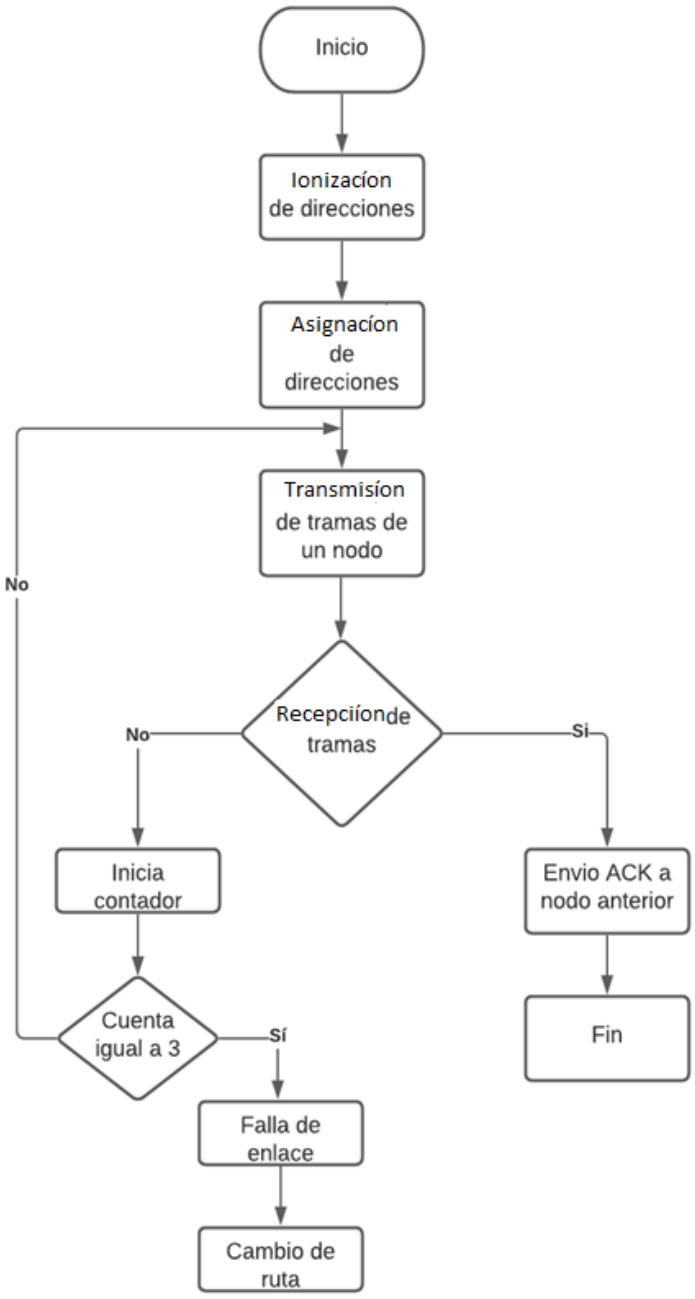

Figura 2.7. Algoritmo para fallo de un enlace

2.11 IMPLEMENTACIÓN

Para poder realizar las pruebas se codifico los diagramas de flujo presentados anteriormente.

2.12 CODIFICACIÓN

Conforme a los diagramas de flujo definidos anteriormente se plantean los códigos para que el prototipo funcione de acuerdo con los parámetros de la red para eso a continuación mostramos los diferentes programas que se utilizaron para que el prototipo funcione.

2.12.1 CÓDIGO DE IMPLEMENTACIÓN DE LA RED CON LIBELIUM

Como se ha mencionado anteriormente lo primero que se realiza es el levantamiento de la red, en esta parte del código se puede ver como se levanta la red definiendo cada uno de los nodos que van a participar en la topología lineal y las funciones que va a cumplir el nodo basándose en la dirección origen y dirección destino ya al levantar la red.

```
namespace com.ibm.saguaro.mrv6.example
{
      using com.ibm.saguaro.mrv6;
      using com.ibm.saguaro.system;
      using com.ibm.saguaro.util;

      public class NodoIntermedio : UDPSocket
      {
            /// <summary>Port that socket binds to </summary>
        internal static uint LOCAL_PORT = 1023;

        /// <summary>Singleton instance </summary>
        internal static NodoIntermedio socket = new NodoIntermedio();

            //Direction destination
            internal static uint direccionDestino = 0;

            internal static uint direccion = 0;

            internal static byte [] datos = null;
        /// <summary>Class to blink LED </summary>
        //internal LEDControl ledctrl;

        /// <summary>Constructor </summary>
        public NodoIntermedio() {
            //ledctrl = new LEDControl(2);
            this.bind(LOCAL_PORT);
        }

        /// <summary>Receive packet </summary>
        public override int onPacket(Packet packet) {
            //ledctrl.setState(1, 500);
            LED.setState(1, (byte)(LED.getState(1)^1));

                  direccion = Mac.getMyShortAddr();
```

```
                direccionDestino = Mac.getParentShortAddr();
            uint len = packet.payloadLen;
            datos = new byte[len];
            Util.copyData(packet.payloadBuf, packet.payloadOff, datos, 0,
len);
            Packet packet2 = Mac.getPacket();
                Address.fromSaddr(packet2.dstaddr,                    0,
direccionDestino);
                //Address.fromSaddr(packet.srcaddr, 0, direccion);
                packet2.srcport = LOCAL_PORT;
              packet2.dstport = 1023;
                try {
                      packet2.create(1023,                  LOCAL_PORT,
(uint)datos.Length /*frst two ADC channels 0 & 1*/);
              } catch {
                   // XXX
                   return 0;
              }
            Util.copyData(datos,          0,          packet2.payloadBuf,
packet2.payloadOff, len);
            this.send(packet2);
            return 0;
        }
      }
}
```

2.12.2 CÓDIGO DE LIBELIUM SIN PROCESAMIENTO

En base al diagrama de flujo de la figura 2,6 se va a mostrar el código a utilizar para realizar la transmisión de los datos sin procesamiento.

Así podemos observar como el dato que llega es retransmitido sin des encapsularlo solo se reenvía la información.

```
namespace com.ibm.saguaro.mrv6.example
{
      using com.ibm.saguaro.mrv6;
      using com.ibm.saguaro.system;
      using com.ibm.saguaro.util;

      public class NodoIntermedio : UDPSocket
      {
            /// <summary>Port that socket binds to </summary>
        internal static uint LOCAL_PORT = 1023;

        /// <summary>Singleton instance </summary>
        internal static NodoIntermedio socket = new NodoIntermedio();

            //Direction destination
            internal static uint direccionDestino = 0;

            internal static uint direccion = 0;
```

```
            internal static byte [] datos = null;
        /// <summary>Class to blink LED </summary>
        //internal LEDControl ledctrl;

        /// <summary>Constructor </summary>
        public NodoIntermedio() {
            //ledctrl = new LEDControl(2);
            this.bind(LOCAL_PORT);
        }

        /// <summary>Receive packet </summary>
        public override int onPacket(Packet packet) {
            //ledctrl.setState(1, 500);
            LED.setState(1, (byte)(LED.getState(1)^1));

                  direccion = Mac.getMyShortAddr();
                  direccionDestino = Mac.getParentShortAddr();
            uint len = packet.payloadLen;
            datos = new byte[len];
            Util.copyData(packet.payloadBuf, packet.payloadOff, datos, 0,
len);
            Packet packet2 = Mac.getPacket();
                  Address.fromSaddr(packet2.dstaddr,                    0,
direccionDestino);
                  //Address.fromSaddr(packet.srcaddr, 0, direccion);
                  packet2.srcport = LOCAL_PORT;
              packet2.dstport = 1023;
                  try {
                        packet2.create(1023,                    LOCAL_PORT,
(uint)datos.Length /*frst two ADC channels 0 & 1*/);
               } catch {
                     // XXX
                     return 0;
                  }
            Util.copyData(datos,          0,          packet2.payloadBuf,
packet2.payloadOff, len);
            this.send(packet2);
            return 0;
        }
      }
}
```

2.12.3 CÓDIGO DE LIBELIUM CON PROCESAMIENTO

En base al diagrama de flujo de la figura 2,6 se va a mostrar el código a utilizar para realizar la transmisión de los datos sin procesamiento.

Así podemos observar como el dato que llega es retransmitido y se realiza un procesamiento adicional al extraer el dato y cargarlo con información adicional, el programa lo encapsula nuevamente y lo envía al siguiente nodo de esta manera existe un proceso adicional en cada nodo, con el new byte estamos generando ese dato adicional que hace que exista procesamiento en el nodo

```
namespace com.ibm.saguaro.mrv6.example
{
      using com.ibm.saguaro.mrv6;
      using com.ibm.saguaro.system;
      using com.ibm.saguaro.util;

      public class NodoIntermedioProcesa : UDPSocket
      {
            /// <summary>Port that socket binds to </summary>
        internal static uint LOCAL_PORT = 1023;

        /// <summary>Singleton instance </summary>
        internal    static    NodoIntermedioProcesa    socket    =    new
NodoIntermedioProcesa();

            //Direction destination
            internal static uint direccionDestino = 0;

            //Direction source
            internal static uint direccion = 0;

            internal static uint maxMotes = 5;

        /// <summary>Class to blink LED </summary>
        //internal LEDControl ledctrl;

        /// <summary>Constructor </summary>
            public NodoIntermedioProcesa ()
            {
                  //ledctrl = new LEDControl(2);
            this.bind(LOCAL_PORT);
            }

            /// <summary>Receive packet </summary>
        public override int onPacket(Packet packet) {
            //ledctrl.setState(1, 500);
            LED.setState(1, (byte)(LED.getState(1)^1));

                  direccion = Mac.getMyShortAddr();
                  direccionDestino = Mac.getParentShortAddr();
            uint len = packet.payloadLen;
            byte [] aux1 = new byte[len];
            Util.copyData(packet.payloadBuf, packet.payloadOff, aux1, 0,
len);
            Packet packet2 = Mac.getPacket();
                  Address.fromSaddr(packet2.dstaddr,                    0,
direccionDestino);
                  packet2.srcport = LOCAL_PORT;
              packet2.dstport = 1023;
                  try {
                        packet2.create(1023, LOCAL_PORT, (uint)aux1.Length
/*frst two ADC channels 0 & 1*/);
              } catch {
                   // XXX
                   return 0;
              }
```

```
                  for(uint i =0; i<len; i++)
                  {
                        if(aux1[i]==0x00 || aux1[i]==direccion)
                        {
                              aux1[i]=(byte)direccion;
                              break;
                        }
                  }
                  //Agregar(len, datos);
            Util.copyData(aux1,           0,           packet2.payloadBuf,
packet2.payloadOff, len);
            this.send(packet2);
            return 0;
        }

            static void Agregar(uint len, byte[] buf)
            {
                  for(uint i =0; i<len; i++)
                  {
                        if(buf[i]==0x00 || buf[i]==direccion)
                        {
                              buf[i]=(byte)direccion;
                              break;
                        }
                  }
            }
      }
}
```

2.12.4 CÓDIGO DE MEMSIC CON PROCESAMIENTO

Como se menciona en la figura 2.6 donde indicamos los diagramas de flujo, para realizar las pruebas en modo no ranurado se ocuparon los nodos MEMSIC, en los cuales igualmente que en los nodos Libelium la lógica del diagrama de flujo es la misma solo se modifica que no vamos a tener slots de tiempo para la transmisión; al ser el mismo esquema de trabajo pero sin slots o ranuras de tiempo lo que se modifica en la lógica de la programación es como definir el procesamiento, este se va a realizar con el des encapsulamiento y encapsulamiento de la información ese es el proceso extra que se lo entiende como procesamiento en los nodos memsic

```
#include "Comunicacion.h"
module ComunicacionC{
      /*Intefaz para leds*/
   uses interface Leds;
   /*Interfaz de booteo*/
   uses interface Boot;
   /*Interfaz para recepcion de mesaje*/
   uses interface Receive;
   /*Interfaz para radio*/
   uses interface AMSend;
   /*Intefaz para control de canal de radio*/
   uses interface SplitControl as AMControl;
```

```
   /*Intefaz de paquete para payload*/
   uses interface Packet;
   uses interface Timer<TMilli> as TimerWireless;
}
implementation{
       /*Ponemos nuestro paquete con la estructura message_t creada en el
header*/
       message_t packet;

     /*Iniciamos la variable para contador*/
     uint16_t contador = 0;

      event void Boot.booted() {
            /*Evento de booteo se inicia con el llamado a control de canal
*/
       call AMControl.start();
      }
     event void AMControl.startDone(error_t err) {
         if (err == SUCCESS) {
            comunicacion_msg_t*   paquete   =   (comunicacion_msg_t*)call
Packet.getPayload(&packet, sizeof(comunicacion_msg_t));
                paquete->contador = contador;
            paquete->nodo = TOS_NODE_ID;

         }
        else {
            /*Si el control del canal no inicio correctamente se vuelve a
llamar*/
          call AMControl.start();
        }
      }
     event void AMControl.stopDone(error_t err) {
         /* No hace nada Pero e lo invoca porque tiene que ejecutarse todos
los
          * eventos en AMControl*/
      }

  event void AMSend.sendDone(message_t* bufPtr, error_t error) {
    if (&packet == bufPtr) {

    }
  }
  event message_t* Receive.receive(message_t* bufPtr,
                           void* payload, uint8_t len) {
     /*Comprobamos que el paquete sea valido*/
    if (len != sizeof(comunicacion_msg_t)) {return bufPtr;}
    else {
            /*Si es valido se extrae del payload de datos*/
     comunicacion_msg_t* paquete = (comunicacion_msg_t*)payload;
     /*Extraemos el ID del nodo sensor y lo comparamos
      *para envio*/


      if (paquete->nodo & 0x05){

               call TimerWireless.startPeriodic(5000);

            }
```

```
            if(paquete->nodo & 0x02){
                  call Leds.led0Toggle();
                  call Leds.led2Toggle();
                call AMSend.send(5, &packet, sizeof(comunicacion_msg_t));
                }
          return bufPtr;
      }
    }



      event void TimerWireless.fired(){
            // TODO Auto-generated method stub
            call AMSend.send(2, &packet, sizeof(comunicacion_msg_t));
      }
}
```

2.13 ESCENARIOS PARA LAS PRUEBAS

Al momento de realizar las pruebas primero se estableció los lugares donde se realizarían las pruebas, para los nodos de Libelium se estableció realizarlas en un ambiente externo debido a que como se mencionó anteriormente el radio del alcance de los sensores es de 20m por esta razón se escogió realizarlas en el parque Equinoccial en el norte de Quito para cumplir con las distancias de los sensores y teniendo el espacio abierto para que no existan objetos que puedan alterar las mediciones de los tiempos al realizar las pruebas.

Figura 2.8. Parque Equinoccial

Los nodos que se utilizaron son los nodos Libelium IBM que se describen en el capítulo anterior,

Figura 2.9. Nodo Libelium

Para los sensores de MEMSIC como se mencionó antes las distancias son cortas cuando se los configura con la menor potencia posible por esa razón las pruebas se realizaron en un ambiente interno ya que las distancias especificadas anteriormente podían cumplirse en este tipo de ambiente sin inconveniente, para que cumpla con la topología lineal que es la que nos estamos basando.

Figura 2.10. Espacio interno para Pruebas nodos MEMSIC

Para las pruebas mencionadas se ocuparon los nodos MEMSIC que se mencionó anteriormente

Figura 2.11. Nodo Memsic

2.14 PRUEBAS

Lo primero que se realizó para realizar las pruebas fue levantar la red en topología lineal con los dos tipos de sensores, para ello se procedió a colocarlos en las posiciones descritas anteriormente respetando las distancias definidas con anterioridad, se tuvo que considerar parámetros como la línea de vista al momento de posicionarlos. Ya que se encontraban en su lugar se fueron encendiendo uno a uno los sensores para que se establezca la red recordando que a través de su configuración los nodos realizan una comparación con los niveles de potencia de cada uno de los sensores y así, se determina la proximidad y cuál es el sensor más próximo para enviar la información con esto se da el establecimiento de la red. Así al prender el sniffer se pudo medir el tiempo de convergencia de la red, esta prueba se hizo al inicializar la red y sirve para confirmar que la red esta levantada y transmitiendo en un orden específico. Posterior a esto se procedió a realizar las pruebas. En la parte de resultados se evidenciará las tablas con los datos obtenidos.

Ya con la red de sensores levantada en las dos tecnologías se procedió a realizar la primera prueba

2.14.1 PRUEBA DEL NODO FALLIDO

En esta prueba lo que se procedió a realizar fue apagar un nodo de la topología lineal y se midieron los tiempos que la topología lineal se demoró para que la red establezca otro camino y pueda enviar la información al siguiente nodo y así el paquete llegue a su destino, de esta manera se pudo medir el tiempo de convergencia de la red por nodo caído mencionado anteriormente.

2.14.2 PRUEBA DEL FALLO DE ENLACE

En esta prueba el objetivo era generar interferencia así colocando barreras para que no exista línea de vista y que el nodo no pueda enviar información de confirmación de que el paquete ha llegado, con las tramas ACK se confirmó el número de intentos posibles que hace el sistema antes de buscar otra ruta debido a que determina que el nodo al que se envió no está a su alcance. Como se menciona con anterioridad al momento de definir el tiempo de recuperación por fallo de enlace.

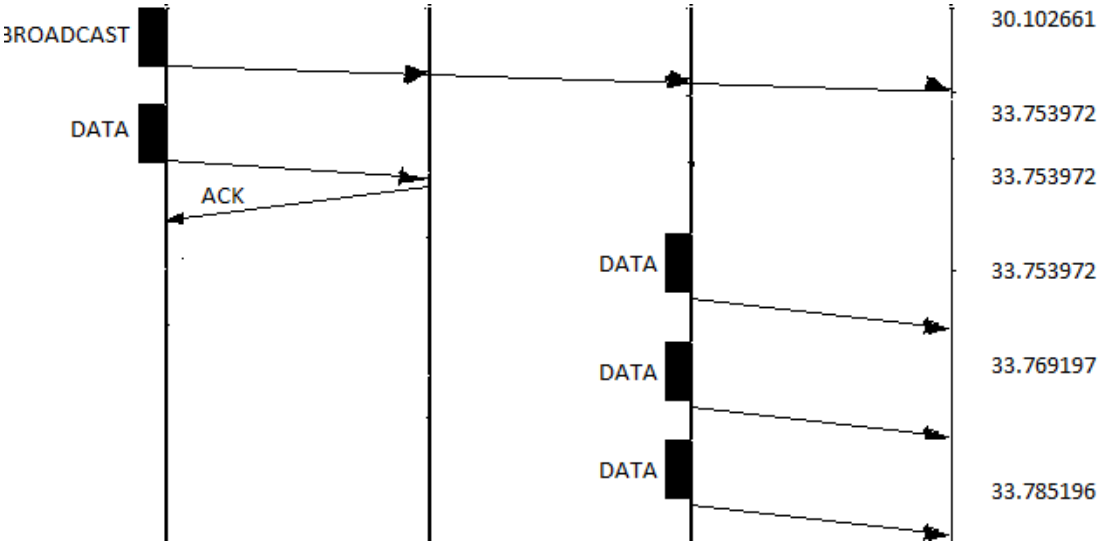

Figura 2.12. Captura de fallo de enlace

2.14.3 PRUEBA DEL TIEMPO DE RETARDO EN EL NODO POR PROCESAMIENTO

En esta prueba se procedió a cambiar el programa, para que des encapsule la información con esta premisa al des encapsular y volverla a encapsular lo que se necesitaba es medir el tiempo que se demora con este procesamiento extra a comparación del tiempo que tarda solo enviar el dato, por ello se tomaron las medidas de los dos tiempos tanto con procesamiento como sin procesamiento para poder llegar a una comparación esta prueba

se realizó con las dos tecnologías por lo ya mencionado anteriormente que tiene que ver con los modos de trabajo beacon de la una tecnología y no beacon de la otra.

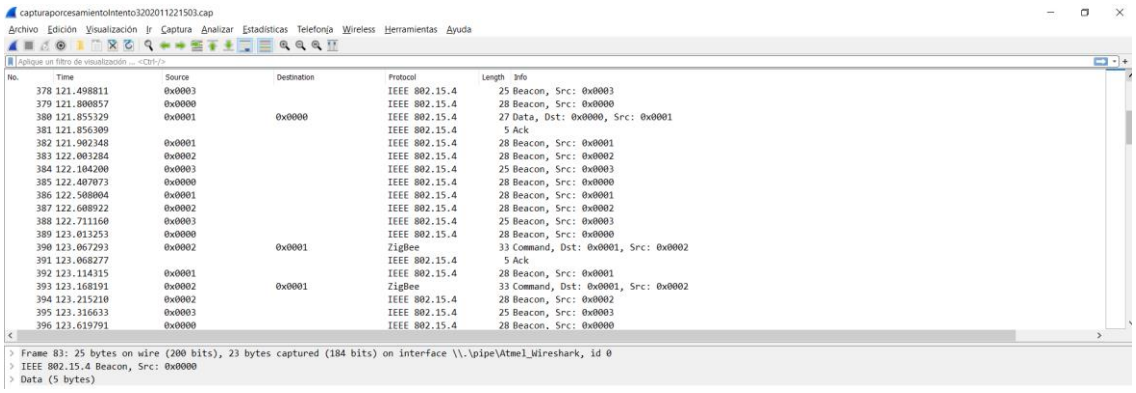

Figura 2.13. 6lowpan con procesamiento

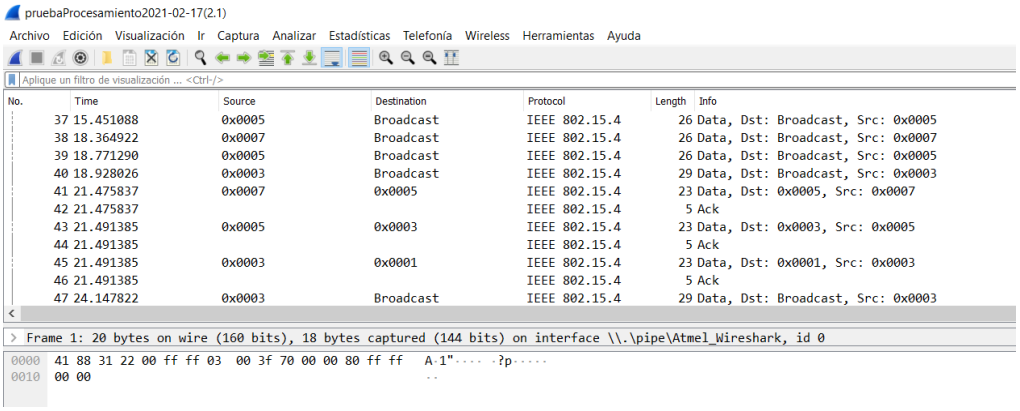

Figura 2.14. Memsic con procesamiento

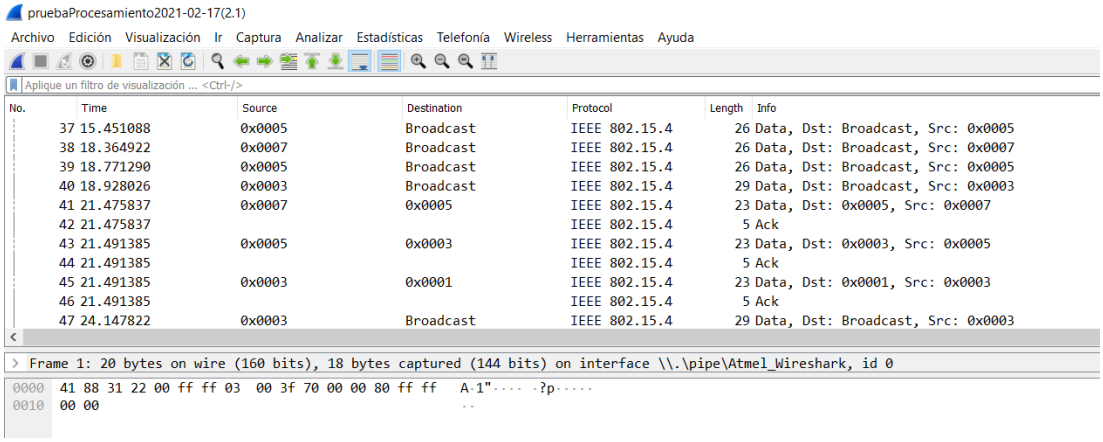

Figura 2.15. Capturas de pantalla del Sniffer

2.14.4 PRUEBA DEL TIEMPO DE RETARDO EN EL NODO SIN PROCESAMIENTO

En esta prueba se procedió a cambiar el programa, para que la información la reenvíe solamente con esta premisa la idea fue medir el tiempo que se demora solo al enviar el dato, por ello se tomaron las medidas de los dos tiempos tanto con procesamiento como sin procesamiento para poder llegar a una comparación esta prueba se realizó con las dos tecnologías por lo ya mencionado anteriormente que tiene que ver con los modos de trabajo de una red ranurada y no ranurada.

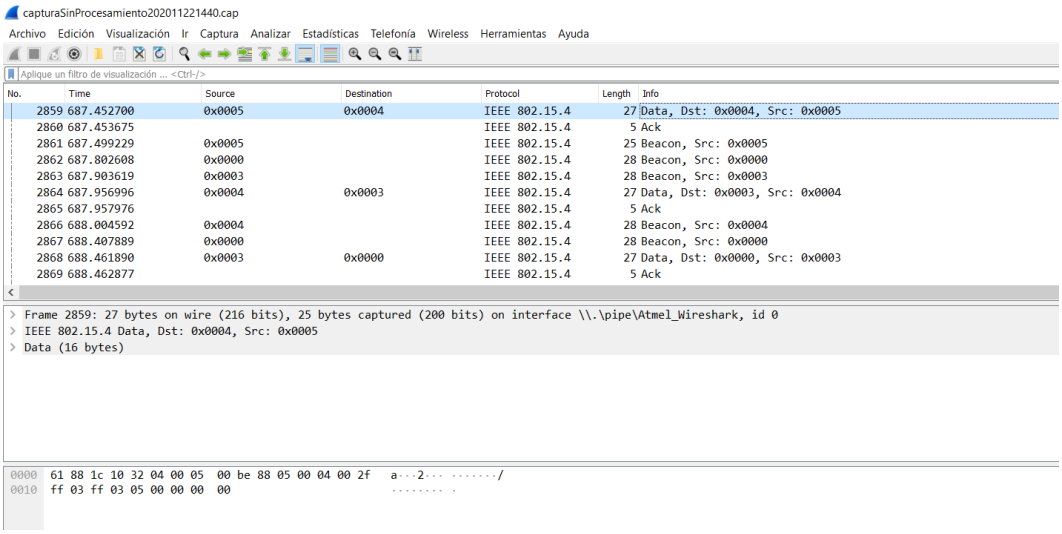

Figura 2.16. 6lowpan sin procesamiento

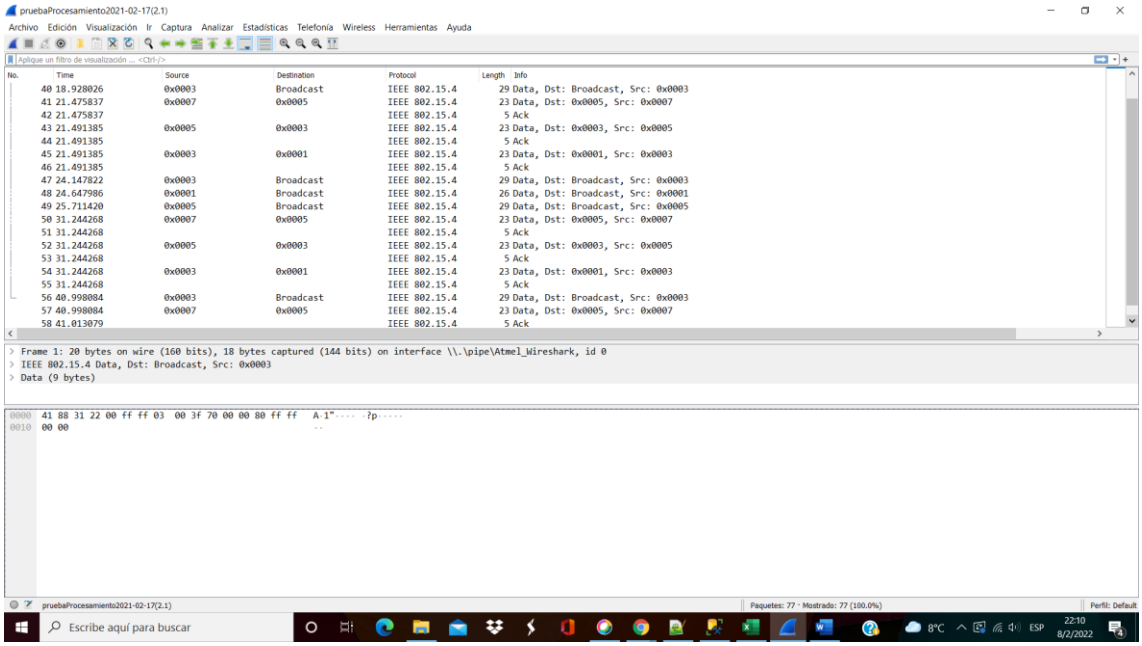

Figura 2.17. Memsic sin procesamiento

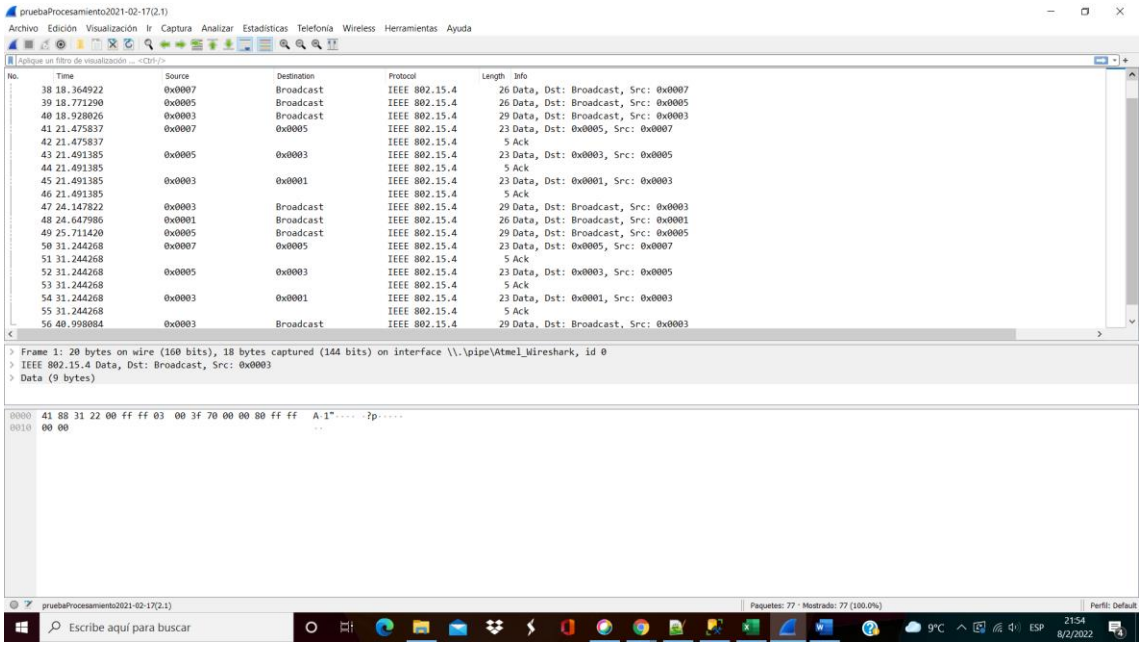

Figura 2.18. Memsic con procesamiento

2.15 Análisis de Gráfica de Tiempos con procesamiento

Aquí vamos a ver un diagrama de tiempos de los nodos Libelium con procesamiento y los nodos memsic con procesamiento

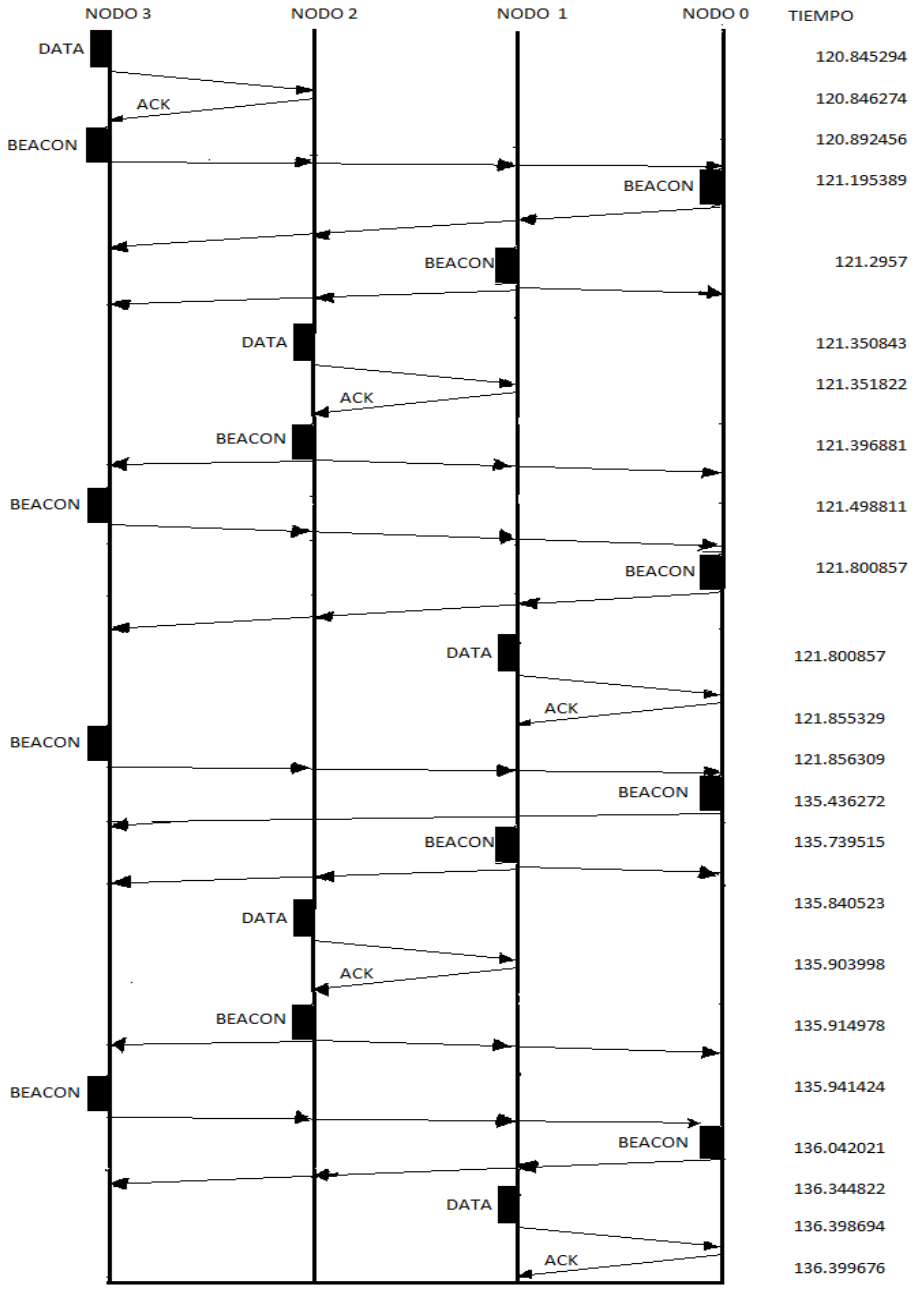

Figura 2.19. 6LOWPAN CON PROCESAMIENTO

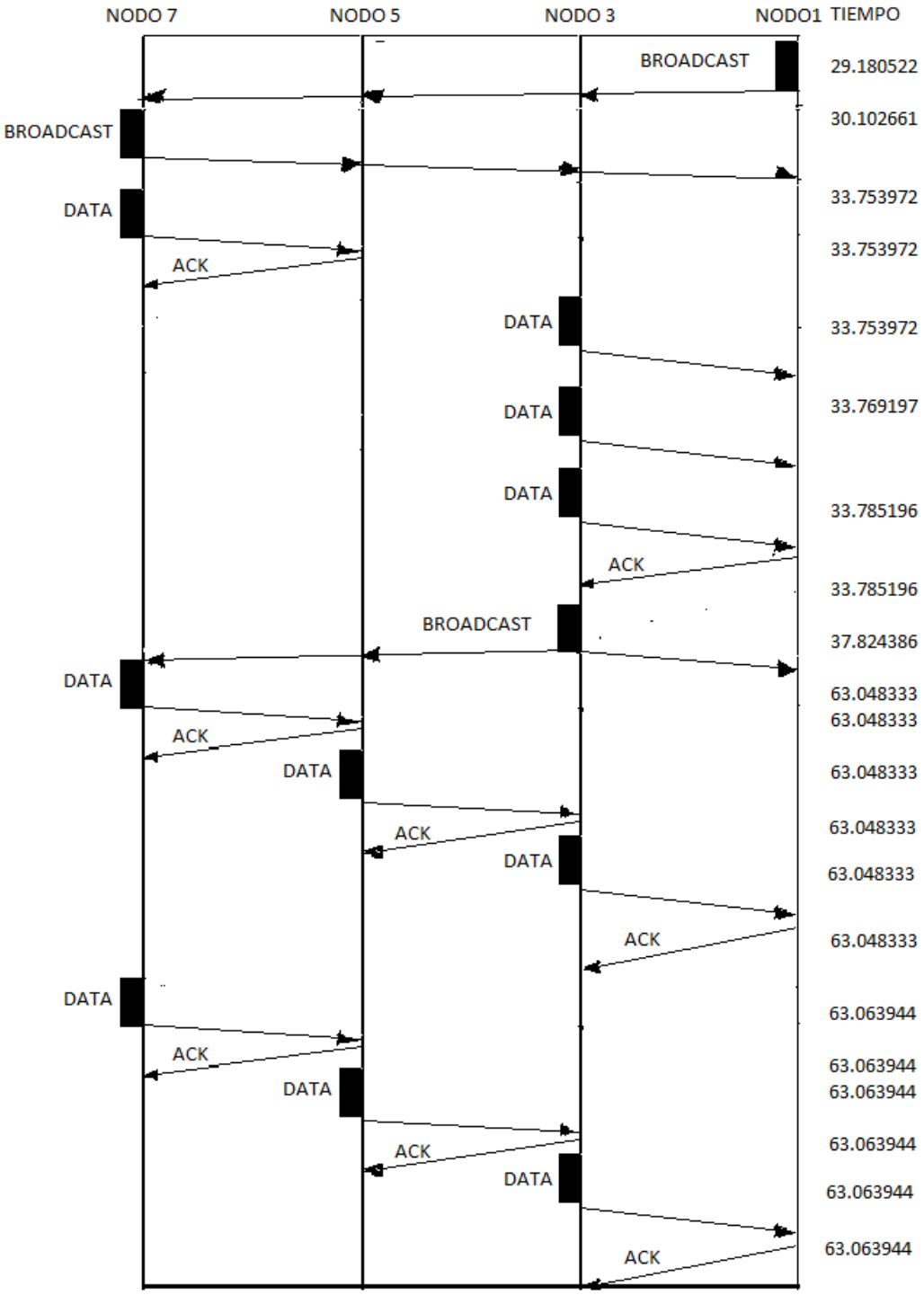

Figura 2.20. MEMSIC CON PROCESAMIENTO

2.16 Análisis de Gráficos con tiempo de procesamiento

Aquí vamos a ver un diagrama de tiempos de los nodos Libelium y Memsic sin procesamiento

MEMSIC SIN PROCESAMIENTO

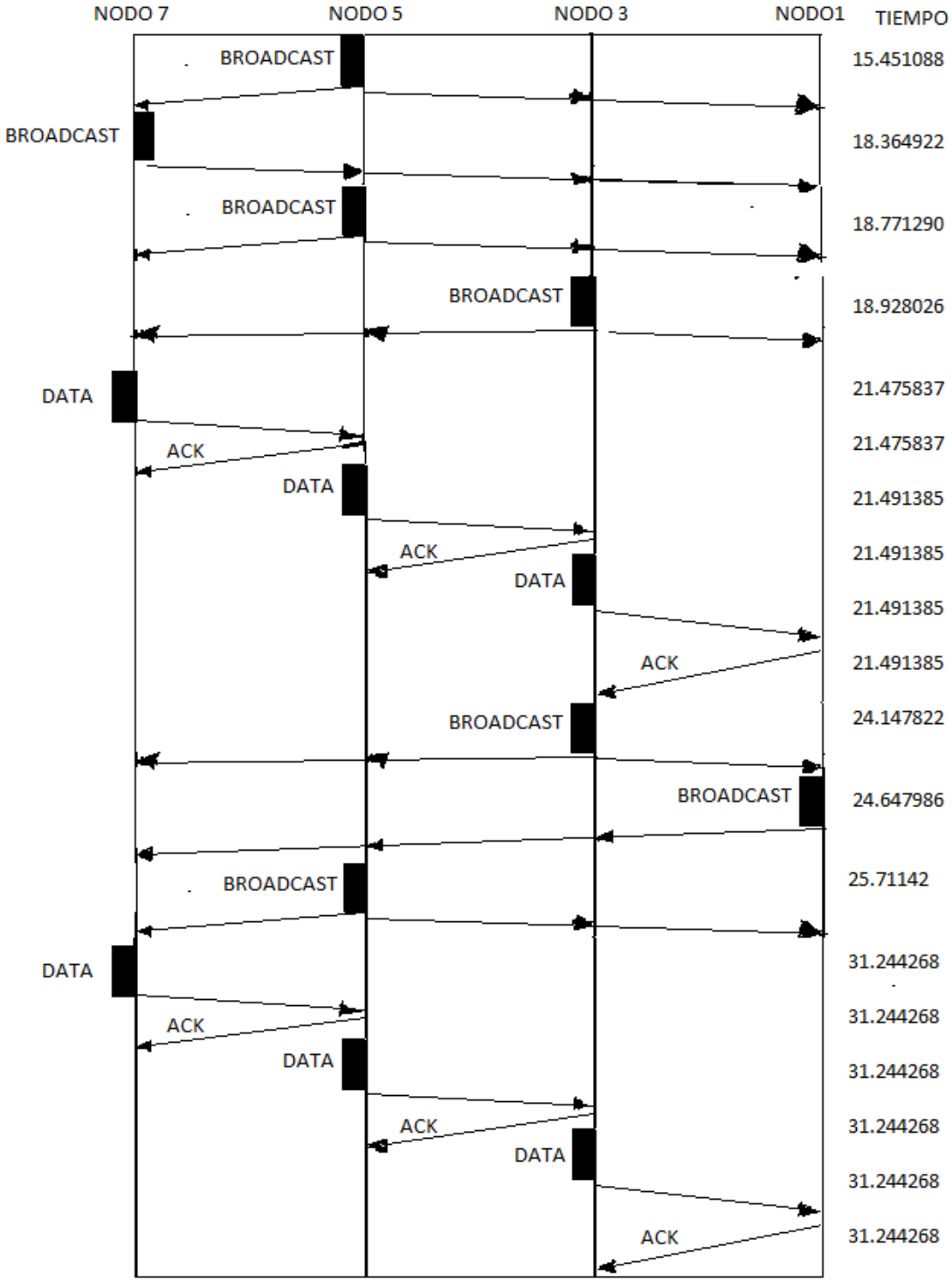

Figura 2.21. MEMSIC SIN PROCESAMIENTO DE PAYLOAD

LIBELIUM SIN PROCESAMIENTO

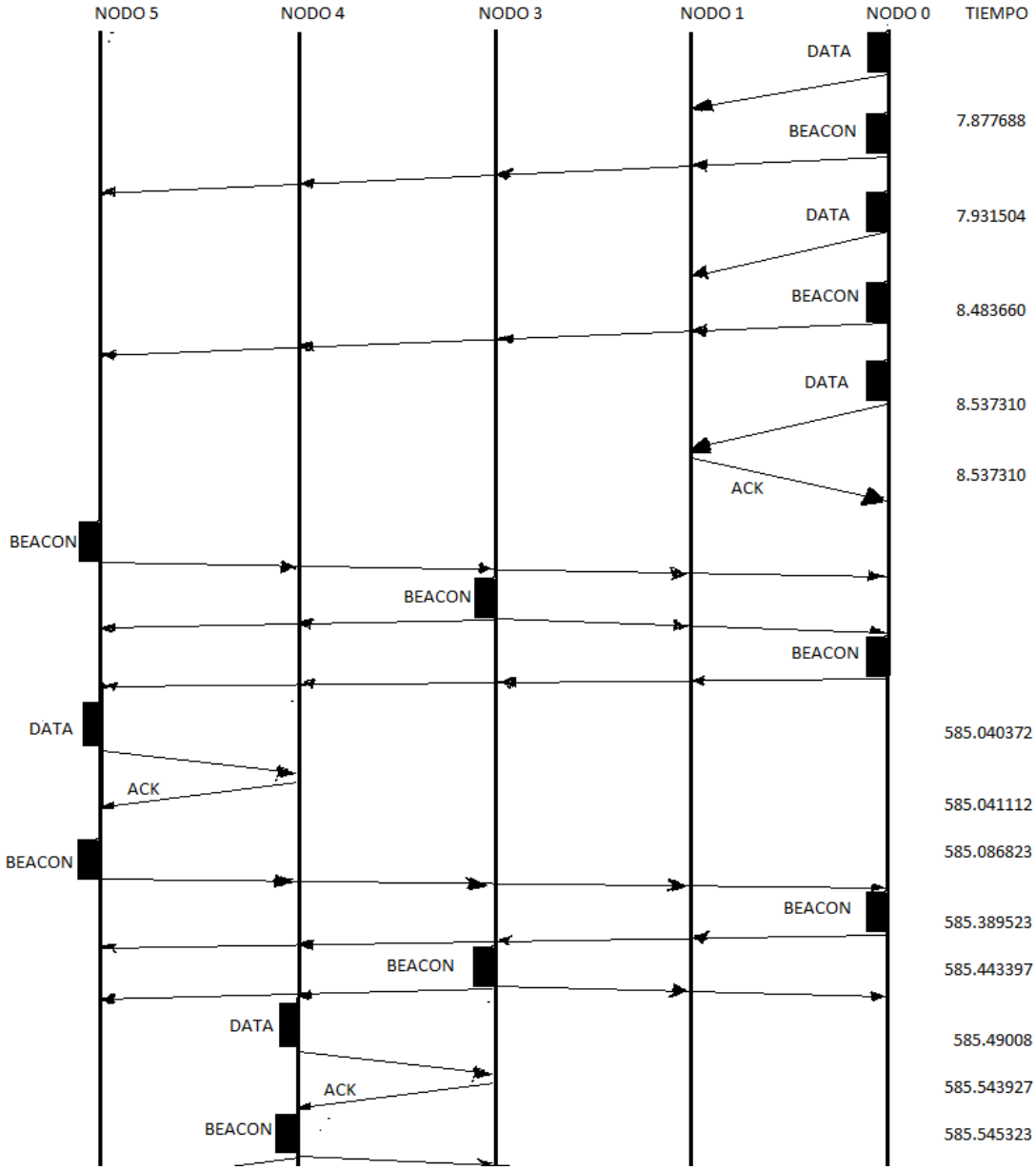

Figura 2.22. 6LOWPAN SIN PROCESAMIENTO

3 RESULTADOS Y DISCUSIÓN

Ya que se definieron los escenarios, la metodología con la que se trabajó, vamos a analizar los resultados obtenidos en las pruebas de campo realizados con los dos tipos de sensores, que debido a las especificaciones nos ayudaron en la caracterización, en este capítulo vamos a entregar los resultados hechos por las pruebas de campo para luego compararlos con la caracterización de la fórmula realizada anteriormente.

TABLA DE TIEMPO DE ESTABLECIMIENTO DE LA RED MEMSIC

Tabla 3.1. TIEMPO DE ESTABLECIMIENTO DE LA RED MEMSIC

TIEMPO	**SOURCE**	**DESTINO**	**TIEMPO seg**
0	0x0001	BROADCAST	
4.751345	0X0002	BROADCAST	
20.912931	0X0003	BROADCAST	
387.8985	0X0005	BROADCAST	
419.00755	0X0003	0X0001	419.00755
0	0X0003	BROADCAST	
0.34364	0X0001	BROADCAST	
5.579393	0X0005	BROADCAST	
13.144161	0X0007	BROADCAST	
21.475837	0X0007	0X0005	21.475837
0	0X0003	BROADCAST	
0.093807	0X0001	BROADCAST	
4.844244	0X0005	BROADCAST	
10.814112	0X0007	BROADCAST	
20.472708	0X0007	0X0003	20.472708

0	0X0003	BROADCAST	
0.093809	0X0001	BROADCAST	
10.17446	0X0005	BROADCAST	
14.910522	0X0007	BROADCAST	
23.991469	0X0007	0X0005	23.991469
0	0X0002	BROADCAST	
3.314056	0X0001	BROADCAST	
5.080097	0X0003	BROADCAST	
10.566089	0X0005	BROADCAST	
20.803384	0X0007	BROADCAST	
28.368157	0X0007	0X0003	28.368157
57.657525	0X0007	0X0005	
		PROMEDIO	23.577043

TABLA DE ESTABLECIMIENTO DE LA RED LIBELIUM

El tiempo de establecimiento de la red es el tiempo desde que se enciende el primer nodo hasta que se enciende el último y entablan una comunicación entre todos los nodos en este caso el tiempo vario debido a la distancia que se encuentra el nodo del siguiente por esta razón el tiempo promedio es de **49.362219 seg**

Tabla 3.2. ESTABLECIMIENTO DE LA RED LIBELIUM

Tiempo unidad	**Origen**	**Destino**	
64.864.125	02:00:00:00:8e:d7:28:1d	0x0000	IEEE 802.15.4
64.865.107			IEEE 802.15.4
65.443.113	0x0000		IEEE 802.15.4
66.050.023	0x0000		IEEE 802.15.4
66.077.446	0x0000	02:00:00:00:8e:d7:28:1d	LwMesh
66.078.427			IEEE 802.15.4

66.151.899	0x0001		IEEE 802.15.4
66.656.390	0x0000		IEEE 802.15.4
66.757.442	0x0001		IEEE 802.15.4
67.262.531	0x0000		IEEE 802.15.4
67.363.551	0x0001		IEEE 802.15.4
67.868.449	0x0000		IEEE 802.15.4
67.970.787	0x0001		IEEE 802.15.4
68.474.156	0x0000		IEEE 802.15.4
68.575.165	0x0001		IEEE 802.15.4
69.080.144	0x0000		IEEE 802.15.4
69.181.077	0x0001		IEEE 802.15.4
69.686.039	0x0000		IEEE 802.15.4
69.786.709	0x0001		IEEE 802.15.4
70.291.810	0x0000		IEEE 802.15.4
70.393.431	0x0001		IEEE 802.15.4
70.900.141	0x0000		IEEE 802.15.4
71.000.051	0x0001		IEEE 802.15.4
71.503.862	0x0000		IEEE 802.15.4
71.604.249	0x0001		IEEE 802.15.4
72.110.473	0x0000		IEEE 802.15.4
72.163.211	0x0000	0x0001	IEEE 802.15.4
72.164.190			IEEE 802.15.4
72.210.220	0x0001		IEEE 802.15.4
72.716.195	0x0000		IEEE 802.15.4
72.816.146	0x0001		IEEE 802.15.4
73.323.973	0x0000		IEEE 802.15.4
73.421.968	0x0001		IEEE 802.15.4
73.928.335	0x0000		IEEE 802.15.4
74.027.895	0x0001		IEEE 802.15.4
74.534.139	0x0000		IEEE 802.15.4
74.634.211	0x0001		IEEE 802.15.4
75.142.801	0x0000		IEEE 802.15.4
75.239.671	0x0001		IEEE 802.15.4
75.746.662	0x0000		IEEE 802.15.4

75.845.971	0x0001		IEEE 802.15.4
76.351.291	0x0000		IEEE 802.15.4
76.452.203	0x0001		IEEE 802.15.4
76.957.136	0x0000		IEEE 802.15.4
77.058.409	0x0001		IEEE 802.15.4
77.563.632	0x0000		IEEE 802.15.4
77.664.148	0x0001		IEEE 802.15.4
78.170.047	0x0000		IEEE 802.15.4
78.270.021	0x0001		IEEE 802.15.4
78.775.016	0x0000		IEEE 802.15.4
78.829.873	0x0000	0x0001	IEEE 802.15.4
78.829.873			IEEE 802.15.4
78.875.913	0x0001		IEEE 802.15.4
79.381.419	0x0000		IEEE 802.15.4
79.482.662	0x0001		IEEE 802.15.4
79.987.321	0x0000		IEEE 802.15.4
80.088.229	0x0001		IEEE 802.15.4
80.592.881	0x0000		IEEE 802.15.4
80.694.510	0x0001		IEEE 802.15.4
81.198.835	0x0000		IEEE 802.15.4
81.300.130	0x0001		IEEE 802.15.4
81.805.372	0x0000		IEEE 802.15.4
81.906.243	0x0001		IEEE 802.15.4
82.411.984	0x0000		IEEE 802.15.4
82.512.368	0x0001		IEEE 802.15.4
83.018.225	0x0000		IEEE 802.15.4
83.118.569	0x0001		IEEE 802.15.4
83.623.610	0x0000		IEEE 802.15.4
83.723.775	0x0001		IEEE 802.15.4
84.229.867	0x0000		IEEE 802.15.4
84.329.776	0x0001		IEEE 802.15.4
84.835.628	0x0000		IEEE 802.15.4
84.936.038	0x0001		IEEE 802.15.4
85.440.817	0x0000		IEEE 802.15.4

85.495.495	0x0000	0x0001	IEEE 802.15.4
85.495.495			IEEE 802.15.4
85.542.402	0x0001		IEEE 802.15.4
86.047.483	0x0000		IEEE 802.15.4
86.148.372	0x0001		IEEE 802.15.4
86.652.887	0x0000		IEEE 802.15.4
86.753.759	0x0001		IEEE 802.15.4
87.259.199	0x0000		IEEE 802.15.4
87.360.581	0x0001		IEEE 802.15.4
87.870.370	0x0000		IEEE 802.15.4
87.966.365	0x0001		IEEE 802.15.4
88.470.866	0x0000		IEEE 802.15.4
88.571.778	0x0001		IEEE 802.15.4
89.077.810	0x0000		IEEE 802.15.4
89.178.297	0x0001		IEEE 802.15.4
89.683.404	0x0000		IEEE 802.15.4
89.784.122	0x0001		IEEE 802.15.4
90.289.556	0x0000		IEEE 802.15.4
90.390.292	0x0001		IEEE 802.15.4
90.895.369	0x0000		IEEE 802.15.4
90.996.263	0x0001		IEEE 802.15.4
91.501.173	0x0000		IEEE 802.15.4
91.602.085	0x0001		IEEE 802.15.4
92.107.186	0x0000		IEEE 802.15.4
92.161.976	0x0000	0x0001	IEEE 802.15.4
92.161.976			IEEE 802.15.4
92.208.394	0x0001		IEEE 802.15.4
92.713.631	0x0000		IEEE 802.15.4
92.813.907	0x0001		IEEE 802.15.4
93.319.135	0x0000		IEEE 802.15.4
93.420.269	0x0001		IEEE 802.15.4
93.925.752	0x0000		IEEE 802.15.4
94.026.087	0x0001		IEEE 802.15.4
94.531.175	0x0000		IEEE 802.15.4

94.632.469	0x0001		IEEE 802.15.4
95.137.155	0x0000		IEEE 802.15.4
95.238.541	0x0001		IEEE 802.15.4
95.743.162	0x0000		IEEE 802.15.4
95.843.911	0x0001		IEEE 802.15.4
96.349.232	0x0000		IEEE 802.15.4
96.450.546	0x0001		IEEE 802.15.4
96.954.717	0x0000		IEEE 802.15.4
97.056.024	0x0001		IEEE 802.15.4
97.561.293	0x0000		IEEE 802.15.4
97.661.963	0x0001		IEEE 802.15.4
98.167.255	0x0000		IEEE 802.15.4
98.268.363	0x0001		IEEE 802.15.4
98.773.397	0x0000		IEEE 802.15.4
98.828.223	0x0000	0x0001	IEEE 802.15.4
98.829.202			IEEE 802.15.4
98.893.851	0x0001		IEEE 802.15.4
99.379.608	0x0000		IEEE 802.15.4
99.480.331	0x0001		IEEE 802.15.4
99.985.103	0x0000		IEEE 802.15.4
100.110.810	0x0001		IEEE 802.15.4
100.591.147	0x0000		IEEE 802.15.4
100.692.132	0x0001		IEEE 802.15.4
101.197.678	0x0000		IEEE 802.15.4
101.298.476	0x0001		IEEE 802.15.4
101.803.122	0x0000		IEEE 802.15.4
101.904.016	0x0001		IEEE 802.15.4
101.931.442	02:00:00:00:5d:f6:87:f7	0x0001	IEEE 802.15.4
101.931.442			IEEE 802.15.4
102.409.381	0x0000		IEEE 802.15.4
102.464.240	0x0001	0x0000	IEEE 802.15.4
102.465.221			IEEE 802.15.4
102.510.298	0x0001		IEEE 802.15.4
103.015.239	0x0000		IEEE 802.15.4

103.116.322	0x0001		IEEE 802.15.4
103.621.053	0x0000		IEEE 802.15.4
103.675.922	0x0000	0x0001	LwMesh
103.676.900			IEEE 802.15.4
104.227.243	0x0000		IEEE 802.15.4
104.328.173	0x0001		IEEE 802.15.4
104.355.597	0x0001	02:00:00:00:5d:f6:87:f7	LwMesh
104.356.577			IEEE 802.15.4
104.429.063	0x0002		IEEE 802.15.4
104.832.764	0x0000		IEEE 802.15.4
104.934.561	0x0001		IEEE 802.15.4
105.035.105	0x0002		IEEE 802.15.4
105.439.279	0x0000		IEEE 802.15.4
105.540.173	0x0001		IEEE 802.15.4
105.641.008	0x0002		IEEE 802.15.4
106.045.157	0x0000		IEEE 802.15.4
106.146.072	0x0001		IEEE 802.15.4
106.246.872	0x0002		IEEE 802.15.4
106.651.223	0x0000		IEEE 802.15.4
106.752.548	0x0001		IEEE 802.15.4
106.853.123	0x0002		IEEE 802.15.4
107.256.889	0x0000		IEEE 802.15.4
107.357.985	0x0001		IEEE 802.15.4
107.460.896	0x0002		IEEE 802.15.4
107.863.539	0x0000		IEEE 802.15.4
107.963.768	0x0001		IEEE 802.15.4
108.064.629	0x0002		IEEE 802.15.4
108.469.710	0x0000		IEEE 802.15.4
108.569.779	0x0001		IEEE 802.15.4
108.671.016	0x0002		IEEE 802.15.4
109.075.582	0x0000		IEEE 802.15.4
109.176.772	0x0001		IEEE 802.15.4
109.277.221	0x0002		IEEE 802.15.4
109.681.169	0x0000		IEEE 802.15.4

109.782.461	0x0001		IEEE 802.15.4
109.883.420	0x0002		IEEE 802.15.4
110.287.276	0x0000		IEEE 802.15.4
110.341.145	0x0000	0x0001	IEEE 802.15.4
110.342.129			IEEE 802.15.4
110.388.162	0x0001		IEEE 802.15.4
110.443.040	0x0001	0x0002	IEEE 802.15.4
110.443.040			IEEE 802.15.4
110.489.685	0x0002		IEEE 802.15.4
110.893.614	0x0000		IEEE 802.15.4
110.994.070	0x0001		IEEE 802.15.4
111.094.964	0x0002		IEEE 802.15.4
111.122.398	02:00:00:00:77:ff:3a:3d	0x0002	IEEE 802.15.4
111.498.948	0x0000		IEEE 802.15.4
111.600.544	0x0001		IEEE 802.15.4
111.653.810	0x0002	0x0001	IEEE 802.15.4
111.700.841	0x0002		IEEE 802.15.4
112.105.077	0x0000		IEEE 802.15.4
112.158.954	0x0001	0x0000	IEEE 802.15.4
112.205.970	0x0001		IEEE 802.15.4
112.307.463	0x0002		IEEE 802.15.4
112.710.906	0x0000		IEEE 802.15.4
112.812.736	0x0001		IEEE 802.15.4
112.913.045	0x0002		IEEE 802.15.4
113.316.963	0x0000		IEEE 802.15.4
113.372.777	0x0000	0x0001	LwMesh
113.373.756			IEEE 802.15.4
113.420.774	0x0001		IEEE 802.15.4
113.473.670	0x0001	0x0002	LwMesh
113.474.652			IEEE 802.15.4
113.519.822	0x0002		IEEE 802.15.4
113.923.368	0x0000		IEEE 802.15.4
114.024.447	0x0001		IEEE 802.15.4
114.125.422	0x0002		IEEE 802.15.4

114.152.850	0x0002	02:00:00:00:77:ff:3a:3d	LwMesh
114.153.831			IEEE 802.15.4
114.226.344	0x0003		IEEE 802.15.4

TABLA DE RESULTADOS PRUEBA DE NODO FALLIDO MEMSIC

El tiempo que tarda en restablecerse la comunicación en la red después de que un nodo falle.

Tabla 3.3. RESULTADOS PRUEBA DE NODO FALLIDO MEMSIC

TIEMPO (SEG)	SOURCE	DESTINO	TIEMPO DE REESTABLECIMIENTO (SEG)
93.737852	0X0007	0X0002	
96.909269	0X0007	0X0005	
96.909269	0X0005	0X0002	
97.096826	0X0005	0X0007	
97.847056	0X0005	0X0001	4.109204

TABLA DE PRUEBA DE NODO FALLIDO LIBELIUM

El tiempo que tarda en restablecerse la comunicación en la red después de que un nodo falle experimentalmente fue 19.23656 s. Lo que se hizo fue apagar uno de los nodos y esperar que la red envíe un dato por otra ruta. Para eso debemos percatarnos de el nodo origen al nodo destino.

Tabla 3.4. PRUEBA DE NODO FALLIDO LIBELIUM

Tiempo seg	Origen	Destino
899.550.054	0x0005	0x0004
899.900.668	0x0000	
900.001.395	0x0003	
900.055.161	0x0004	0x0003

900.102.226	0x0004	
900.203.379	0x0005	
900.505.657	0x0000	
900.607.115	0x0003	
900.709.109	0x0004	
900.809.319	0x0005	
901.111.688	0x0000	
901.165.691	0x0003	0x0000
901.212.994	0x0003	
901.313.907	0x0004	
901.414.636	0x0005	
901.717.776	0x0000	
901.819.161	0x0003	
902.021.307	0x0005	
902.324.503	0x0000	
902.378.132	0x0004	0x0003
902.425.219	0x0003	
902.478.981	0x0004	0x0003
902.479.941		
902.627.487	0x0005	
902.930.224	0x0000	
903.031.574	0x0003	
903.232.928	0x0005	
903.536.216	0x0000	
903.636.721	0x0003	
903.839.347	0x0005	
904.142.416	0x0000	
904.243.223	0x0003	
904.445.343	0x0005	
904.747.923	0x0000	
904.848.950	0x0003	
905.051.221	0x0005	
905.354.217	0x0000	
905.455.417	0x0003	

905.657.366	0x0005	
905.960.603	0x0000	
906.061.601	0x0003	
906.263.119	0x0005	
906.565.555	0x0000	
906.869.402	0x0005	
907.172.350	0x0000	
907.475.627	0x0005	
907.778.863	0x0000	
908.384.628	0x0000	
908.990.123	0x0000	
909.043.479	0x0000	0x0003
909.596.968	0x0000	
909.648.881	0x0000	0x0003
910.202.428	0x0000	
910.254.807	0x0000	0x0003
910.807.944	0x0000	
910.861.479	0x0000	0x0003
911.414.131	0x0000	
911.467.293	0x0000	0x0003
912.020.031	0x0000	
912.073.593	0x0000	0x0003
912.626.407	0x0000	
913.232.447	0x0000	
913.837.679	0x0000	
914.443.837	0x0000	
914.471.334	02:00:00:00:77:ff:3a:3d	0x0000
914.471.334		
915.049.710	0x0000	
915.656.378	0x0000	
915.683.179	0x0000	02:00:00:00:77:ff:3a:3d
916.262.127	0x0000	
916.288.649	0x0000	02:00:00:00:77:ff:3a:3d
916.868.437	0x0000	

916.895.459	0x0000	02:00:00:00:77:ff:3a:3d
917.474.213	0x0000	
917.500.845	0x0000	02:00:00:00:77:ff:3a:3d
918.079.658	0x0000	
918.107.342	0x0000	02:00:00:00:77:ff:3a:3d
918.685.720	0x0000	
918.786.614	0x0006	

TABLA DE RESULTADOS DE MEMSIC CON PROCESAMIENTO

En esta tabla se muestran los tiempos medidos entre que se envía una trama hasta que llega al siguiente nodo ahí se des encapsula se ingresa un dato se encapsula y se transmite desde el nodo que llegó hasta el siguiente, la diferencia de tiempos capturados por el Sniffer se presenta en la siguiente tabla.

Tabla 3.5. RESULTADOS DE MEMSIC CON PROCESAMIENTO

TIEMPO (SEG)	ORIGEN	DESTINO	TIEMPO DE PROCESAMIENTO (SEG)
21.475837	0X0007	0X0005	
21.491385	0X0005	0X0003	0.015548
21.491385	0X0003	0X0001	0
31.244268	0X0007	0X0005	
31.244268	0X0005	0X0003	0
31.244268	0X0003	0X0001	0
50.766372	0X0007	0X0005	
50.781498	0X0005	0X0003	0.015126
50.781498	0X0003	0X0001	0
60.528268	0X0007	0X0005	

60.543898	0X0005	0X0003	0.01563
60.543898	0X0003	0X0001	0
20.472708	0X0007	0X0003	
20.472708	0X0003	0X0001	0
30.245743	0X0007	0X0003	
30.245743	0X0003	0X0001	0
59.518545	0X0007	0X0003	
59.534173	0X0003	0X0001	0.015628
63.048333	0X0007	0X0005	
63.048333	0X0005	0X0003	0
63.048333	0X0003	0X0001	0
28.368157	0X0007	0X0003	
28.368157	0X0003	0X0002	0
28.398957	0X0002	0X0001	0.0308
		PROMEDIO	0.0154553

TABLA DE RESULTADOS LIBELIUM CON PROCESAMIENTO

En esta tabla se muestran los tiempos medidos entre que se envía una trama llega al siguiente nodo ahí se des encapsula se ingresa un dato se encapsula y se transmite desde el nodo que llega hasta el siguiente, la diferencia de tiempos capturados por el Sniffer se presenta en la siguiente tabla.

Tabla 3.6. RESULTADOS LIBELIUM CON PROCESAMIENTO

TIEMPO (SEG)	SOURCE	DESTINO	TIEMPO CON PROCESAMIENTO (SEG)

120.845294	0x0003	0x0002	0.505549
121.350843	0x0002	0x0001	0.504486
121.855329	0x0001	0x0000	
124.380526	0x0002	0x0001	0.505451
124.885977	0x0001	0x0000	
128.117573	0x0003	0x0002	0.504885
128.622458	0x0002	0x0001	0.505239
129.127697	0x0001	0x0000	
131.046312	0x0002	0x0001	0.504697
131.551009	0x0001	0x0000	
133.369922	0x0001	0x0000	
135.389257	0x0003	0x0002	0.524741
135.913998	0x0002	0x0001	0.484696
136.398694	0x0001	0x0000	
138.317711	0x0002	0x0001	0.505157
138.822868	0x0001	0x0000	
142.66154	0x0003	0x0002	0.5048
143.16634	0x0002	0x0001	0.504764
143.671104	0x0001	0x0000	
145.590231	0x0002	0x0001	0.504543
146.094774	0x0001	0x0000	
149.932935	0x0003	0x0002	0.505541
150.438476	0x0002	0x0001	0.50452
150.942996	0x0001	0x0000	

157.205213	0x0003	0x0002	
164.478373	0x0003	0x0002	
165.0835	0x0003	0x0002	0.506571
165.590071	0x0002	0x0001	0.503685
166.093756	0x0001	0x0000	
166.296519	0x0003	0x0002	
168.618341	0x0002	0x0001	0.50408
169.122421	0x0001	0x0000	
172.961374	0x0003	0x0002	0.505128
173.466502	0x0002	0x0001	0.504144
173.970646	0x0001	0x0000	
175.889605	0x0002	0x0001	0.504196
176.393801	0x0001	0x0000	
180.233165	0x0003	0x0002	1.109515
181.34268	0x0002	0x0001	0.505593
181.848273	0x0001	0x0000	
184.3738	0x0002	0x0001	0.504812
184.878612	0x0001	0x0000	
187.503912	0x0003	0x0002	
188.514264	0x0001	0x0000	
190.432896	0x0002	0x0001	0.505359
190.938255	0x0001	0x0000	
194.776621	0x0003	0x0002	0.50452
195.281141	0x0002	0x0001	0.516212

195.797353	0x0001	0x0000	
197.705156	0x0002	0x0001	0.505047
198.210203	0x0001	0x0000	
202.048507	0x0003	0x0002	0.504951
202.553458	0x0002	0x0001	0.518887
203.072345	0x0001	0x0000	
204.976995	0x0002	0x0001	0.505271
205.482266	0x0001	0x0000	
209.320127	0x0003	0x0002	0.50465
209.824777	0x0002	0x0001	0.50557
210.330347	0x0001	0x0000	
212.249189	0x0002	0x0001	0.505016
212.754205	0x0001	0x0000	
226.287723	0x0003	0x0002	
229.217431	0x0002	0x0001	0.504974
229.722405	0x0001	0x0000	
237.196597	0x0003	0x0002	
245.680767	0x0003	0x0002	
248.609317	0x0002	0x0001	0.50514
249.114457	0x0001	0x0000	
252.952235	0x0003	0x0002	0.504991
253.457226	0x0002	0x0001	0.504939
253.962165	0x0001	0x0000	
255.88122	0x0002	0x0001	0.504524

256.385744	0x0001	0x0000	
260.83048	0x0003	0x0002	0.504929
261.335409	0x0002	0x0001	0.504611
261.84002	0x0001	0x0000	
263.75939	0x0002	0x0001	0.503649
264.263039	0x0001	0x0000	
268.10222	0x0003	0x0002	0.50401
268.60623	0x0002	0x0001	0.505264
269.111494	0x0001	0x0000	
271.029885	0x0002	0x0001	0.505452
271.535337	0x0001	0x0000	
275.372936	0x0003	0x0002	
283.251443	0x0003	0x0002	0.505103
283.756546	0x0002	0x0001	0.50465
284.261196	0x0001	0x0000	
286.180194	0x0002	0x0001	0.505097
286.685291	0x0001	0x0000	
291.128743	0x0003	0x0002	0.505758
291.634501	0x0002	0x0001	0.504701
292.139202	0x0001	0x0000	
294.058334	0x0002	0x0001	0.50482
294.563154	0x0001	0x0000	
298.401374	0x0003	0x0002	0.504571
298.905945	0x0002	0x0001	0.505085

299.41103	0x0001	0x0000	
305.672996	0x0003	0x0002	0.504852
306.177848	0x0002	0x0001	0.504763
306.682611	0x0001	0x0000	
308.602674	0x0002	0x0001	
312.945641	0x0003	0x0002	
315.874209	0x0002	0x0001	0.50533
316.379539	0x0001	0x0000	
320.21711	0x0003	0x0002	0.505309
320.722419	0x0002	0x0001	0.505373
321.227792	0x0001	0x0000	
327.489801	0x0003	0x0002	
330.418009	0x0002	0x0001	0.505245
330.923254	0x0001	0x0000	
335.366928	0x0003	0x0002	
338.296279	0x0002	0x0001	0.504544
338.800823	0x0001	0x0000	
342.639957	0x0003	0x0002	
345.568416	0x0002	0x0001	0.504826
346.073242	0x0001	0x0000	
349.911375	0x0003	0x0002	
352.839963	0x0002	0x0001	0.504358
353.344321	0x0001	0x0000	
357.183096	0x0003	0x0002	0.503822

357.686918	0x0002	0x0001	0.505584
358.192502	0x0001	0x0000	
360.110934	0x0002	0x0001	0.505779
360.616713	0x0001	0x0000	
365.059578	0x0003	0x0002	0.505448
365.565026	0x0002	0x0001	0.504872
366.069898	0x0001	0x0000	
367.98918	0x0002	0x0001	0.505292
368.494472	0x0001	0x0000	
		PROMEDIO	0.51432748

TABLA DE RESULTADOS MEMSIC SIN PROCESAMIENTO

En esta tabla se muestran los tiempos medidos entre que se envía una trama llega al siguiente nodo y se reenvía desde el nodo que llego hasta el siguiente, la diferencia de tiempos capturados por el Sniffer se presenta en la siguiente tabla.

Tabla 3.7. RESULTADOS MEMSIC SIN PROCESAMIENTO

TIEMPO (SEG)	ORIGEN	DESTINO	TIEMPO (SEG)
0.010774	0x0005	BROADCAST	
0.012714	0X0004	0X0005	0.00194
0.017619	0x0005	BROADCAST	
0.021549	0X0004	0X0005	0.00393
0.023488	0x0005	BROADCAST	
0.027405	0X0004	0X0005	0.003917
0.040142	0x0005	BROADCAST	

0.0460017	0X0004	0X0005	0.0058597
0.051911	0x0005	BROADCAST	
0.055833	0X0004	0X0005	0.003922
0.058752	0x0005	BROADCAST	
0.06367	0X0004	0X0005	0.004918
0.068568	0x0005	BROADCAST	
0.070529	0X0004	0X0005	0.001961
0.07443	0x0005	BROADCAST	
0.078363	0X0004	0X0005	0.003933
0.080303	0x0005	BROADCAST	
0.083241	0X0004	0X0005	0.002938
0.091079	0x0005	BROADCAST	
0.093037	0X0004	0X0005	0.001958
0.095977	0x0005	BROADCAST	
0.098914	0X0004	0X0005	0.002937
0.100885	0x0005	BROADCAST	
0.103812	0X0004	0X0005	0.002927
0.105793	0x0005	BROADCAST	
0.10773	0X0004	0X0005	0.001937
0.11069	0x0005	BROADCAST	
0.115586	0X0004	0X0005	0.004896
0.122429	0x0005	BROADCAST	
0.12734	0X0004	0X0005	0.004911
0.132221	0x0005	BROADCAST	

0.13712	0X0004	0X0005	0.004899
0.141034	0x0005	BROADCAST	
0.142996	0X0004	0X0005	0.001962
29.296723	0x0005	BROADCAST	
29.2977	0X0004	0X0005	0.000977
29.30062	0x0005	BROADCAST	
29.302581	0X0004	0X0005	0.001961
29.306506	0x0005	BROADCAST	
29.308457	0X0004	0X0005	0.001951
			0.00323174

TABLA DE RESULTADOS LIBELIUM SIN PROCESAMIENTO

En esta tabla se muestran los tiempos medidos entre que se envía una trama llega al siguiente nodo y se reenvía desde el nodo que llego hasta el siguiente, la diferencia de tiempos capturados por el Sniffer se presenta en la siguiente tabla.

Tabla 3.8. RESULTADOS LIBELIUM SIN PROCESAMIENTO

TIEMPO(SEG)	ORIGEN	DESTINO	TIEMPO(SEG)
585.040372	0X0005	0X0004	0.503555
585.543927	0X0004	0X0003	
586.655859	0X0003	0X0000	
591.705715	0X0004	0X0005	0.605411
592.311126	0X0005	0X0004	0.505678
592.816804	0X0004	0X0003	0.494642
593.311446	0X0003	0X0000	

595.240216	0X0004	0X0003	0.505089
595.745305	0X0003	0X0000	
599.583695	0X0005	0X0004	0.504226
600.087921	0X0004	0X0003	0.505464
600.593385	0X0003	0X0000	
602.512647	0X0004	0X0003	0.505077
603.017724	0X0003	0X0000	
606.855221	0X0005	0X0004	
609.783989	0X0004	0X0003	0.50485
610.288839	0X0003	0X0000	
613.521131	0X0004	0X0005	
618.26847	0X0004	0X0003	0.606271
618.874741	0X0004	0X0003	0.50432
619.379061	0X0003	0X0000	
621.904755	0X0004	0X0003	0.504366
622.409121	0X0003	0X0000	
624.328284	0X0004	0X0003	
625.439423	0X0003	0X0000	
631.600276	0X0004	0X0003	0.505364
632.10564	0X0003	0X0000	
638.872924	0X0004	0X0003	0.504945
639.377869	0X0003	0X0000	
650.992277	0X0004	0X0003	0.505336
651.497613	0X0003	0X0000	

653.416428	0X0004	0X0003	0.505059
653.921487	0X0003	0X0000	
658.264166	0X0004	0X0003	0.505229
658.769395	0X0003	0X0000	
660.688702	0X0004	0X0003	0.504999
661.193701	0X0003	0X0000	
665.536205	0X0004	0X0003	0.505154
666.041359	0X0003	0X0000	
668.565948	0X0004	0X0003	0.504489
669.070437	0X0003	0X0000	
675.838681	0X0004	0X0003	0.503556
676.342237	0X0003	0X0000	
680.685305	0X0004	0X0003	0.50535
681.190655	0X0003	0X0000	
687.4527	0x0005	0x0004	0.504296
687.956996	0x0004	0x0003	0.504894
688.46189	0X0003	0X0000	
694.723955	0x0005	0x0004	
695.12859	0X0003	0X0000	
702.500939	0x0004	0x0003	
703.612579	0X0003	0X0000	
706.137266	0x0004	0x0003	0.50462
706.641886	0X0003	0X0000	
709.268247	0x0005	0x0004	0.505084

709.773331	0x0004	0x0003	
710.883946	0X0003	0X0000	
714.015299	0x0004	0x0003	0.504516
714.519815	0X0003	0X0000	
720.680991	0x0004	0x0003	0.504707
721.185698	0X0003	0X0000	
724.417671	0x0005	0x0004	0.505973
724.923644	0x0004	0x0003	
725.427763	0X0003	0X0000	
727.347323	0x0004	0x0003	0.505214
727.852537	0X0003	0X0000	
731.690634	0x0005	0x0004	0.504514
732.195148	0x0004	0x0003	0.50514
732.700288	0X0003	0X0000	
739.467165	0x0004	0x0003	0.50512
739.972285	0X0003	0X0000	
741.891572	0x0004	0x0003	0.505038
742.39661	0X0003	0X0000	
746.739336	0x0004	0x0003	
747.850646	0X0003	0X0000	
750.982133	0x0004	0x0003	0.504556
751.486689	0X0003	0X0000	
754.1122	0x0005	0x0004	
755.122133	0X0003	0X0000	

758.253458	0x0004	0x0003	0.505114
758.758572	0X0003	0X0000	
775.928021	0x0005	0x0004	0.504268
776.432289	0x0004	0x0003	0.504789
776.937078	0X0003	0X0000	
783.704954	0x0004	0x0003	
784.815362	0X0003	0X0000	
786.733789	0x0004	0x0003	0.505636
787.239425	0X0003	0X0000	
790.471686	0x0005	0x0004	0.504782
790.976468	0x0004	0x0003	0.504927
791.481395	0X0003	0X0000	
793.399811	0x0004	0x0003	0.505209
793.90502	0X0003	0X0000	
798.248477	0x0004	0x0003	0.504464
798.752941	0X0003	0X0000	
801.277883	0x0004	0x0003	0.504787
801.78267	0X0003	0X0000	
805.520243	0x0004	0x0003	
806.631619	0X0003	0X0000	
808.550248	0x0004	0x0003	0.505296
809.055544	0X0003	0X0000	
812.287573	0x0005	0x0004	0.504709
812.792282	0x0004	0x0003	0.505333

813.297615	0X0003	0X0000	
815.215871	0x0004	0x0003	0.505645
815.721516	0X0003	0X0000	
819.559124	0x0005	0x0004	0.50523
820.064354	0x0004	0x0003	0.505182
820.569536	0X0003	0X0000	
822.488271	0x0004	0x0003	0.504683
822.992954	0X0003	0X0000	
826.830899	0x0005	0x0004	0.505039
827.335938	0x0004	0x0003	0.504938
827.840876	0X0003	0X0000	
829.760427	0x0004	0x0003	0.504748
830.265175	0X0003	0X0000	
834.10278	0x0005	0x0004	0.50584
834.60862	0x0004	0x0003	0.504756
835.113376	0X0003	0X0000	
837.032269	0x0004	0x0003	0.50511
837.537379	0X0003	0X0000	
841.375756	0x0005	0x0004	
842.385663	0X0003	0X0000	
844.304274	0x0004	0x0003	0.505138
844.809412	0X0003	0X0000	
848.647113	0x0005	0x0004	0.505183
849.152296	0x0004	0x0003	0.504612

849.656908	0X0003	0X0000	
851.5762	0x0004	0x0003	0.504988
852.081188	0X0003	0X0000	
856.42405	0x0004	0x0003	0.506021
856.930071	0X0003	0X0000	
858.847962	0x0004	0x0003	0.504198
859.35216	0X0003	0X0000	
863.191249	0x0005	0x0004	0.505186
863.696435	0x0004	0x0003	0.504152
864.200587	0X0003	0X0000	
866.119381	0x0004	0x0003	0.504937
866.624318	0X0003	0X0000	
870.462726	0x0005	0x0004	0.504481
870.967207	0x0004	0x0003	0.505037
871.472244	0X0003	0X0000	
873.391308	0x0004	0x0003	0.504538
873.895846	0X0003	0X0000	
880.663075	0x0004	0x0003	0.504602
881.167677	0X0003	0X0000	
885.006378	0x0005	0x0004	0.505153
885.511531	0x0004	0x0003	0.505033
886.016564	0X0003	0X0000	
887.935201	0x0004	0x0003	0.504834
888.440035	0X0003	0X0000	

892.278817	0x0005	0x0004	0.50424
892.783057	0x0004	0x0003	0.505109
893.288166	0X0003	0X0000	
895.207227	0x0004	0x0003	0.505188
895.712415	0X0003	0X0000	
899.550054	0x0005	0x0004	0.505107
900.055161	0x0004	0x0003	
901.165691	0X0003	0X0000	
		PROMEDIO	0.50740681

CUADRO COMPARATIVO DE TIEMPOS PRÁCTICOS Y TEÓRICOS.

Para calcular el retardo por procesamiento en el nodo se tomará los siguientes parámetros.

Las medidas se hicieron con procesamiento y sin procesamiento que representa medir el tiempo que demora en transmitir.

Tabla 3.9. Valores dados por el estándar [27]

TBOslot	0.00096s
IFS (SIFS oTIFS)	0.000192
T_{CCA}	0.000128
T_{TA}	0.000192
T_{ACK}	0.000352
Valores dedo por el estandar	

Utilizando la ecuación 1 con x = 12 bytes se tiene la siguiente tabla

Tabla 3.10. Cuadro comparativo de tiempos prácticos y teóricos MEMSIC

Tframe(12)	T_{nrr} calculado	T_{nrr} Medido sin procesamiento mseg	T_{nrr} Medido con procesamiento mseg
0,992	2,62	3.2	15.4

En el modo ranurado utilizando la ecuación con x igual 16 adicionalmente se considera

T_i = 40 mseg T_{SD} = 140 mseg

Tabla 3.11. Cuadro comparativo de tiempos prácticos y teóricos LIBELIUM

Ttrama(16)	T_{rr} calculado	T_{rr} medido con procesamiento	T_{rr} medido sin procesamiento
1.12 mseg	182.25mseg	514 mseg	507mseg

DISCUSIÓN DE LOS RESULTADOS

RESULTADOS DEL RETARDO EN EL NODO EN MODO NO BEACON

Como puede apreciar los valores calculados son menores que los medidos, el valor medido del retardo sin procesamiento tiene un porcentaje de error del 22 %, una de las razones es porque en condiciones ideales, el nodo debería recibir el mensaje en el buffer de recepción y automáticamente reenviarlo, en la practica la trama es almacenada en el buffer de recepción, el programa implementado la recupera y por programa se la envía nuevamente. Consideramos que este proceso es el causante del valor mayor obtenido el momento de la medición

En el caso del retardo con procesamiento, debido a la utilización del protocolo 6LowPan en el nodo es evidente que el procesamiento de la cabecera de 6LowPan, tiene como consecuencia el aumento del retardo por procesamiento en el nodo, y la diferencia es del 487 %. Por lo que es importante disminuir los procesos que se realizan en el nodo para disminuir el retardo

RESULTADO DE VALORES DE RETARDO EN EL MODO RANURADO

En este caso, cuando el recibe una trama en el intervalo de tiempo asignado, el nodo tiene que esperar un tiempo adicional dado por T_{sd} y T_i para poder retransmitirla, en este caso el tiempo de procesamiento adicional está relacionado con el tiempo requerido para acceder

a la siguiente ranura. La diferencia entre el valor medido sin procesamiento en el nodo es del 182% y la diferencia con el valor medido con procesamiento es del 178%

Como se puede apreciar los porcentajes son prácticamente iguales, y la razón es porque, en tiempo de duración de una ranura es el mismo cuando se mide el retardo con procesamiento y sin procesamiento. Y a pesar de que sin procesamiento se tenga lista la trama para ser transmitida, sin embargo, tiene que esperar el siguiente intervalo de tiempo.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

Se puede concluir que los resultados obtenidos en la realidad son mayores que los resultados calculados en la teoría, debido a los procesos que se realizan en los nodos y eso dependerá de la tecnología que estemos ocupando en lo que tiene que ver con los retardos en los nodos.

Es importante entender que en el modo ranurado los tiempos aumentan por lo que las ranuras de tiempo hacen que los nodos transmitan en ciertos slots haciendo que exista espacios específicos para transmitir si por alguna razón no se pudo enviar la información se tiene que esperar a que nuevamente el canal se encuentre libre.

Para el estudio de esto se tuvo que considerar tiempos de retardo por transmisión muy cortos debido que a la distancia que se encontraban los nodos el tiempo que demora la señal en llegar de un punto a otro en realidad es muy pequeño considerando la velocidad de transmisión con la cual se está trabajando.

Los tiempos de establecimiento de la red dependen de las distancias a las cuales, se encontraban los nodos, debido a que, al momento de encender los nodos, el tiempo que me demoraba en trasladarme de un nodo a otro nodo influyen en el tiempo que demora en establecerse la red, es por eso que en los nodos Libelium el tiempo es mayor ya que los nodos fueron colocados a una distancia mayor que los nodos Memsic.

Se pudo observar que al utilizar este tipo de topología se garantiza la confiabilidad de la red, ya que al cambiarse de ruta al momento de que exista un fallo en algún nodo, el tiempo de restablecimiento de la red por la caída de un nodo no es alto y así podemos hacer llegar la información a más largas distancias.

Se pudo comprobar que el modo ranurado tiene un tiempo mayor de demora en transmitir un dato y retransmitir información a comparación de los nodos que ocupan el modo no ranurado. Esta diferencia es por lo que esperan una ranura de tiempo para poder transmitir.

4.2 RECOMENDACIONES

Al momento de realizar la prueba es recomendable que todos los nodos se encuentren con la misma carga en las baterías para evitar considerar el efecto que ocasione que las baterías de los nodos se encuentren descargadas.

Al realizar las pruebas con los nodos se debe considerar la línea de vista para las antenas de los sensores, es recomendable que los nodos estén en línea recta par que la medida de los tiempos sea tomada de mejor manera, además de que se deben encontrar a la misma altura, para obtener mejores medidas de los datos.

Es recomendable tomar en cuenta el radio de cobertura de cada uno de los sensores para establecer de mejor manera la zona de cobertura y considerar el parámetro de la mitad de la distancia del radio para que de esa manera se puedan distribuir los sensores de manera adecuada y puedan trabajar en la topología lineal que hemos planteado.

Se puede recomendar que si deseamos tener una eficiencia mejor en la comunicación se ocupen nodos que trabajen en modo ranurado que a pesar de que los tiempos son mayores en relación con el modo no ranurado, este esquema previene que existan más colisiones.

5 REFERENCIAS BIBLIOGRÁFICA

[1] A. B. Corral Ignoto , "Diseño e implementación de un entorno de simulación para redes de sensores inalámbricos," Universidad Politécnica de Cartagena, 2005.

[2] C. Aranzazu Suescún y G. A. Moreno López," Revisión del estado del arte de redes de sensores inalámbricas," *Revista Politécnica ,* vol. 5, nº 8, pp. 94-111, 2009.

[3] F. Ortiz Tapia , “Redes de sensores inalámbricos,” 2006.

[4] S. E. Castillo Rodríguez , “Análisis de tecnologías WiFi y ZigBee que optimice las comunicaciones inalámbricas para el control de temperatura de un invernadero,” Escuela Superior Politécnica de Chimborazo , Riobamba, 2012.

[5] P. Magaña Espinoza , “Diseño, desarrollo e implementación de un protocolo de comunicaciones multimétrica para redes inalámbricas de sensores,” Universidad de Colima, México , 2017.

[6] A. Koubaa, M. Alves y E. Tovar , IEEE 802.15.4 for Wireless Sensor Networks: A Techinical Overview, IPP Hurray, 2005.

[7] I. Tustón Torres, “Evaluación del rendimiento del Estándar IEEE 802.15.4 (ZIGBEE) en entornos de interferencia.,” Escuela Superior Politécnica de Chimborazo, 2011.

[8] J. X. Arciniega Barahona , “Diseño e implementación de una herramienta de monitoreo y captura de tramas en redes IEEE 802.15.4,” Escuela Politécnica Nacional, Quito, 2016.

[9] F. D. Cali Reyes , “Implementación del algoritmo de protocolo de direccionamiento para redes de sensores inalámbricos con el estándar IEEE 802.15.4,” Escuela Politécnica Nacional, Quito, 2018.

[10] V. Alcázar , “Estudio de las prestaciones de IEEE 802.15.4e,” Universidad Politécnica de Catalunya, 2013.

[11] J. P. Dignani, “Análisis del protocolo ZigBee,” Universidad Nacional de la Plata, 2012.

[12] M. R. Modesti, D. O. Tanburi, A. Dimas y L. Plata , “Protocolo de ruteo adaptable para redes inalámbricas de sensores,” no. 1, 2009.

[13] J. S. Paz , “Diseño y construcción de un módulo transmisor receptor inalámbrico para el manejo de sensores con el protocolo estándar IEEE 802.15.4 (ZigBee),” Universidad Tecnológica de la Mixteca , 2008.

[14] IEEE Computer Society, “Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs),” New York , 2006.

[15] Z. Shelby y C. Bormann, 6LoWPAN: the wireless embedded internet, John Wiley & Sons, 2009.

[16] V. Garg, Wireless Communications & Networking., Morgan Kaufmann, 2010.

[17] L. Jiménez Ruiz , “Diseño e implementación de etapas de comunicación basada en 6LowPAN para plataforma modular de redes de sensores inalámbricos.,” Escuela Técnica Superior de Ingenieros Industriales (UPM), 2016.

[18] N. Kushalnagar, G. Montenegro y C. Schumacher, “IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs),” RFC 4919, 2007.

[19] Libelium comunicaciones Distribuidas S.L., Waspmote mote runner technical Guide wasp mote, Tech. Guid. , 2015, p. 85.

[20] Anónimo, “FAQ | MonoDevelop,” [En línea]. Available: https://www.monodevelop.com/help/faq/. [Último acceso: 4 09 2021].

[21] D. J. Ledkov, “LTS - Ubuntu Wiki.,” [En línea]. Available: https://wiki.ubuntu.com/LTS. [Último acceso: 30 08 2021].

[22] Canonical Ltd, “1.1. What is Ubuntu?,” [En línea]. Available: https://help.ubuntu.com/lts/installation-guide/s390x/ch01s01.html. [Último acceso: 27 08 2021].

[23] J. Wunsch, “Avrdude(1) - Linux man page.,” [En línea]. Available: https://linux.die.net/man/1/avrdude. [Último acceso: 02 09 2021].

[24] Anónimo, “AVRDUDE - AVR Downloader/UploaDEr.,” [En línea]. Available: http://www.nongnu.org/avrdude/. [Último acceso: 02 09 2021].

[25] “Wireshark User's Guide - v1.11.4-rc1-54-g9496733 for Wireshark 1.11 - user-guide-a4.pdf,” [En línea]. Available: http://download2.upload.de/software/41563/14/user-guide-a4.pdf. [Último acceso: 21 07 2021].

[26] B. Latré, P. Mil, I. Moerman, B. Dhoedt, P. Demeester y N. Dierdonck, “Throughput and Delay Analysis of Unslotted IEEE 802.15.4,” *Journal of Networks,* vol. 1, nº 1, 2006.

[27] NXP SEMICONDUCTORS, “JN-AN-1035 Calculating 802-15-4 Data Rates,” nº 1.4, p. 12, 2020.

[28] Prismmodelchecker, “IEEE 802.15.4 CSMA-CA Protocol (ZigBee),” 2011. [En línea]. Available: http://www.prismmodelchecker.org/casestudies/zigbee.php. [Último acceso: 17 09 2021].

[29] S. R. Caprile , Equisbí : desarrollo de aplicaciones con comunicación remota basadas en módulos ZigBee y 802.15.4, 1a edición ed., Buenos Aires: Gran Aldea Editores - GAE, 2009.

ANEXOS

ANEXO A. MANUAL DE INSTALACÓN MOTE RUNNER.

ANEXO B. ELEMENTOS DEL KIT DE DESARROLLO MENSIC Y PROCESO DE INSTALACIÓN DE LOS SENSORES PARA SU CORRECTO USO.

ANEXO A

MANUAL DE INSTALACIÓN MOTE RUNNER

ARCHIVOS NECESARIOS PARA LA INSTALACIÓN:

- UBUNTU 12,04 LTS
- JAVA JRE/SDK :
- SOFTWARE MONO DEVELOP
- FIREFOX
- SOFTWARE MOTERUNNER IBM:

INSTALACIÓN UBUNTU 12,04 LTS

Ingresar al *BIOS* para seleccionar el booteo del dvd. Se aconseja que se ingrese en “algo más” en caso de tener el sistema operativo Windows en la computadora para ver las particiones, estas se crearán a gusto y necesidad del usuario, pero se recomienda mínimo para Linux 100GB.

- Si no se asigna un disco fat32 no importa, ya que de esa partición se encargará Windows.

- Una vez finalizada la instalación del sistema operativo Ubuntu 12.04LTS, se debe ingresar a la "Terminal”.

- Cambiar al modo de súper usuario y ejecutar los siguientes comandos provistos en Código A-1, previos a la instalación de *Moterunner*

```
ubuntu@ubuntu:$ sudo su
[sudo] password for ubuntu:
root@ubuntu:~# sudo apt-get update
root@ubuntu:~# sudo apt-get upgrade
root@ubuntu:~# sudo apt-get dist-upgrade
root@ubuntu:~# sudo apt-get install aptitude
root@ubuntu:~# sudo aptitude full-upgrade
```

Código A-1 Instalaciones previas a instalar Moterunner

INSTALACIÓN DE COMPILADORES DE JAVA JRE/SDK

Para la instalación de estos elementos se puede realizar de dos maneras:

a. Mediante el link oficial de descarga de Oracle que se presenta a continuación:

En la siguiente dirección se descarga el JDK:

- http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html.

En la siguiente dirección se descarga el JRE:

- http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html.

b. La segunda opción es ejecutar los comandos como se muestra en el Código A-2

```
root@ubuntu:~# sudo apt-get update
root@ubuntu:~# sudo apt-get upgrade
root@ubuntu:~# sudo apt-get install aptitude
root@ubuntu:~# sudo aptitude full-upgrade
root@ubuntu:~# sudo apt-get install icedtea-7-plugin openjdk-7-
jre
root@ubuntu:~# sudo apt-get install openjdk-7-jdk
```

Código A-2 Actualizaciones previas e instalación de Java

Para verificar que se han instalado satisfactoriamente los paquetes se ejecutan en la terminal los siguientes comandos:

```
root@ubuntu:~# java -version
```

Código A-3 Comprobación de la instalación de Java

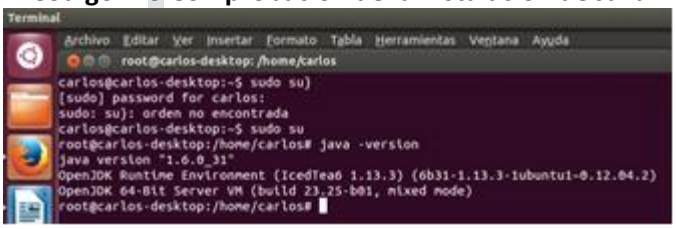

Figura A-1 Comprobación de la versión de Java

```
root@ubuntu:~# javac -version
```

Código A-4 Comprobación de Java

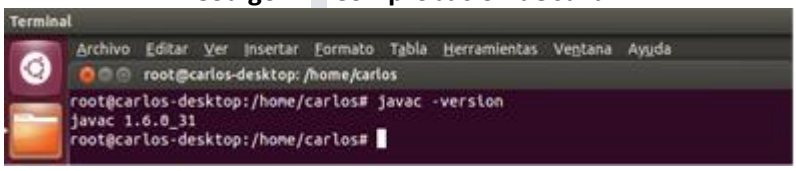

Figura A-2 Comprobación de la versión de Java

SOFTWARE MONO DEVELOP

Para la instalación de este *software*, que permitirá desarrollar aplicaciones en *C#*, se puede adquirir o descargar desde el Centro de software de Ubuntu, como se indica en la Figura A-3.

Figura A-3 Instalación del software MonoDevelop

Además, es muy importante instalar además estos compiladores de C# para poder compilar código desarrollado para las aplicaciones.

El programa que se presenta en el Código A-5 es un compilador para C#.

```
root@ubuntu:~# sudo apt-get install mono-mcs
```

Código A-5 Instalador del compilador para C#

El programa que se presenta en el Código A-6 es un compilador para archivos binarios.

```
root@ubuntu:~# sudo apt-get install chicken-bin
```

Código A-6 Instalador de un compilador de archivos binarios

SOFTWARE MOTERUNNER

Para su instalación es necesario descargar desde la siguiente dirección, dependiendo la distribución que se dispone de Ubuntu, sea de 32 o 64 bits. Para lo cual en la dirección indicada posteriormente se deberá aceptar las licencias presentadas para poder acceder a la descarga:

http://www.zurich.ibm.com/moterunner/downloads.html

Descargar la versión 13.1 más reciente que solo está disponible para sistemas Linux de 64 bits por lo cual se recomienda obligatoriamente tener el Sistema Operativo de 64 bits, ya que la versión 13.1 maneja el hardware de los *Waspmote* de *Libelium* y como se mencionó anteriormente solo está disponible para la versión de 64 bits.
Una vez que se haya descargado, se procede a realizar lo que se muestra a continuación:

- Se descomprime el archivo descargado.
- Se ejecuta el archivo *setup* que está en la carpeta descomprimida, mediante la terminal se debe dirigir a la ubicación de dicha carpeta. Se encontrará tres archivos donde uno es el setup, para ejecutarlo se debe ejecutar el siguiente comando: ***sudo ./setup*** dentro de la carpeta que se encuentra el archivo ejecutable**.**
- Seguir los pasos indicados en el terminal, en los cuales se pedirá aceptar la licencia para la instalación de *Moterunner.*

Al finalizar se creará una carpeta llamada "moterunner" en la carpeta anteriormente descomprimida, la cual se procederá a mover a los archivos raíz del sistema con el siguiente comando:

```
1  root@ubuntu:~# cp -R "dirección completa de nuestra carpeta actual" /
```

Código A-7 Copia de la carpeta Moterunner al archivo raíz del sistema "/"

A continuación, se presenta un ejemplo de la copia de la carpeta *Moterunner*.

```
1  root@ubuntu:~#cp -R /home/Escritorio/moterunner-install-
   temp/moterunner /
```

Código A-8 Ejemplo de copia de la carpeta Moterunner en el archivo raíz "/"

- Exportación de las variables de entorno del sistema mediante los siguientes comandos en la terminal.

Esto permitirá utilizar *Moterunner Shell* "*mrsh*", el cual inicializará el servidor y el compilador *mrc*. Con la exportación de estos *path* o variables de entorno se podrá ejecutar estos comandos desde cualquier directorio que se encuentre el usuario, como se indica a continuación.

Linux (32bits)

```
1  root@ubuntu:~#export PATH=/moterunner/linux/bin:$PATH
2  root@ubuntu:~#export
   LD_LIBRARY_PATH=/moterunner/linux/bin:$LD_LIBRARY_PATH
```

Código A-9 Exportación de variables de entorno para 32 bits

Linux (64bits)

```
1  root@ubuntu:~#export PATH=/moterunner/linux64/bin:$PATH
2  root@ubuntu:~#export
   LD_LIBRARY_PATH=/moterunner/linux64/bin:$LD_LIBRARY_PATH
```

Código A-10 Exportación de variables de entorno para 64 bits

Para la exportación de estas variables de entorno y la ejecución del servidor "*mrsh*" se ha realizado un pequeño Script llamado "*ExportMoterunner.sh*", es importante la extensión *.sh,* que permitirá ejecutar el mismo y además debe tener todos los permisos para su ejecución, en el script se tiene lo siguiente:

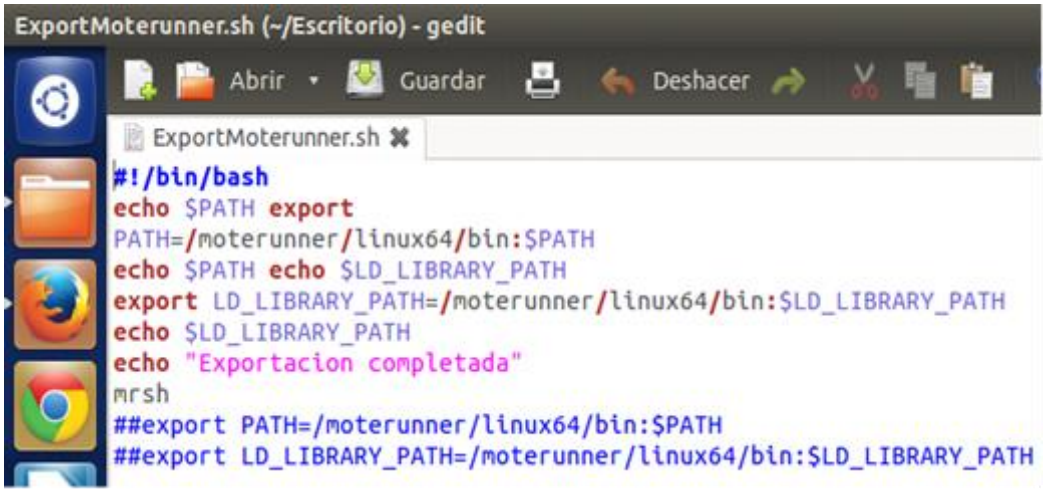

Figura A-4 Script para exportación de las variables de entorno y ejecución del servidor Moterunner Shell MRSH

Para su comprobación se ingresa la siguiente dirección web *http://localhost:5000* en el navegador web *Mozilla.*

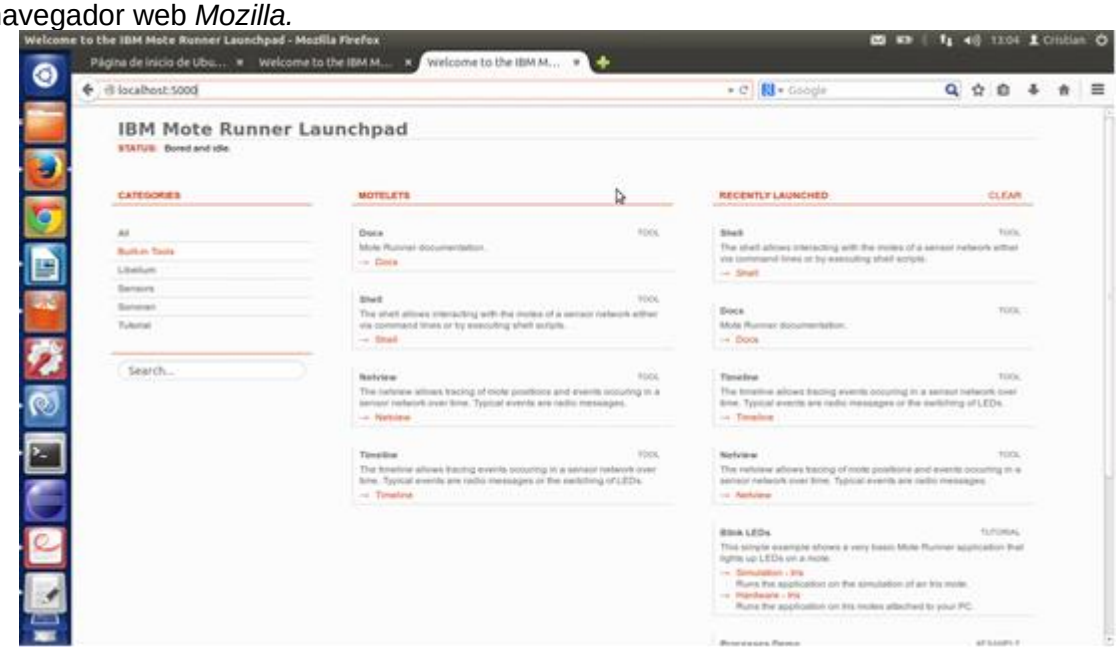

Figura A-5 Servidor MRSH

AÑADIR LAS LIBRERIAS DEL MOTORUNNER EN MONODEVELOP

Se crea una nueva solución con los siguientes pasos:

- Archivo
- Nuevo
- Solución

En ubicación se selecciona en que directorio de nuestro PC se desea crear la solución, que para estas aplicaciones es una librería como se puede apreciar en la siguiente Figura A-6.

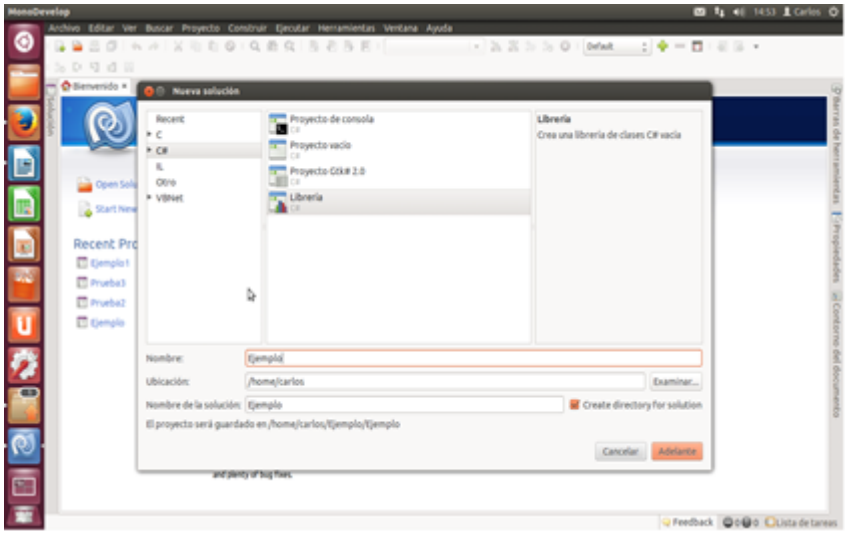

Figura A-6 Creando una solución en MonoDevelop

Se integra el soporte GTK# y la Integración con Unix.

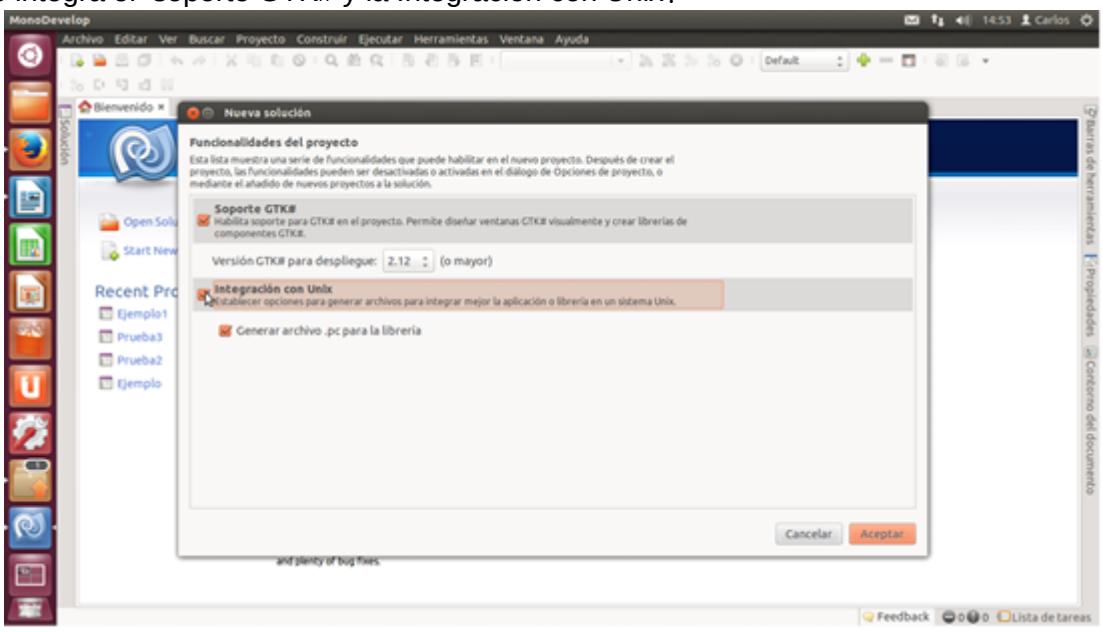

Figura A-7 Integración de GTK# y Unix

Se presenta la librería de la clase que se ha creado.

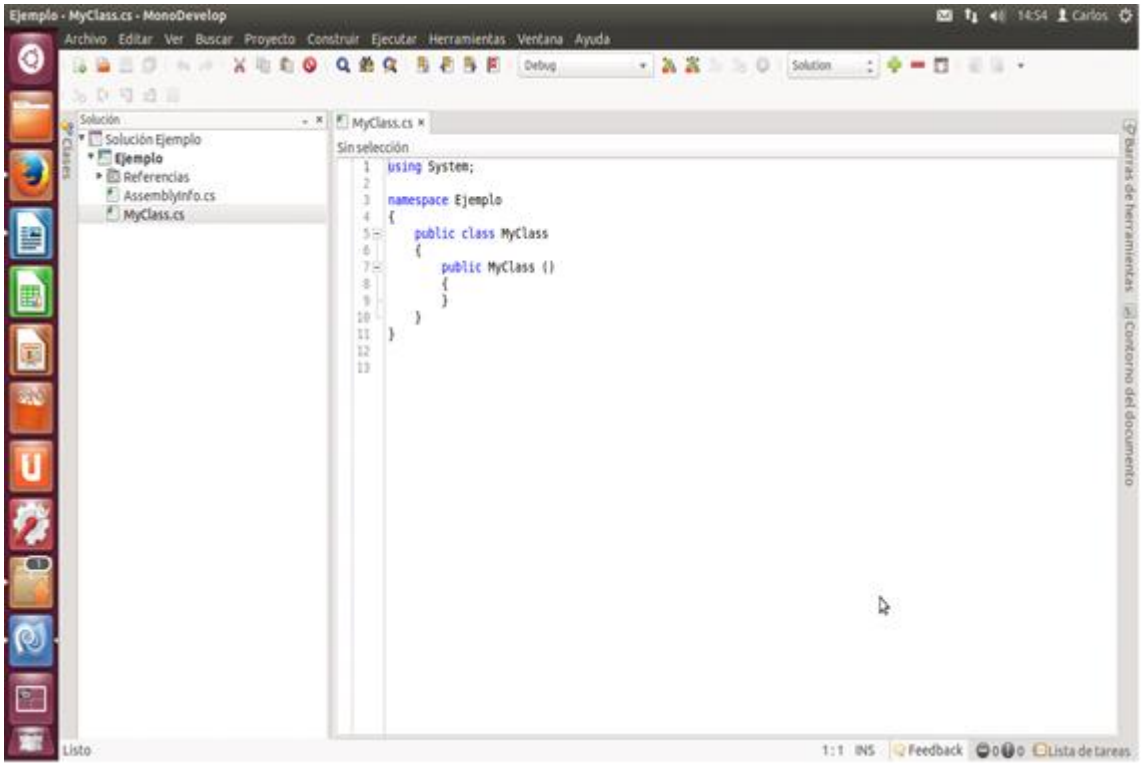

Figura A-8 Creación de la clase

Para agregar las librerías se debe dar click derecho en Referencias.

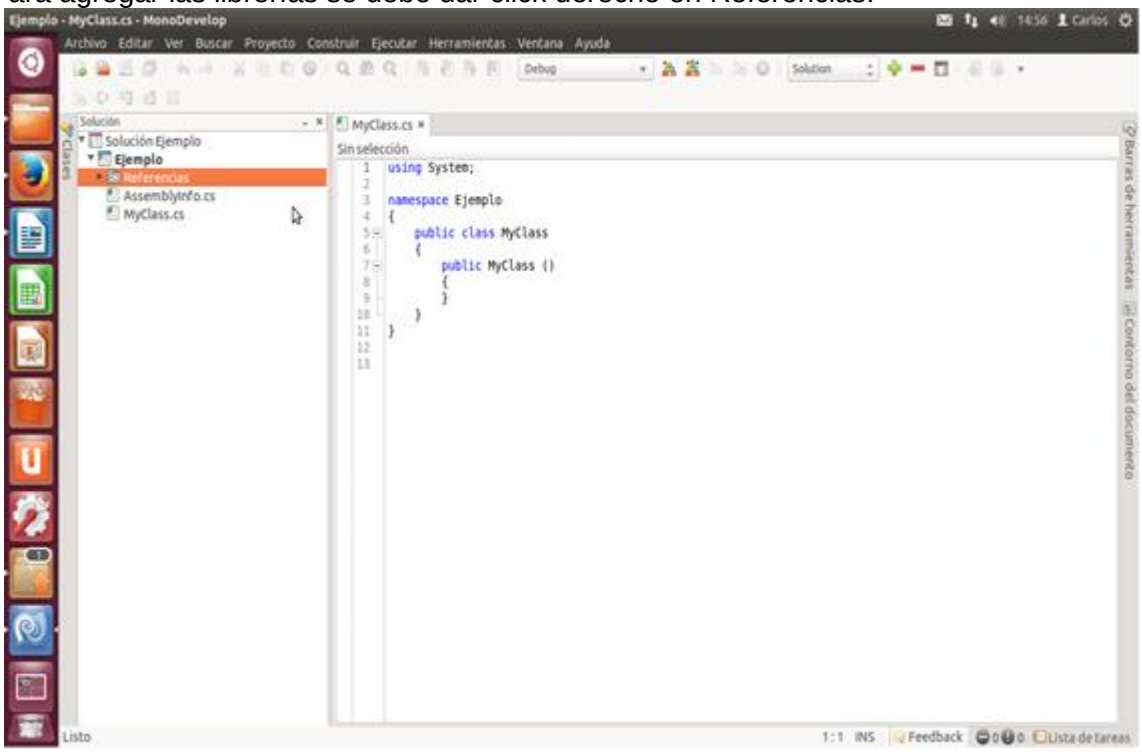

Figura A-9 Añadiendo las referencias de Moterunner

Se selecciona "*editar referencias*".

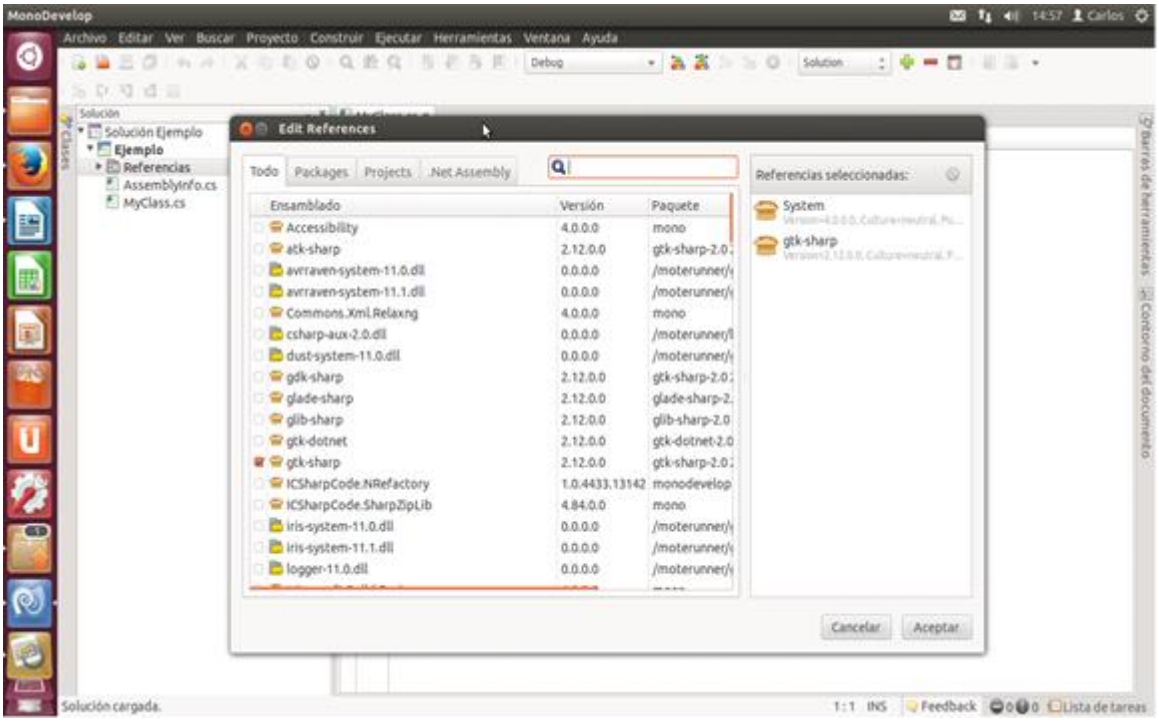

Figura A-10 Añadir referencias

Se selecciona la pestaña "*.Net Assembly"*.

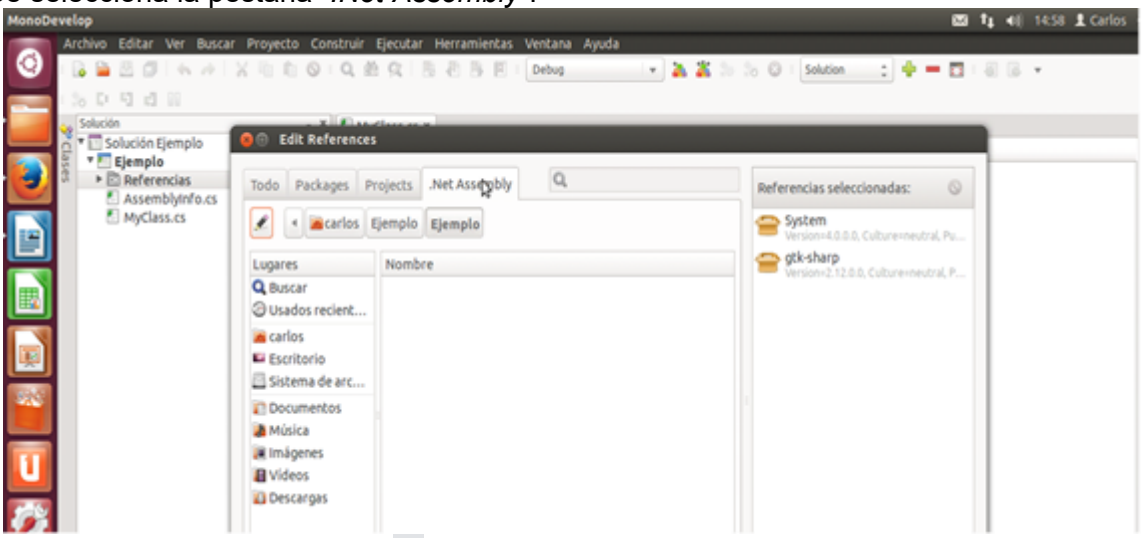

Figura A-11 Añadiendo librerías de Moterunner

Se busca el directorio en donde está instalado el moterunner "*/moterunner*".
Se busca la carpeta *gac* y se selecciona todas las librerías cuya extensión son *.dll*.
Es importante indicar que si hay dos librerías con el mismo nombre, seleccionar la librería con el valor más alto, ya que como se puede observar en la Figura A- 12, dichas librerías poseen número de *assemblies*.
Si seleccionamos varios assemblies con el mismo nombre van a existir conflictos problemas al momento del desarrollo de la aplicación.

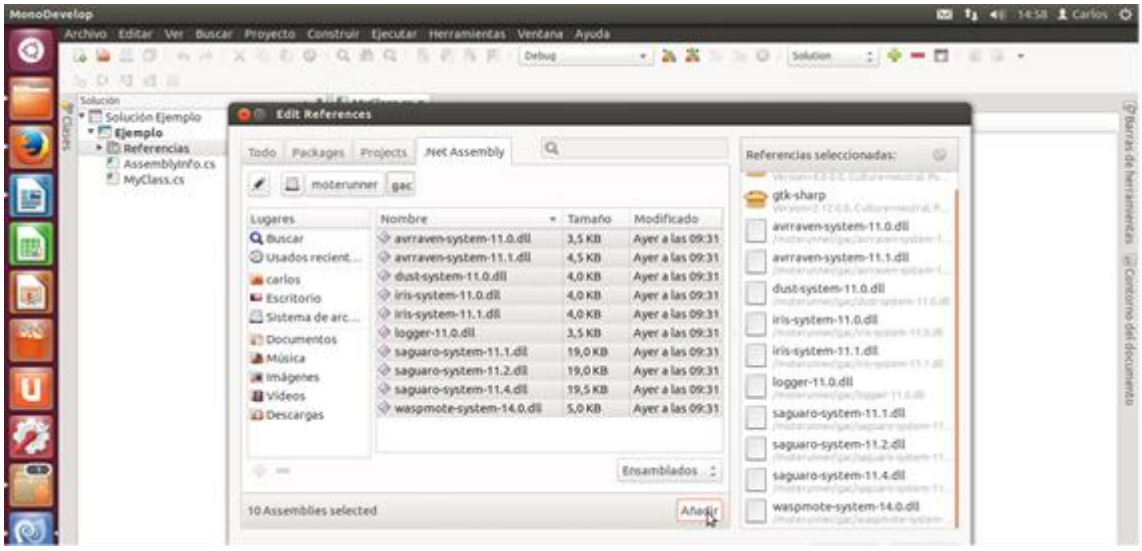

Figura A-12 Insertando librerías de Moterunner de la librería gac

Se busca la carpeta *lib* y se añade las librerías cuya extensión es *.dll*.
De la misma forma solo seleccionar las librerías con el número más alto, en el caso de que existan dos con el mismo nombre.

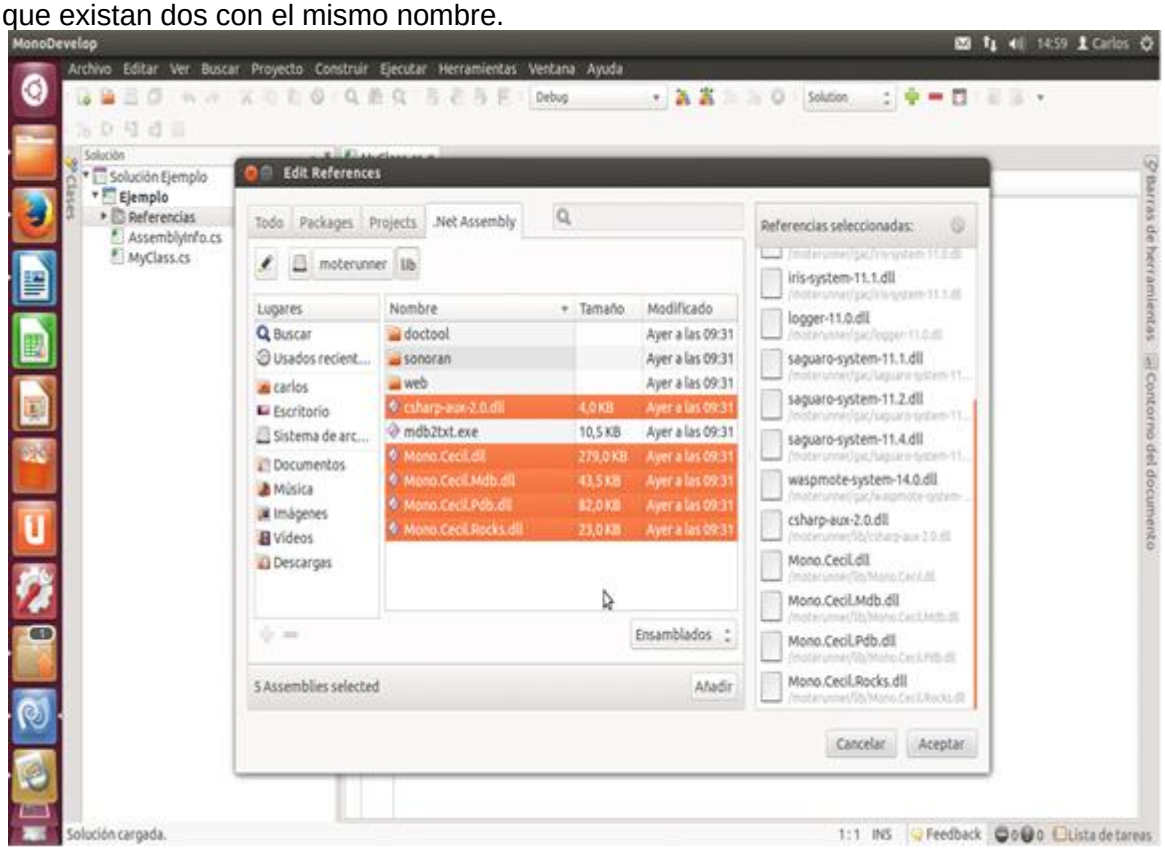

Figura A-13 Insertando librerías de la carpeta lib

Para comprobar la importación de las librerías se observará que se puede agregar a la clase creada, pulsando el botón "*Añadir*".
Ahora para añadir un comando personalizado, que a su vez permite ensamblar la clase con extensión *.cs* directamente desde el programa MonoDevelop, esto se va a realizar con ayuda "*Comandos personalizados*" desde el MonoDevelop, el MRC permite crear los archivos .sba, .sdx y .sxp, los cuales se enviarán al nodo para su ejecución.

Para este propósito es necesario realizar lo que se indica a continuación:

- Se selecciona el proyecto al cual se desea agregar los comandos personalizados.

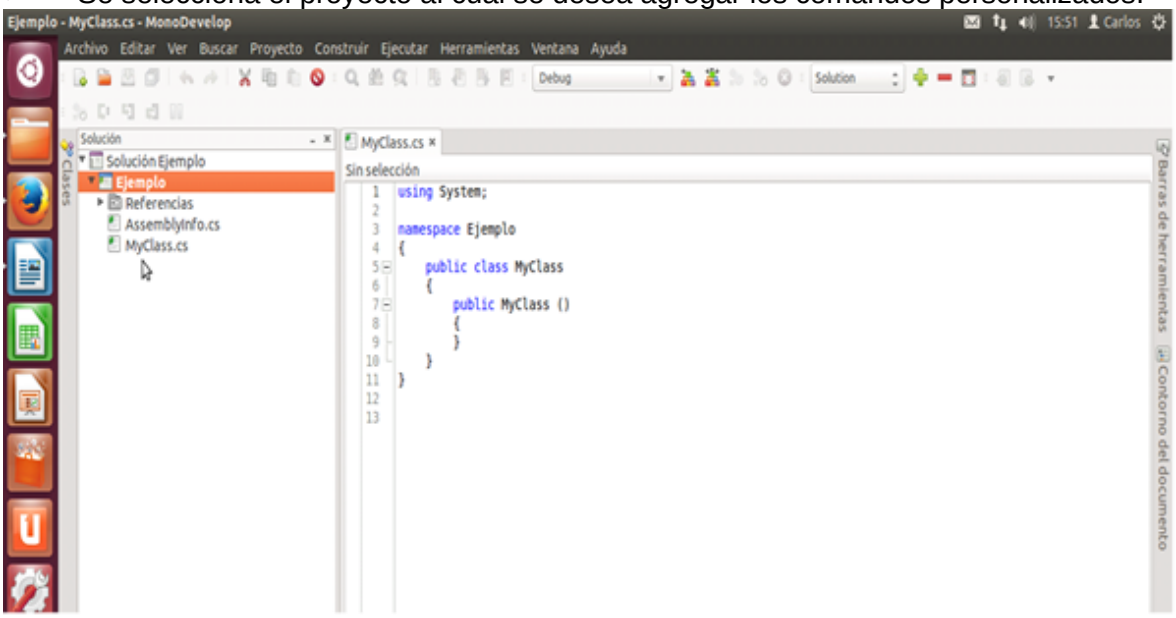

Figura A-14 Añadiendo compiladores dentro de MonoDevelop

- Click derecho, se selecciona "*opciones*".

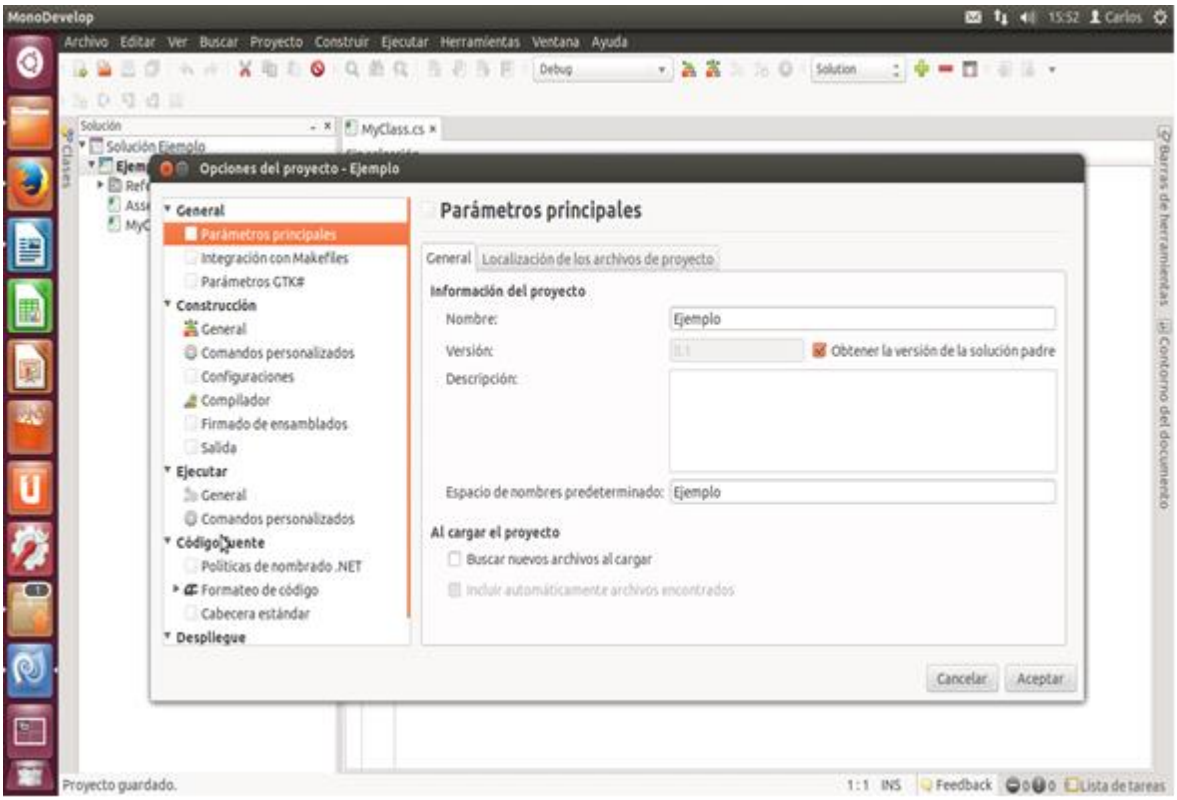

Figura A-15 Opciones del proyecto

- Se selecciona "*comandos personalizados*".

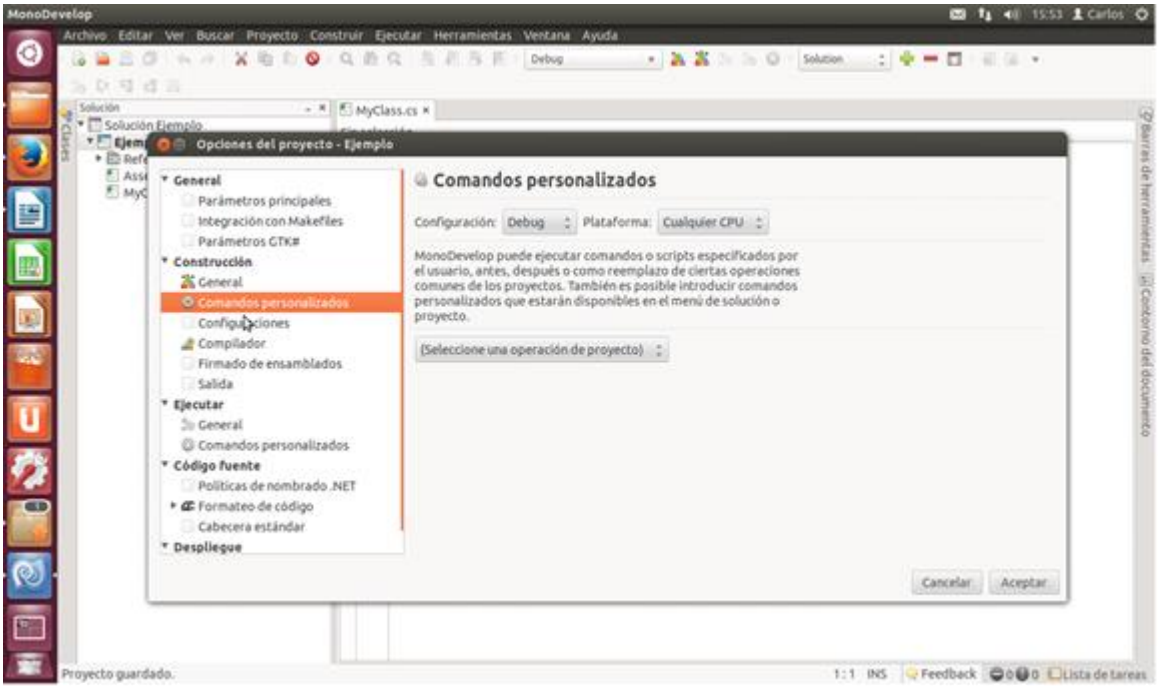

Figura A-16 Comandos personalizados

- En la opción seleccione una operación de proyecto, se selecciona "*Después de construir*".

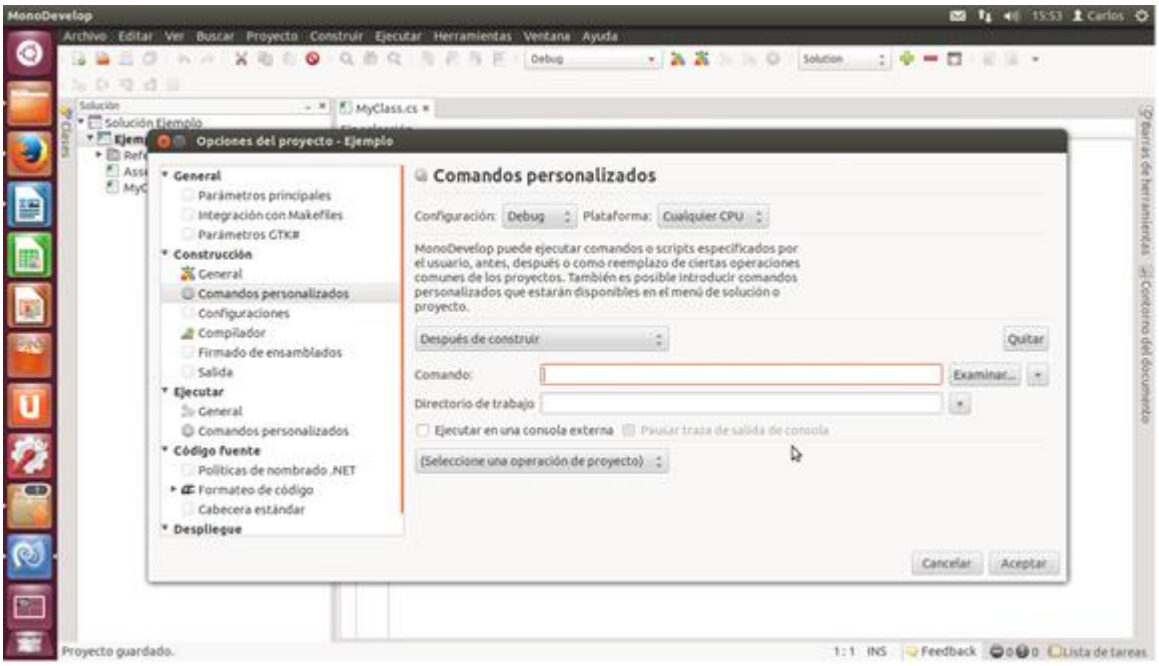

Figura A-17 Añadiendo comandos personalizados

- En comando se selecciona "*examinar*".

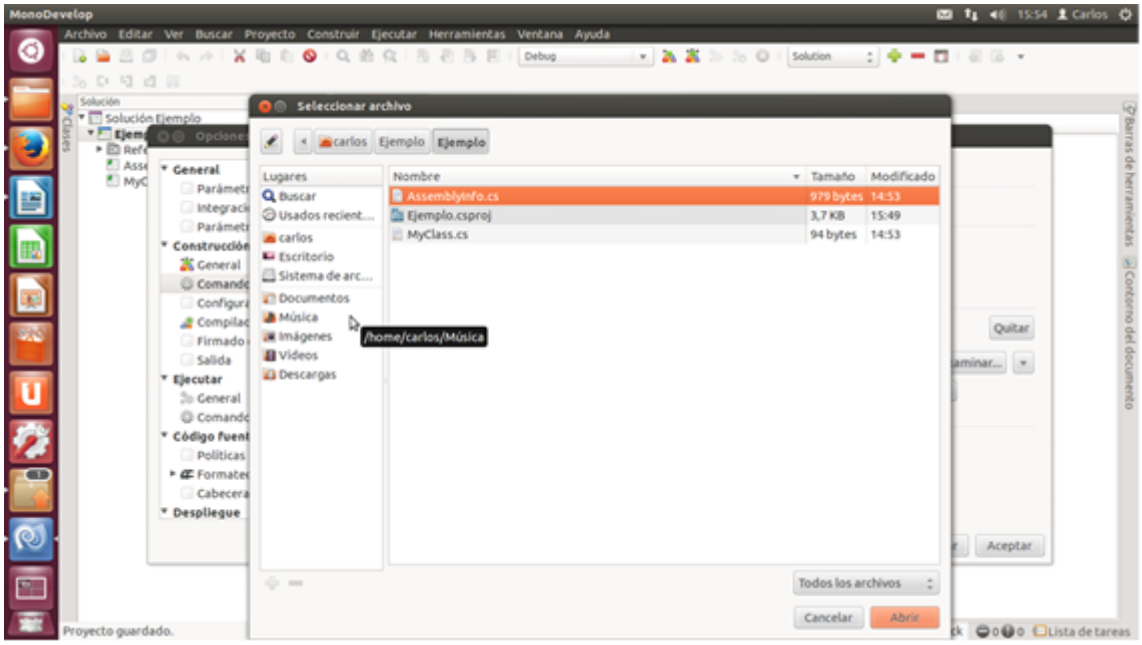

Figura A-18 Compilador MRC

- Se busca el directorio *moterunner* y en este se localiza al compilador *mrc*.
- En este caso la dirección es */moterunner/linux64/bin.*

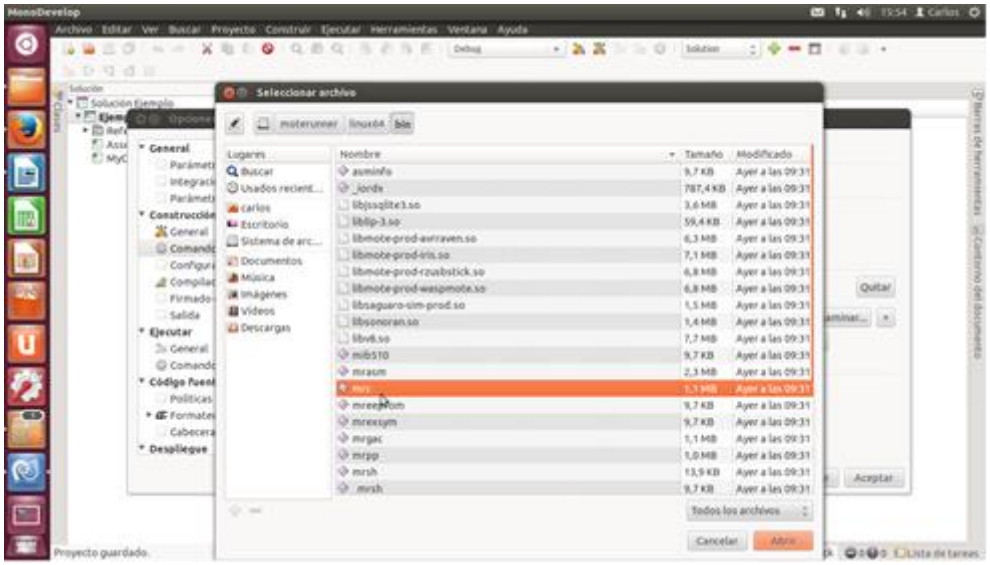

Figura A-19 Compilador MRC

En la caja de edición del comando personalizado, se completa dando un espacio y añadiendo lo siguiente:

```
1    --debug --assembly=Nombre-1.0 --ref=/moterunner/examples/mrv6/src/tmp/mrv6-lib-1.0
     NombreClase.cs
```

Código A-11 Comandos personalizados

Nombre: cualquier nombre
NombreClase.cs: El nombre de la clase que se compila con la extensión *.cs*

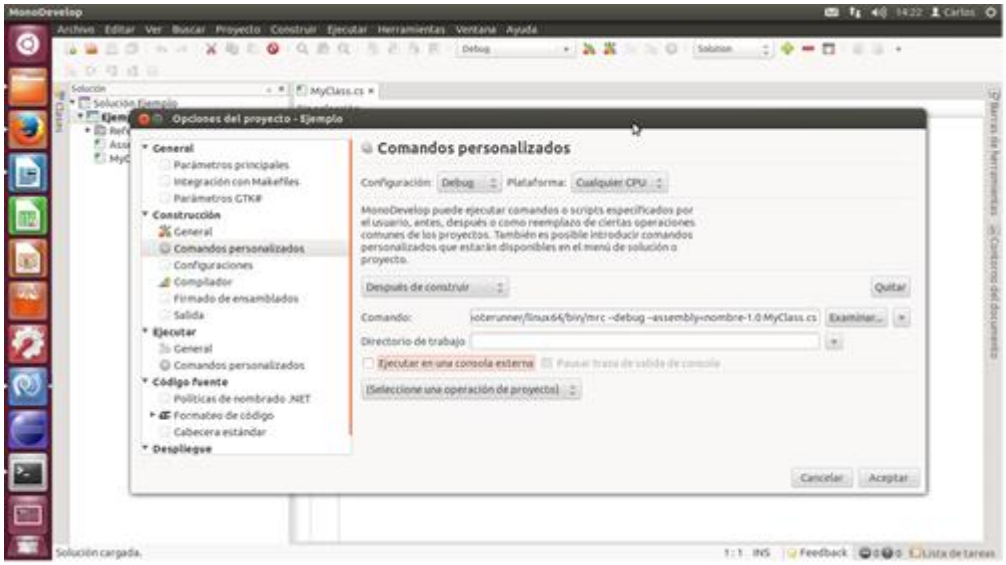

Figura A-20 Comandos personalizados

- En el directorio de Trabajo.
- Click en la pestaña que indica el Directorio de Trabajo y se selecciona Project Directory.

Click en las opciones para ejecutar en consola la depuración, para indicar que se ha compilado correctamente y se presente en consola posibles errores y éxito en la compilación.

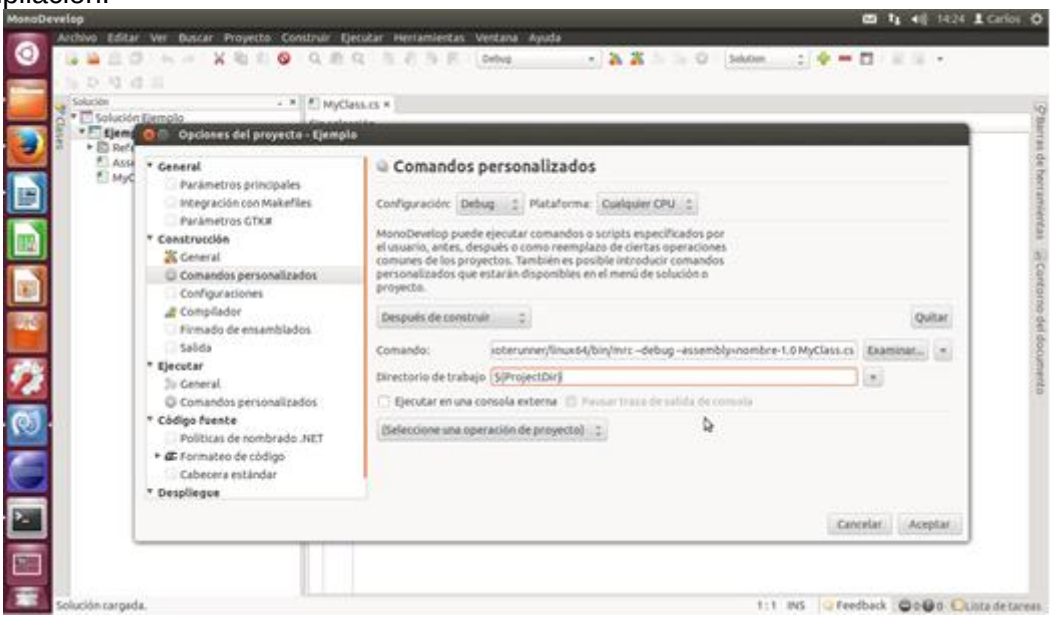

Figura A-21 Directorio actual es donde se va a compilar nuestro código

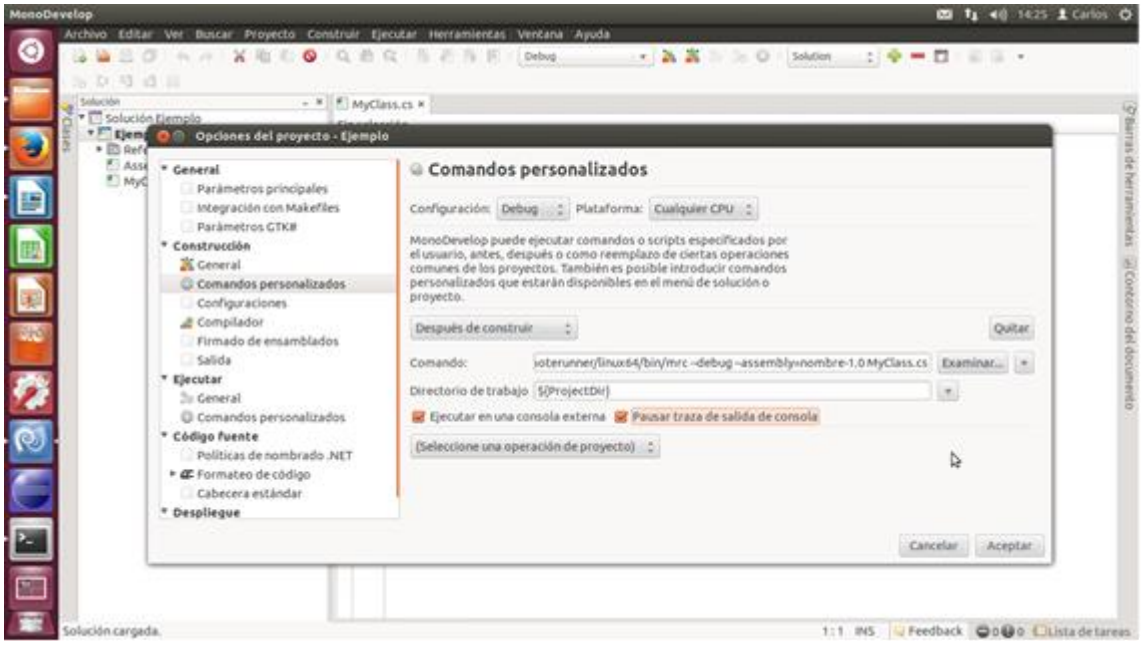

Figura A-22 Ejecución y depuración en la terminal

El proyecto se depura, en una ventana de consola, en la cual se indicará si se ejecutó perfectamente, para su verificación se buscará en la carpeta solución los archivos .sba, .sdx, .spx.

```
namespace com.ibm.saguaro.gateway.generic {
 using com.ibm.saguaro.system;
 using com.ibm.saguaro.mrv6;
 using com.ibm.saguaro.util;
 using com.libelium.rtc;
 using com.libelium.common;
    public class ReplySocket : UDPSocket {
        internal static uint LOCAL_PORT = 1024;
internal static ReplySocket socket = new ReplySocket();
internal static ADC adc;
public ReplySocket() {
            this.bind(LOCAL_PORT);

adc = new ADC();

adc.open(0x01,GPIO.NO PIN,0, 0);
}
public override int onPacket(Packet packet) {
            // An UDP packet has reached this mote
            // Toggle LED 2
            LED.setState(2, (byte)(LED.getState(2)^1));
            // We read ADC channel 0
                    //uint adcRead = adc.readChannel(0);
            // We store the ADC reading on a byte array
            byte[] buf = new byte[2];
            //Util.set16be(buf,0,adcRead);
            // ADC value hexadecimal representation is transformed to ASCII.
            // Doing so we can see the hexa value directly on Netcat
                    uint temp = (uint)RTC.getInstance().getTemperature();
                    byte aux = (byte)temp;
                    buf[0] = aux;
            buf = adc2Ascii(buf);
            uint len = packet.payloadLen;
try {
packet.swap(len);
} catch {
return 0;
}
            Util.copyData(buf, 0, packet.payloadBuf, packet.payloadOff, 4);
this.send(packet);
return 0;
```

```
}

private byte[] adc2Ascii(byte[] adcValue){
            byte[] aux = new byte[4];
            aux[0]=(byte)((adcValue[0] & 0xF0) >> 4);
            aux[1]=(byte)(adcValue[0] & 0x0F);
            aux[2]=(byte)((adcValue[1] & 0xF0) >> 4);
            aux[3]=(byte)(adcValue[1] & 0x0F);

            for(int i=0;i<4;i++){
                if(aux[i] < 9){
                 aux[i] += 48;
}
                            else{
                   aux[i] += 55;
}
}
return aux;
}
}
}

public override int onPacket(Packet packet) {
            // An UDP packet has reached this mote
            // Toggle LED 2
            LED.setState(2, (byte)(LED.getState(2)^1));
            // We read ADC channel 0
                      //uint adcRead = adc.readChannel(0);
            // We store the ADC reading on a byte array
            byte[] buf = new byte[2];
            //Util.set16be(buf,0,adcRead);
            // ADC value hexadecimal representation is transformed to ASCII.
            // Doing so we can see the hexa value directly on Netcat
                      uint temp = (uint)RTC.getInstance().getTemperature();
                      byte aux = (byte)temp;
                      buf[0] = aux;
            buf = adc2Ascii(buf);
            uint len = packet.payloadLen;
try {
packet.swap(len);
} catch {
return 0;
}
            Util.copyData(buf, 0, packet.payloadBuf, packet.payloadOff, 4);
this.send(packet);
return 0;
```

Código A-12 Código para leer la temperatura del sensor del nodo

Para el momento de compilar el código implementado en el lenguaje de programación C#, es importante tener en cuenta todas las referencias utilizadas para la compilación, además si se va a desarrollar aplicaciones con los módulos externos (sensores), tener muy en cuenta el agregar las respectivas librerías.

Se puede descargar las librerías de los distintos sensores mediante este link:

http://www.libelium.com/downloads/moterunner-1.0.zip.

A continuación, se presentará un ejemplo al utilizar el sensor de temperatura del mote. Para lo cual se logró con el siguiente código:

Como se puede ver en la Figura A-23, se agregaron las referencias de las librerías con la versión más alta y solo se selecciona una de ellas como es el ejemplo de la librería *saguaro-system*, ya que existen las versiones *11.0, 11.1 y 11.4* por lo cual se deberá seleccionar la versión 11.4.

Además, como se va a utilizar el sensor de temperatura se necesita las librerías de dicho sensor las cuales son en nuestro caso es "*wasp-rtc-1.0.dll*" y "*wasp-common-1.0*"

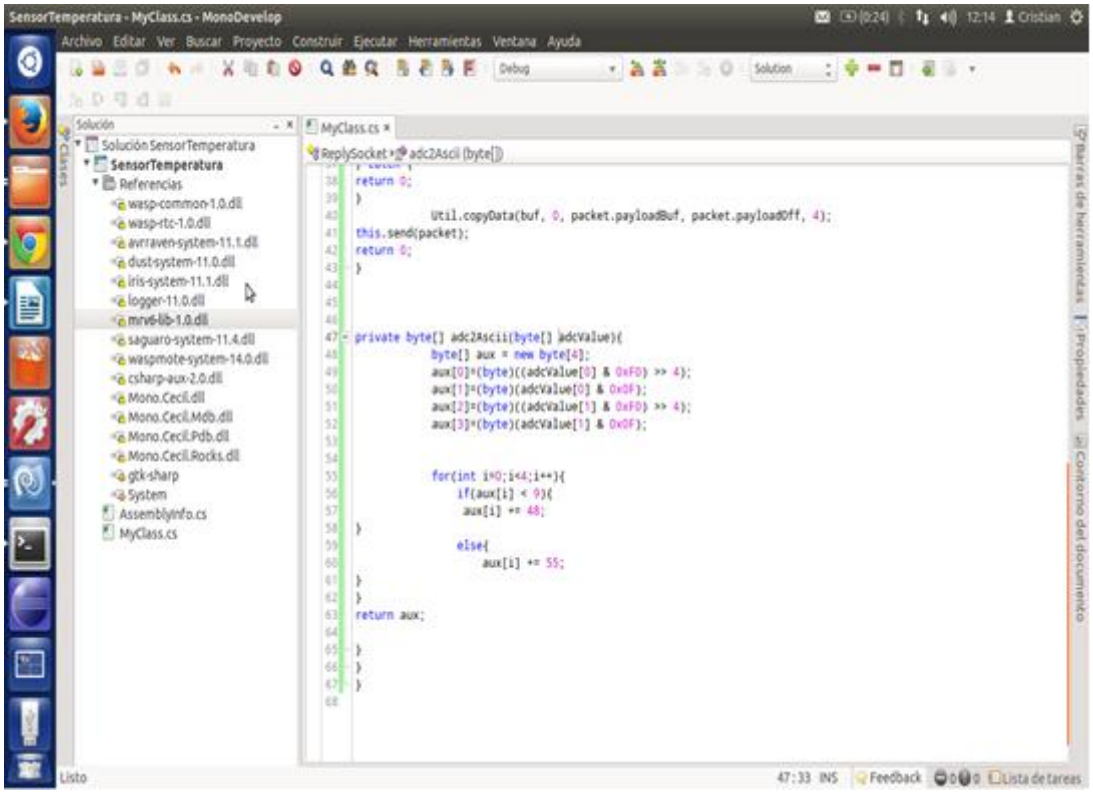

Figura A-23 Librerías wasp-common y wasp-rtc para lector de temperatura

Para la compilación del código implementado, se realiza mediante consola de la aplicación implementada, por lo cual el comando en la terminal sería el siguiente:

```
mrc --debug --assembly=SenseTemp-1.0 --ref=/moterunner/examples/mrv6/src/tmp/mrv6-lib-1.0
--ref=/home/cristian-espinosa/Escritorio/moterunner-1.0/src/com/libelium/rtc/wasp-rtc-1.0
--ref=/home/cristian-espinosa/Escritorio/moterunner-1.0/src/com/libelium/common/wasp-
common-
1.0/home/cristianespinosa/Escritorio/C#_Moterunner/SensorTemperatura/SensorTemperatura/My
Class.cs
```

Código A-13 Compilación de la aplicación del sensor de temperatura

Figura A-24 Compilación MRC en la terminal

A continuación, se puede observar que en el escritorio se han creado los 3 archivos compilados.

Figura A-25 Creación de los tres archivos mediante el compilador MRC

INSTALACIÓN DE FIRMWARE EN CADA NODO

El firmware se carga o instala a cada uno de los nodos, con ayuda del programador AVR. Cabe recalcar que el firmware es algo parecido al BIOS de nuestro PC, esto es solo para activar el hardware en cada nodo, para que éste pueda ser utilizado de la forma que requiera el usuario y puedan ser cargadas las aplicaciones desarrolladas.

Para esto, se puede realizar con el AVR Studio en Windows o mediante Linux gracias al AVRdude.

- Para el caso de Windows es importante realizar una descarga en el link indicado, pero antes de proceder a la descarga se tiene que registrar con un correo electrónico para la creación de un usuario.
- Link de Descarga: http://wwww.atmel.com/tools/atmelstudio.aspx

Para este proyecto se utilizará todas las herramientas provistas por el sistema operativo Ubuntu 12.04 LTS en Linux, para esto se procede a instalar el software "*avrdude*" desde el "Centro de Software de Ubuntu", como se puede apreciar la búsqueda se realiza por su nombre, es decir *avrdude* y se procede con la descarga.

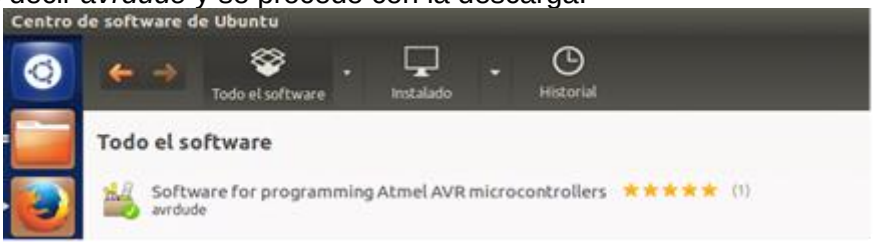

Figura A-26 Centro de Software Ubuntu, instalación de avrdude

CARGANDO EL FIRMWARE EN EL NODO

- Primero antes de todo se conecta el nodo *Waspmote* (Hardware) al cable de poder mediante el conector USB al adaptador de 120V o a la PC, este proceso recomienda el fabricante y debe hacerse por 24 horas. Luego de estar cargado el nodo, se enciende el Waspmote moviendo los interruptores de encendido hacia a la izquierda como muestra en la Figura A-27.

Figura A-27 Encendido del nodo Waspmote

- El siguiente paso es conectar el programador AVR a la PC mediante el cable como muestra la Figura A-28.

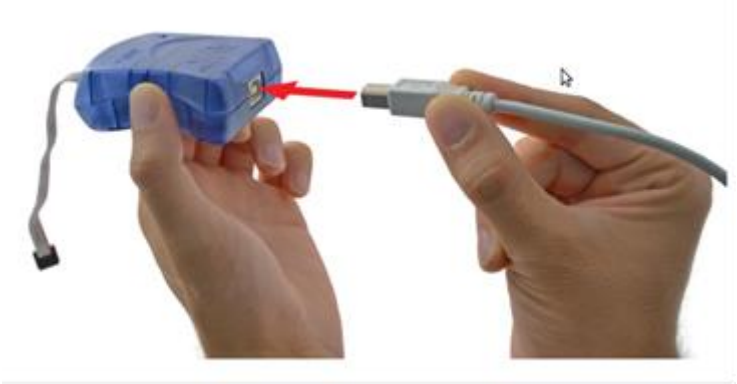

Figura A-28 Insertando el cable USB al quemador AVR

- Ahora el programador puede ser conectado al Waspmote mediante el conector ICSP en la parte trasera del mismo.

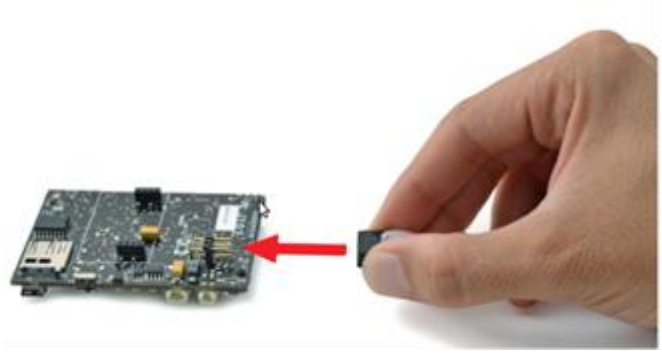

Figura A-29 Conector ICSP conectando al nodo

- Una vez conectado lo anteriormente mencionado, el siguiente paso es cargar el firmware al *Waspmote*, como se ha mencionado en nuestro caso se hizo mediante consola en LINUX con los siguientes comandos:

El comando indicado en el Código A-14 permite establecer y quemar los respectivos fusibles que permitirán transmitir el Archivo .hex del firmware al nodo.

```
1   root@ubuntu:~# sudo avrdude -c avrispmkII -p m1281 -P usb -b 115200 -e -u -U
    efuse:w:0xFF:m -U hfuse:w:0xD0:m -U lfuse:w:0xFF:m
```

Código A-14 Quemando los fusibles de los nodos

Finalmente se procede a instalar el archivo "*.hex"* del firmware que permitirá cargar en cada uno de los nodos, las aplicaciones desarrolladas anteriormente, para este propósito hay que trasladarse al directorio donde se encuentra el archivo hex, el mismo se encuentra en

la siguiente ubicación “/moterunner/firmware/waspmote.hex”. Cómo se indica en el Código A-15 y en las Figuras A-30, A-31, A-32.

```
1   root@ubuntu:~# sudo avrdude -c avrispmkII -p m1281 -P usb -b 115200 -U
    flash:w:/moterunner/firmware/waspmote.hex
```

Código A-15 Quemando el firmware Waspmote

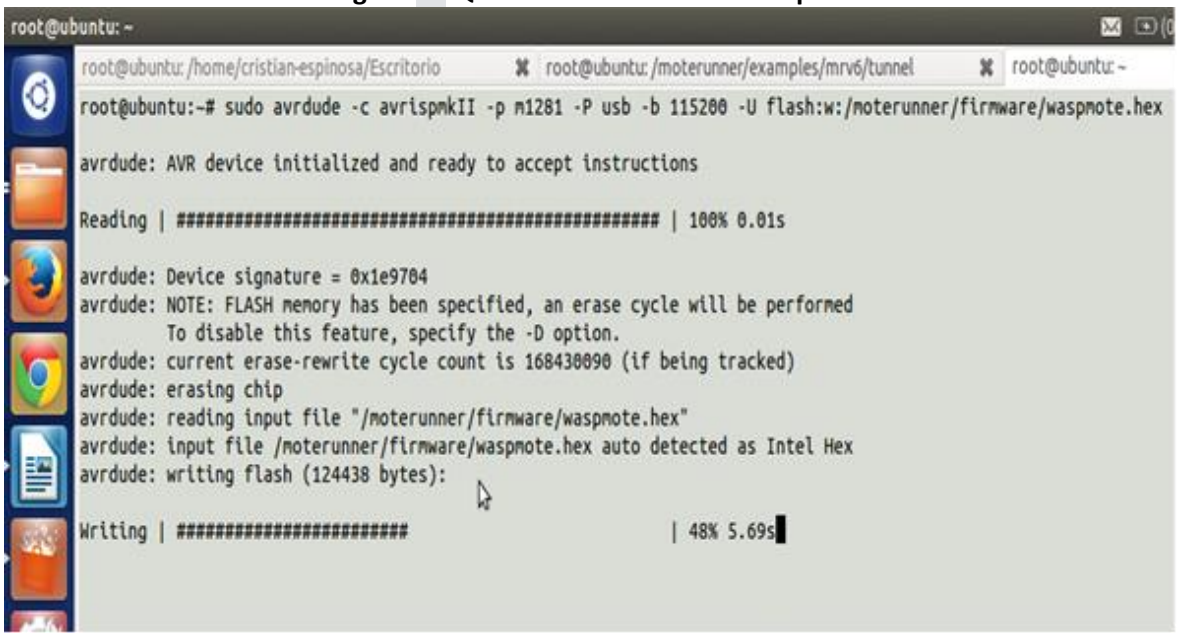

Figura A-30 Ejecutando el primer comando para quemar los fusibles en los nodos

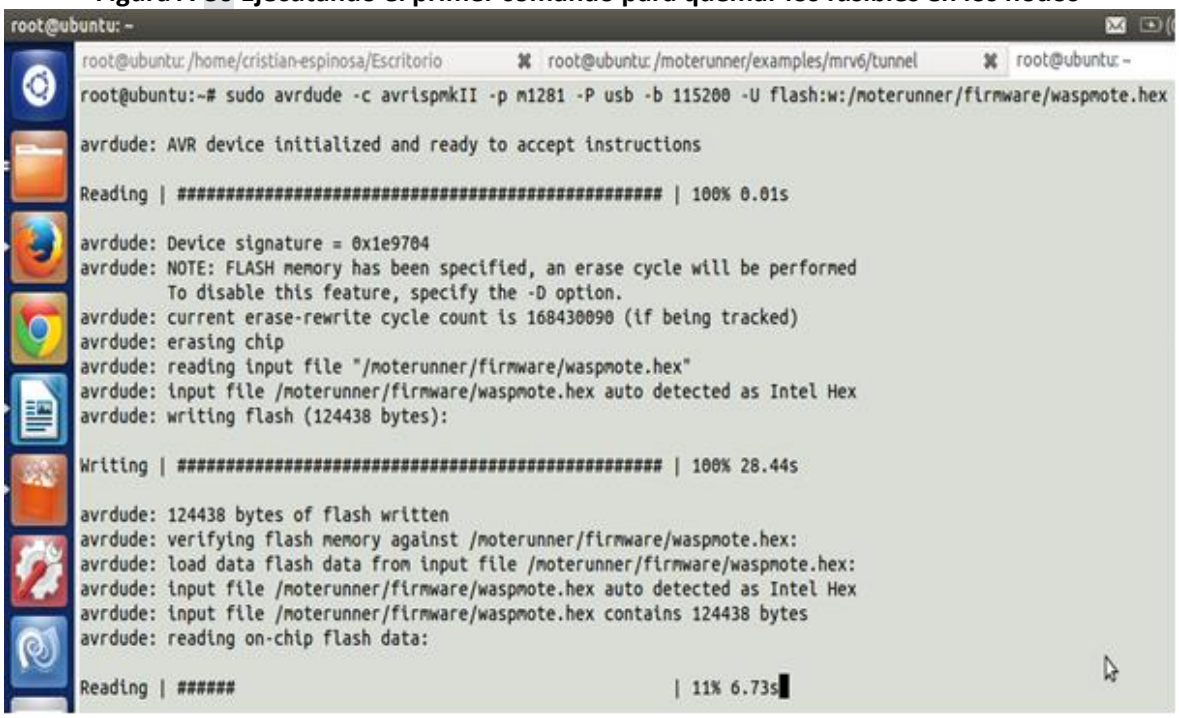

Figura A-31 Quemando el firmware Waspmote en el nodo

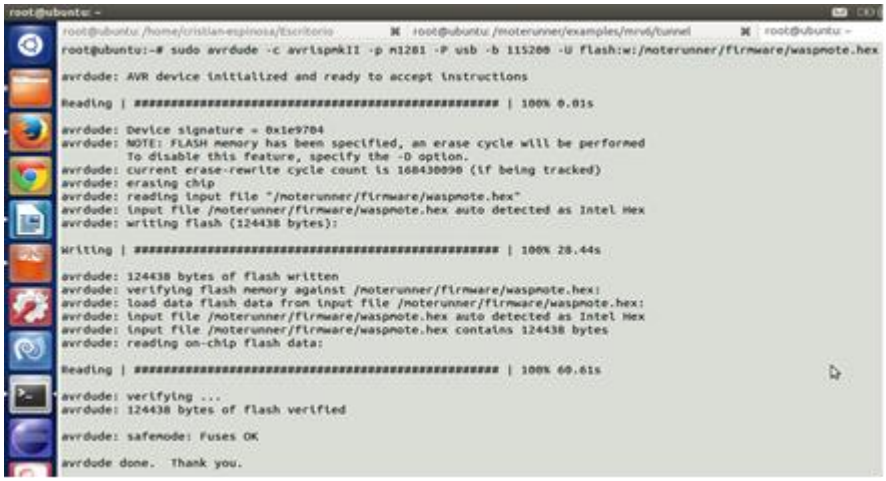

Figura A-32 Finalización de la grabación del firmware en el nodo

Cabe recalcar que una vez instalado el firmware en cada *Waspmote*, ya no se necesita el programador AVR, ya que solo mediante el cable USB conectado al *Waspmote* a la PC y con los comandos del *Moterunner Shell*, se puede cargar el archivo *.sba* que se crea al momento de utilizar el compilador "mrc". Al ejecutar este se nos va a crear tres archivos .sdx, .sba, .sxp el archivo que se envía al *Waspmote* es el .sba(*Saguaro Binary Assembly*) mientras que los otros dos son documentación del assembly.

COMPILANDO APLICACIONES PARA LOS WASPMOTE UTILIZANDO EL SHELL

Esto se puede realizar directamente desde el software MonoDevelop o mediante el shell en la terminal ejecutando el siguiente comando:

```
1   root@ubuntu:~# mrc --debug --assembly=<name>-<major>.<minor> --<Dirección de la
    librería a utilizar> <Dirección donde se encuentre nuestra clase.cs>
```

Código A-16 Ejemplo de la utilización del compilador MRC

Nota: Para la utilización de este comando se necesita exportar las variables de entorno, ya que solo funciona por cada terminal que se ejecute.

Como ejemplo de su utilización, actualmente la terminal se encuentra en el escritorio:

```
1   root@ubuntu:/home/cristian-espinosa/Escritorio# mrc --debug --assembly=SenseTemp-1.0 -
    -ref=/moterunner/examples/mrv6/src/tmp/mrv6-lib-1.0 --ref=/home/cristian-
    espinosa/Escritorio/moterunner-1.0/src/com/libelium/rtc/wasp-rtc-1.0 --
    ref=/home/cristian-espinosa/Escritorio/moterunner-1.0/src/com/libelium/common/wasp-
    common-1.0 /home/cristian-
    espinosa/Escritorio/C#_Moterunner/SensorTemperatura/SensorTemperatura/MyClass.cs
```

Código A-17 Compilación utilizando el compilador MRC

Como se encuentra actualmente en el escritorio se van a crear los 3 archivos antes mencionados en el Escritorio como se puede ver en la imagen.

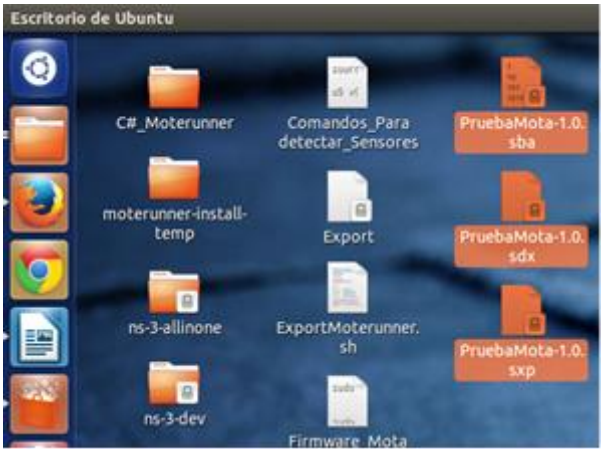

Figura A-33 Archivos generados una vez que se ejecutó la compilación MRC

CARGANDO EL CODIGO DE LAS APLICACIONES A LOS NODOS WASPMOTE

a. Nodos Simulados

Para esto primeramente es necesario tener el servidor "*mrsh*" activo, es decir ejecutándose, como se va a ver en una página web, este servidor se puede visualizar en el navegador web en la siguiente dirección http://localhost:5000/.

En la consola se ejecuta el *script ExportMoterunner.sh* realizado, con el propósito de exportar el path y librería para poder ejecutar el servidor "mrsh" desde cualquier directorio que se esté ubicado, una forma de visualizar que se ha lanzado en el navegador el servidor, es verificándolo en una página web como se mencionó anteriormente.

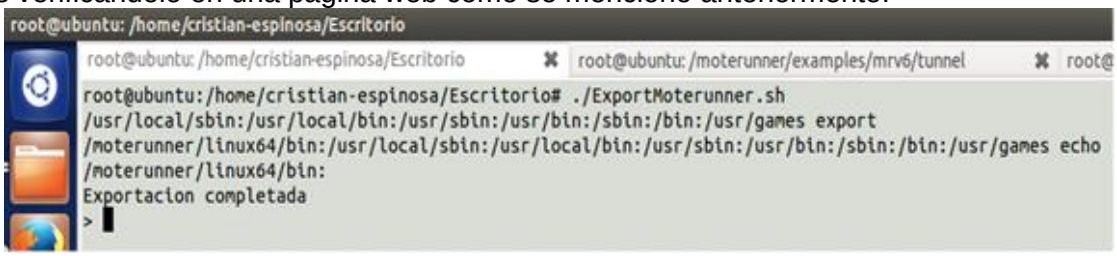

Figura A-34 Ejecución del script

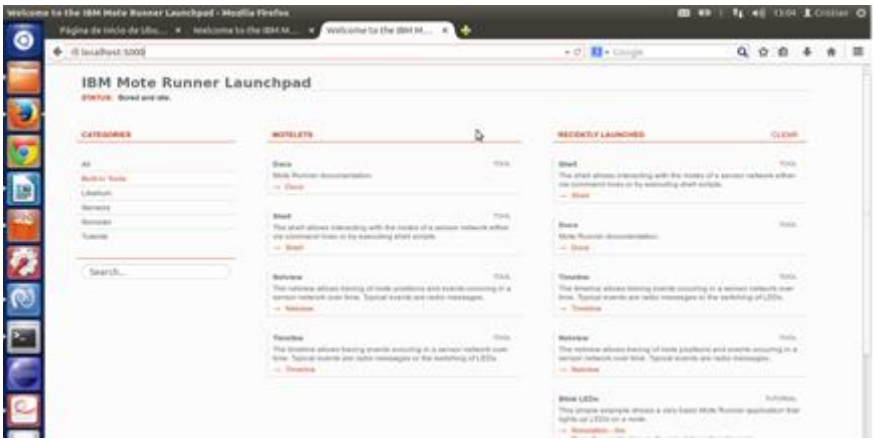

Figura A-35 Servidor MRSH

Algo importante en esta sección es que el servidor Web MRSH, tiene una opción de Shell que es prácticamente la misma que se está ejecutando en la terminal. Por lo tanto se puede ingresar comandos en cualquiera de estos dos. El shell del Servidor Web es mucho más amigable que el de la terminal; por lo cual se recomienda usar el shell del servidor web.

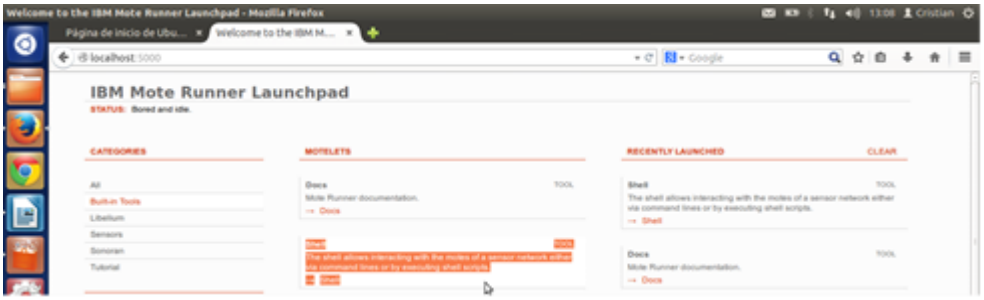

Figura A-36 Shell de MRSH

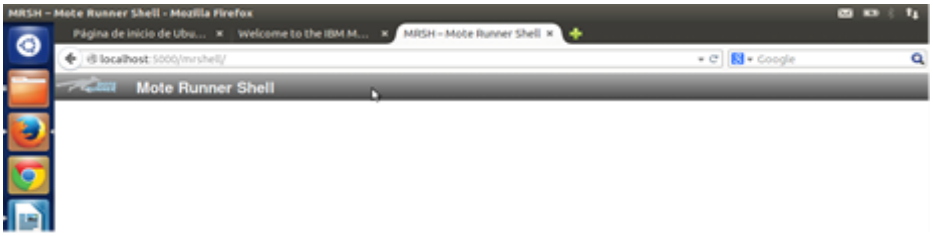

Figura A-37 Shell de MRSH

Ahora una vez que se encuentra dentro del shell de la terminal o del Servidor Web. Para la simulación se debe iniciar el proceso *saguaro* que nos permite especificar en qué puerto en el que se desea trabajar o por defecto está en el puerto 44044.

```
> saguaro-start
sag://localhost:44044
```

Código A-18 Inicio de Saguaro para simular nodos

Al ingresar el comando para el inicio del saguaro, se presentará que está corriendo en el local-host en el puerto 44044.

Ahora se establece la conexión con el proceso saguaro.

```
> saguaro-connect
```

Código A-19 Conexión con el proceso Saguaro

Se crea un *mote* con el proceso ya creado con; el –d se especifica qué tipo de hardware a simular.

```
> mote-create -d
waspmote
```

Código A-20 Creación de un nodo waspmote

Para cargar el programa a simular en el nodo se debe utilizar el *Mote Manager* (MOMA) el comando es el "*moma-load"*. Hay que recalcar que, en esta sección, es que desde la misma *shell* del *Moterunner* se puede ejecutar comandos para ver en qué directorio se encuentra actualmente el administrador, similar a como estar desde el terminal, como cuando se ejecuta el comando "*pwd*", también para ver un listado de los archivos se escribe el comando "*ls*" o "*ls -l*", para un cambio de directorio el comando "*cd*". Se menciona esto ya que al momento de cargar el programa debe desplazar hasta el directorio donde se encuentre el programa que se desea cargar en el nodo, es decir donde se encuentre el archivo generado mediante el compilador MRC que es archivo con extensión. sba.

Ejemplo:

En este ejemplo se carga la librería "*mrv6-lib-1.0*" y "*reply-1.0*" que están en diferentes directorios.

```
> cd /moterunner/examples/mrv6/src/tmp
> moma-load mrv6-lib-1.0
> cd /moterunner/examples/mrv6/reply
> moma-load reply-1.0
```

Código A-21 Cargando librerías

Moma-load: permite cargar el mismo archivo a varios waspmote o solo a un nodo.

Para borrar el programa se utilizará el comando "*moma-delete*", para ello se debe colocar el nombre del archivo cargado observando con el moma-list.

```
1  > moma-delete reply-1.0
```

Código A-22 Borrar un programa

Para conocer todos los comandos y su sintaxis colocar.

```
1 >?
```

Código A-23 Para ver todos los comandos aceptados por la plataforma

b. **Nodos Físicos**

- Se conecta directamente con el cable mini USB desde la PC al nodo.
- Asegurarse que el nodo esté encendido y no haya nodos creados, para esto verifique con el comando *moma-list*, y que no esté funcionando.
- Si se conoce el puerto USB en el cual está el Nodo solo se utiliza el siguiente comando:

```
1  > mote-create -p /dev/ttyUSB0
```

Código A-24 Conexión del nodo físico con el servidor MRSH

Figura A-38 Conexión del nodo con el servidor MRSH

Si no se conoce el puerto entonces se procede a realizar lo siguiente:

- Se desconecta el cable USB de la PC y se ingresa el siguiente comando en el Shell.
- Es importante seguir los pasos antes de ejecutar lip-enumerate.

```
1    >lip-enumerate --wait --waspmote --timeout 30s
```

Código A-25 Comando para verificar en que puerto USB se conectó el nodo

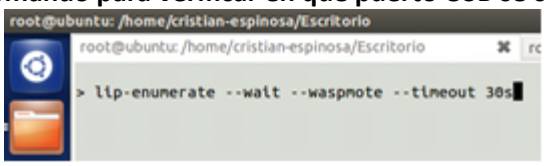

Figura A-39 LIP enumerate

```
1    > lip-enumerate --wait --waspmote --timeout 30s
2    Mote Runner port  (mote-create): /dev/ttyUSB0
```

Código A-26 Verificación del puerto USB

Se procede a conectar el USB a la PC y detectará en que puerto estará conectado.
Como se apreciar se ha detectado que el nodo se encuentra en */dev/ttyUSB0*

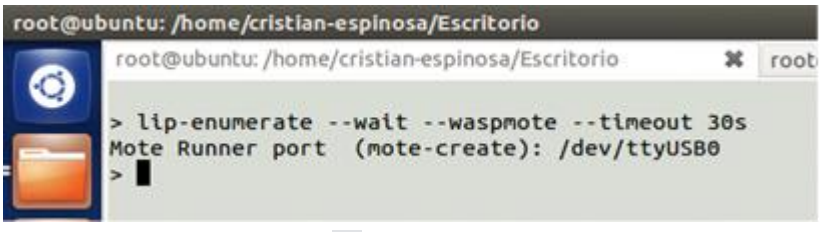

Figura A-40 Conexión del Nodo

Una vez localizado el puerto lo siguiente a realizar es conectar el nodo con el siguiente comando:

```
1    > mote-create -p /dev/ttyUSB0
2
```

Figura A-41 Conexión con el Nodo

Ahora para cargar el archivo se debe mover al directorio en el cual se encuentre la librería y se procede a cargar la librería *mrv6-lib* y el ejemplo del *reply* con el moma-load.

```
1    > cd /moterunner/examples/mrv6/src/tmp
2    > moma-load mrv6-lib-1,0
3    > cd /moterunner/examples/mrv6/reply
4    > moma-load reply-1.0
```

Código A-27 Ejemplo carga de librerías

DEPURACIÓN DE APLICACIONES UTILIZANDO EL PLUGIN DE ECLIPSE

Se puede instalar el Eclipse (Entorno de desarrollo integrado Eclipse) mediante el Centro de Software de Ubuntu.

Figura A-42 Centro de Software de Ubuntu, descarga de Eclipse

Una vez instalado se abre el programa instalado, que es el Eclipse.

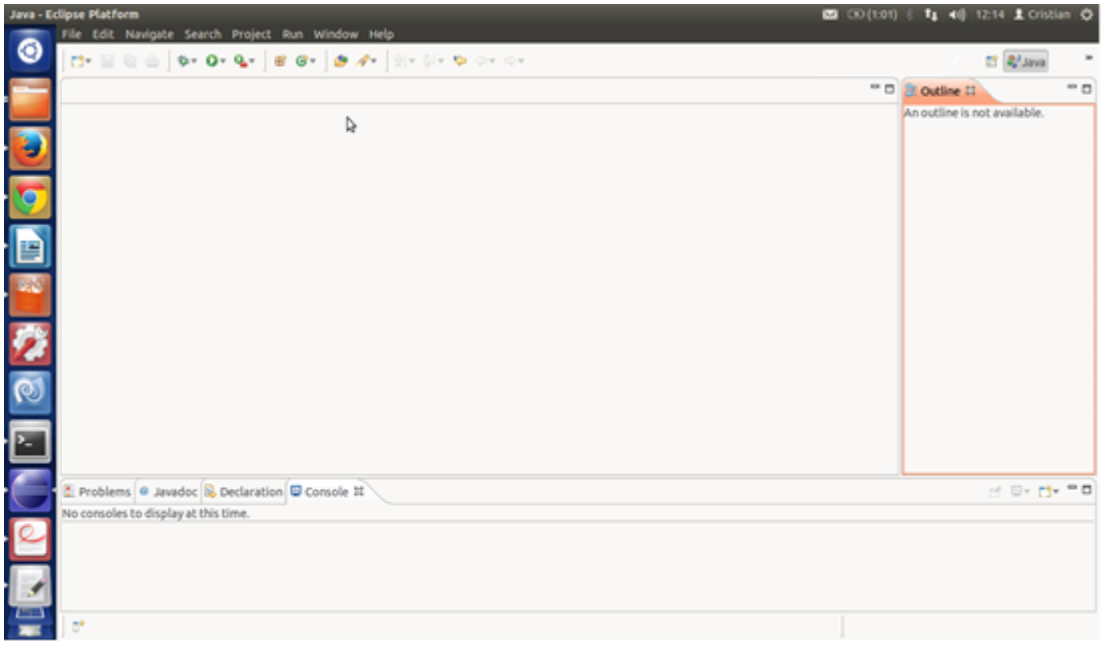

Figura A-43 IDE Eclipse

- Se da un click a la pestaña *Help* e *Install New Software*.

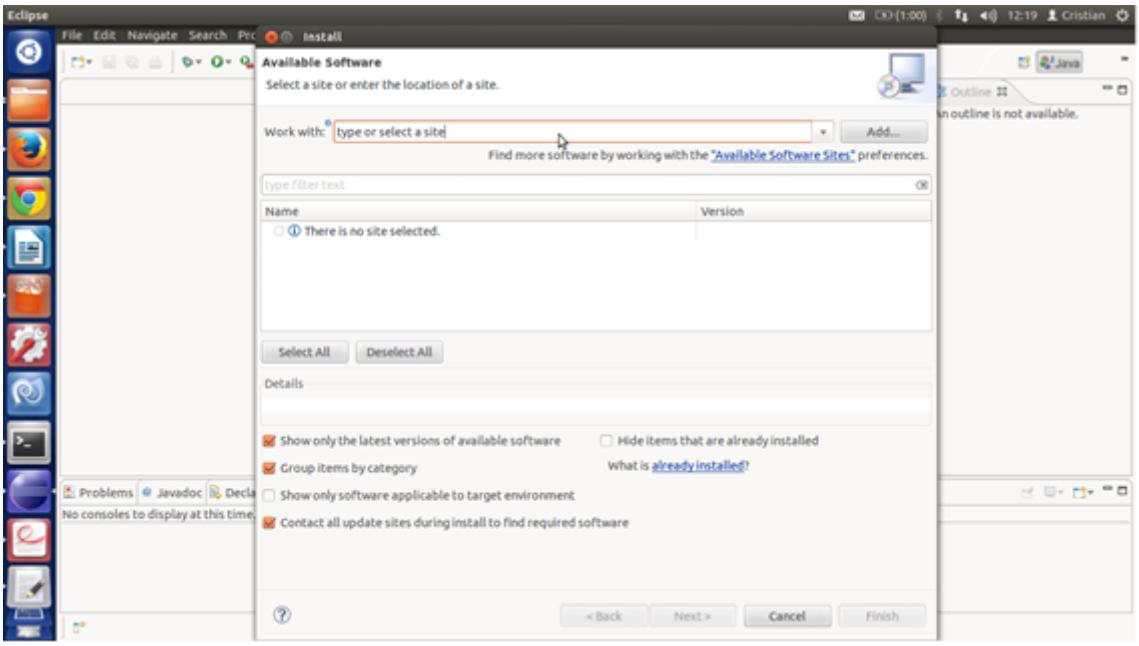

Figura A-44 Instalar plugin-Eclipse

- Se pulsa en la pestaña *Add*.

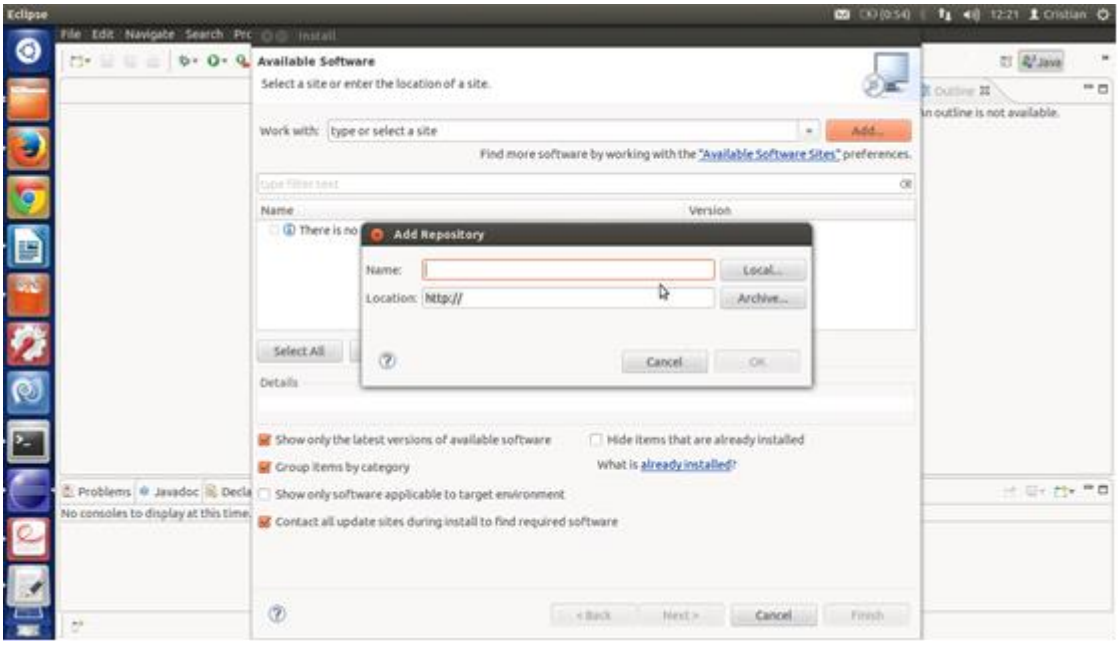

Figura A-45 Añadiendo nuevo software

Se presiona el botón *Archivo* y se busca el Plug-in en la dirección "*/moterunner/IDE/mrdevel-feature.zip".*

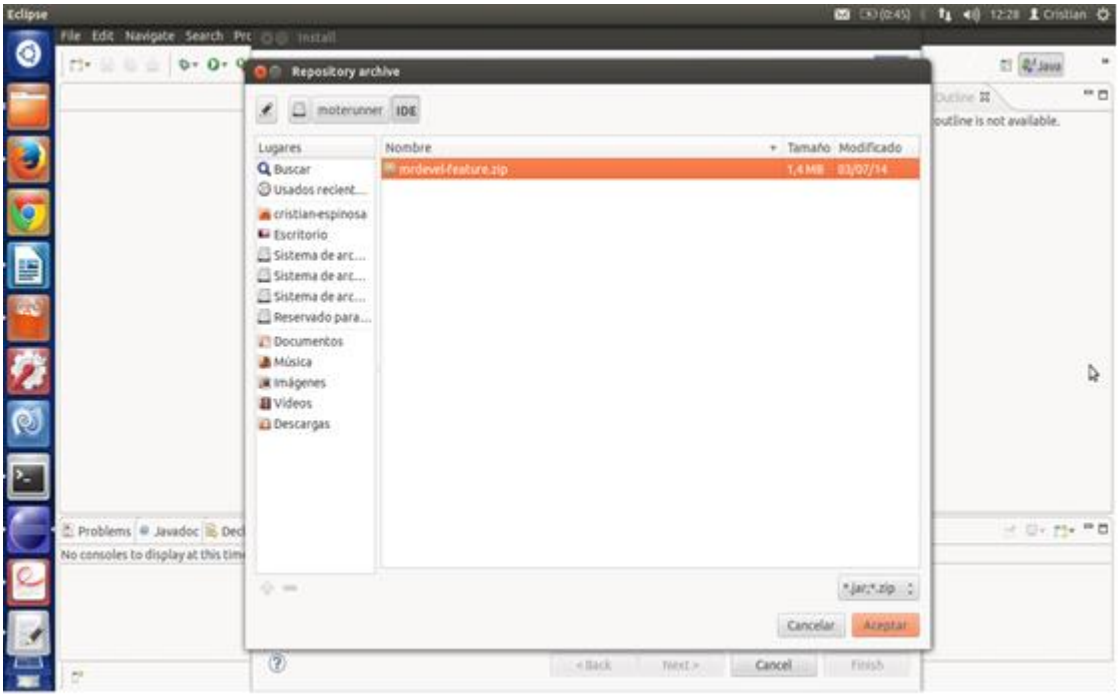

Figura A-46 Añadiendo el plugin

Ahora se pulsa en el botón *Aceptar*, a continuación, se selecciona *Mote Runner Development.*

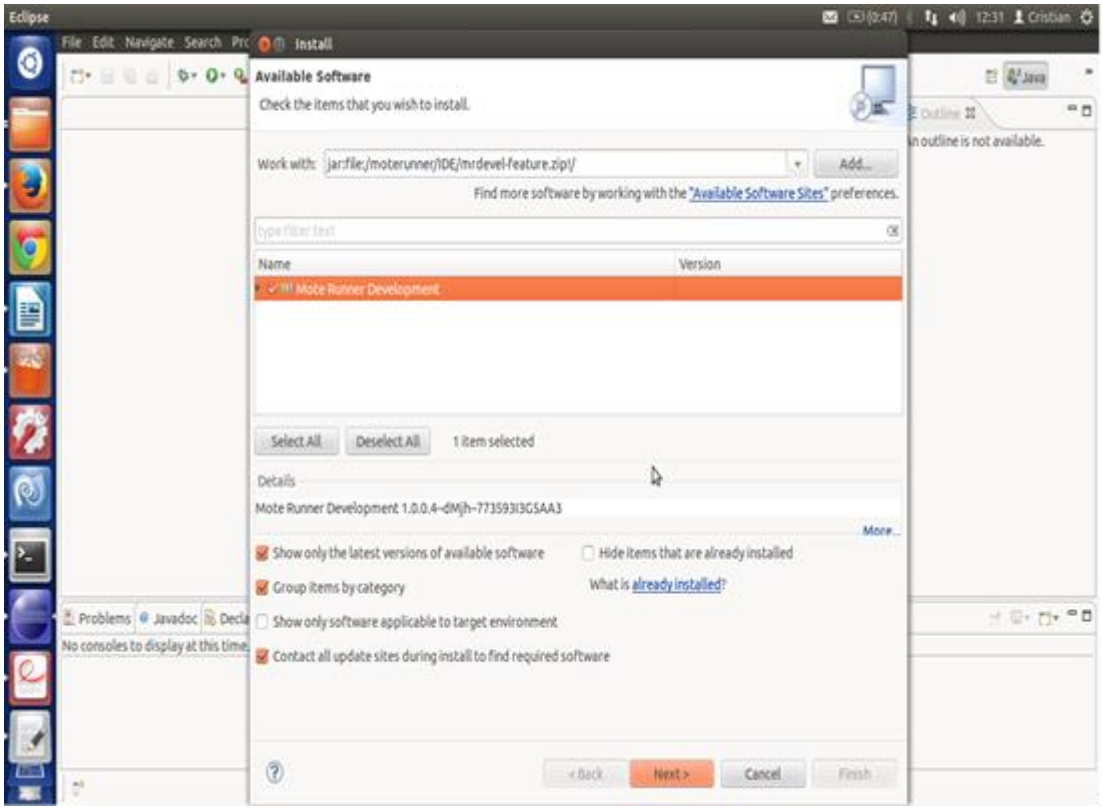

Figura A-47 Eclipse Plugin

Se pulsa el botón *Next*.

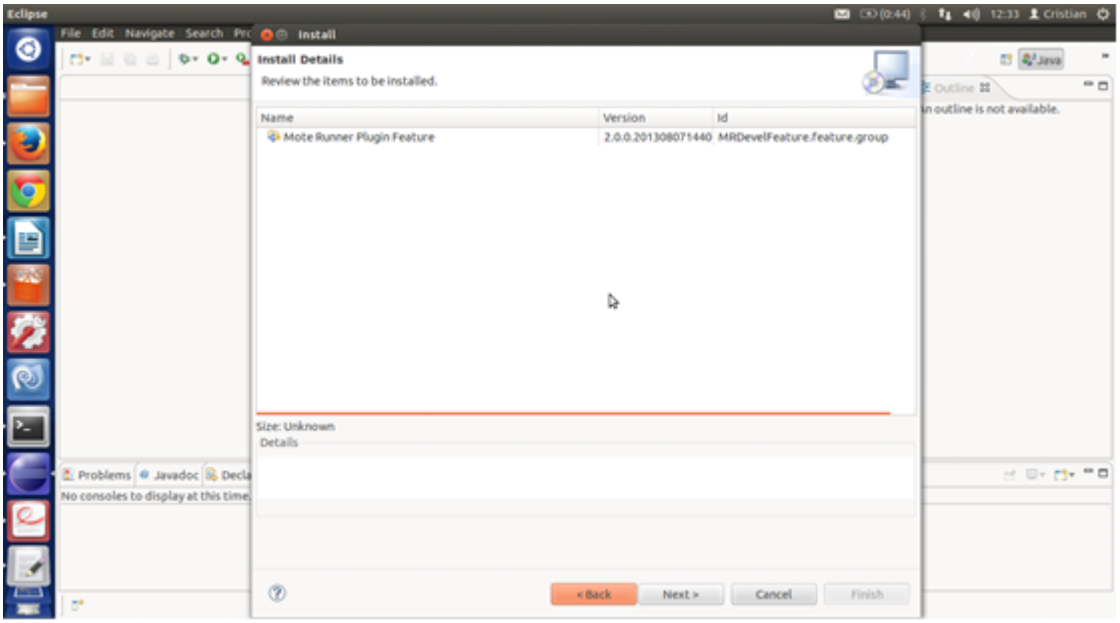

Figura A-48 Instalación Plugin Eclipse

Se pulsa *Next*, a continuación, se acepta los términos de la licencia y finalmente se pulsa el botón *Finish*.

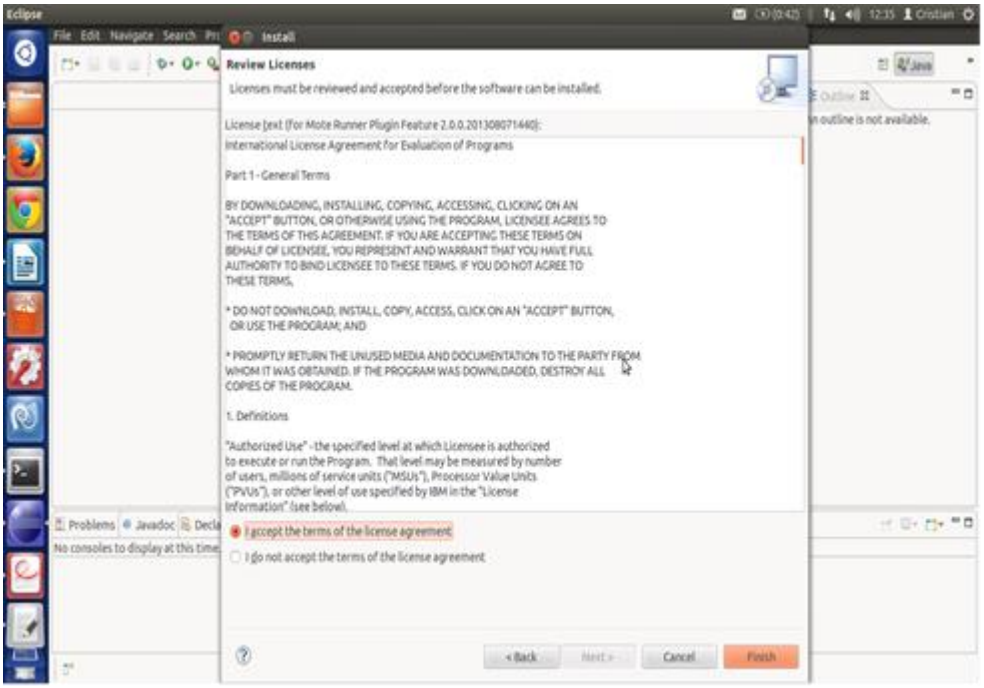

Figura A-49 Licencia de acuerdo del plugin

Aparecerá un mensaje en la pantalla con la indicación de que se va a instalar un software de contenido desconocido; sin embargo, este mensaje es ignorado y se pulsa el botón *Ok*.

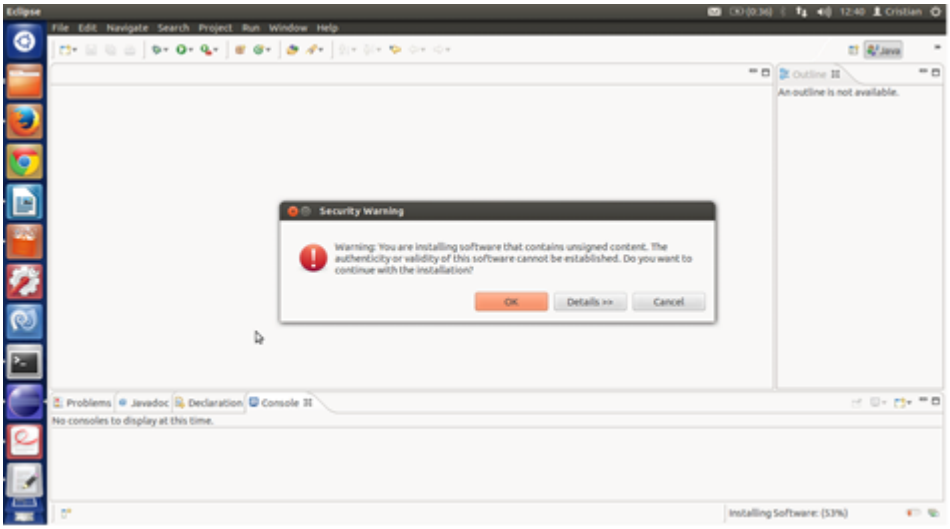

Figura A-50 Finalización de la instalación del plugin

Se finaliza la instalación y la misma solicitará que se debe realizar un reinicio del software Elipse.

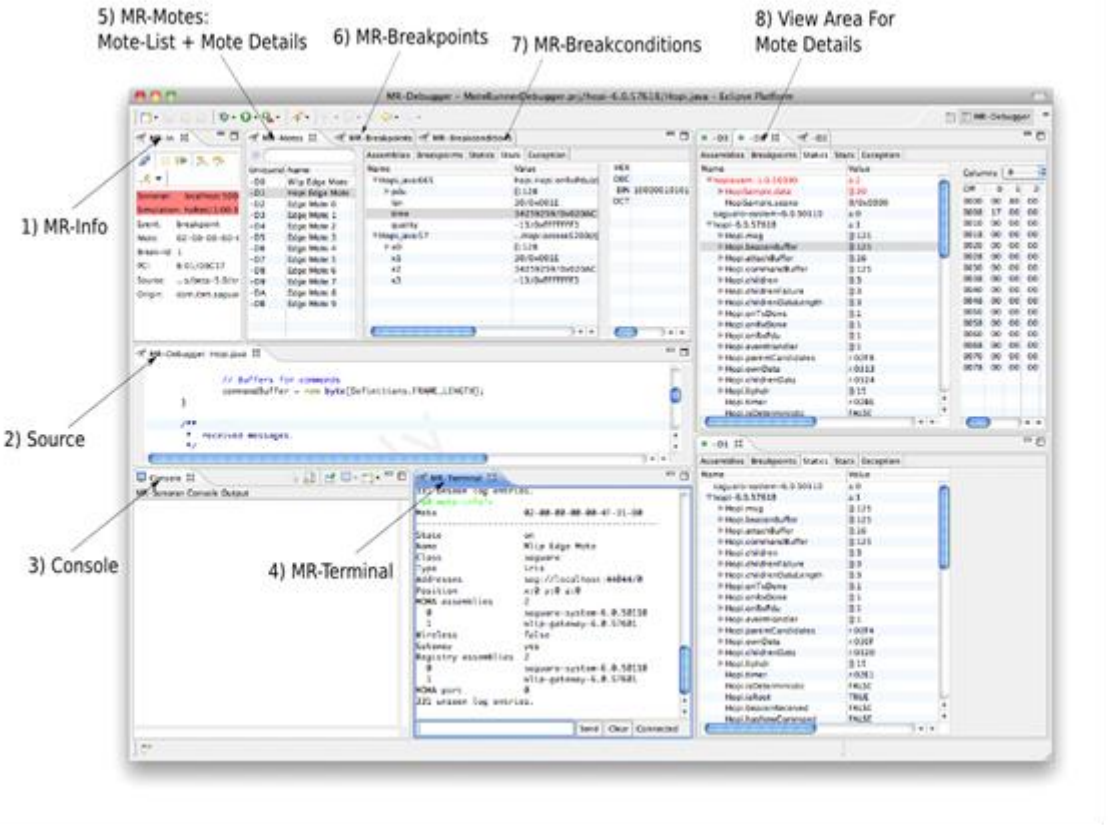

Figura A-51 Explicación del plugin para depuración en Eclipse

PASOS PARA SIMULAR NODOS INALÁMBRICOS

- Eliminar todas las motas creadas.
- Eliminar el proceso saguaro que se estaba ejecutando.
- Iniciar el proceso saguaro con los siguientes comandos.

```
1  >saguaro-start
2  >saguaro-connect
```

Código A-27 Inicio del proceso Saguaro

- Crear una mota.

```
1  >mote-create -d waspmote
```

Código A-28 Creación del mote

- Con el comando *"cd"* se desplaza hacia el directorio donde están los archivos para cargar al nodo.

DESARROLLO DE UNA RED 6LOWPAN UTILIZANDO NODOS MOTERUNNER

Figura A-52 Red que se pretende implementar

Como se puede apreciar en la imagen, se va a desarrollar una red de sensores inalámbricos los cuales trabajan con 6LowPAN, con direccionamiento IPv6, todos los nodos van a enviar información al Gateway. La conexión desde la PC y el Nodo Gateway se realizarán físicamente y en esta se va a estar ejecutado un túnel, el cual va a encapsular paquetes IPv6 dentro de IPv4. Al extremo derecho solo se trabajará con direccionamiento en IPv6.

En este ejemplo el cliente enviará una petición mediante NetCat versión 6 hacia los nodos los cuales devolverán el valor de temperatura que se podrá visualizar en la consola.

NODOS FINALES

En los nodos finales se cargarán los siguientes programas:

- El programa o mejor dicho librería "*mrv-6-lib-1.0*" está ubicado en el siguiente directorio "*/moterunner/examples/mrv6/src/tmp*"
- La librería *wasp-common-1.0* ubicada en la siguiente dirección (dirección de descarga librerías Libelium moterunner-1,0 */src/com/libelium/common*)
- La librería wasp-rtc-1.0 ubicada en la siguiente dirección (dirección de descarga librerías libelium moterunner-1.0 */src/com/libelium/rtc*)
- El programa que se ha desarrollado para medir la temperatura, que en este caso tiene por nombre *SenseTemp-1.0*, el código se muestra a continuación:

```
1   namespace com.ibm.saguaro.gateway.generic {
    using com.ibm.saguaro.system;
    using com.ibm.saguaro.mrv6;
    using com.ibm.saguaro.util;
    using com.libelium.rtc;
    using com.libelium.common;
       public class ReplySocket : UDPSocket {
           internal static uint LOCAL_PORT = 1024;
    internal static ReplySocket socket = new ReplySocket();

    internal static ADC adc;
    public ReplySocket() {
                this.bind(LOCAL_PORT);

    adc = new ADC();

    adc.open(0x01,GPIO.NO_PIN,0, 0);
    }
    public override int onPacket(Packet packet) {
                // An UDP packet has reached this mote
                // Toggle LED 2
                LED.setState(2, (byte)(LED.getState(2)^1));
                // We read ADC channel 0
                          //uint adcRead = adc.readChannel(0);
                // We store the ADC reading on a byte array
                byte[] buf = new byte[2];
                //Util.set16be(buf,0,adcRead);
                // ADC value hexadecimal representation is transformed to ASCII.
                // Doing so we can see the hexa value directly on Netcat
                          uint temp = (uint)RTC.getInstance().getTemperature();
                          byte aux = (byte)temp;
                          buf[0] = aux;
                buf = adc2Ascii(buf);
                uint len = packet.payloadLen;
    try {
    packet.swap(len);
    } catch {
    return 0;
    }
                Util.copyData(buf, 0, packet.payloadBuf, packet.payloadOff, 4);
    this.send(packet);
    return 0;
    }



    private byte[] adc2Ascii(byte[] adcValue){
                byte[] aux = new byte[4];
```

```
            aux[0]=(byte)((adcValue[0] & 0xF0) >> 4);
            aux[1]=(byte)(adcValue[0] & 0x0F);
            aux[2]=(byte)((adcValue[1] & 0xF0) >> 4);
            aux[3]=(byte)(adcValue[1] & 0x0F);


            for(int i=0;i<4;i++){
                if(aux[i] < 9){
                 aux[i] += 48;
}
                                else{
                    aux[i] += 55;
}
}
return aux;

}
```

Código A-29 Programa para devolver la temperatura

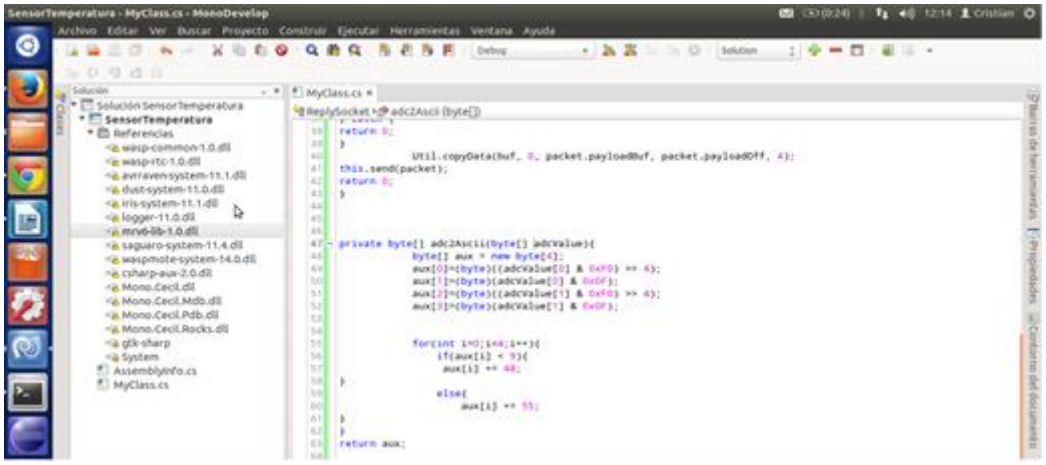

Figura A-53 Programa Sense

Figura A-54 Conexión con el nodo

Figura A-55 Carga de las librerías mrv6-lib y wasp-common

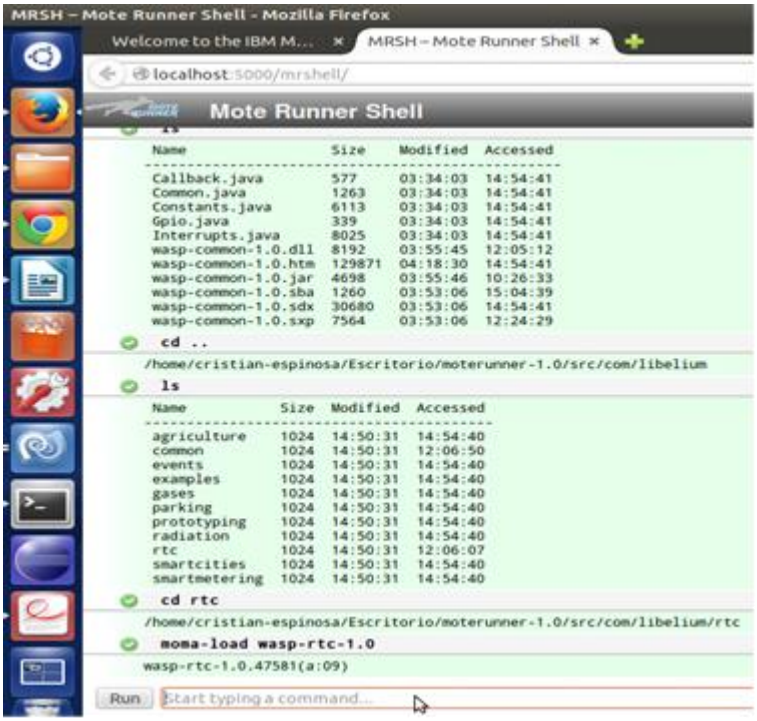

Figura A-56 Carga de la librería wasp-rtc

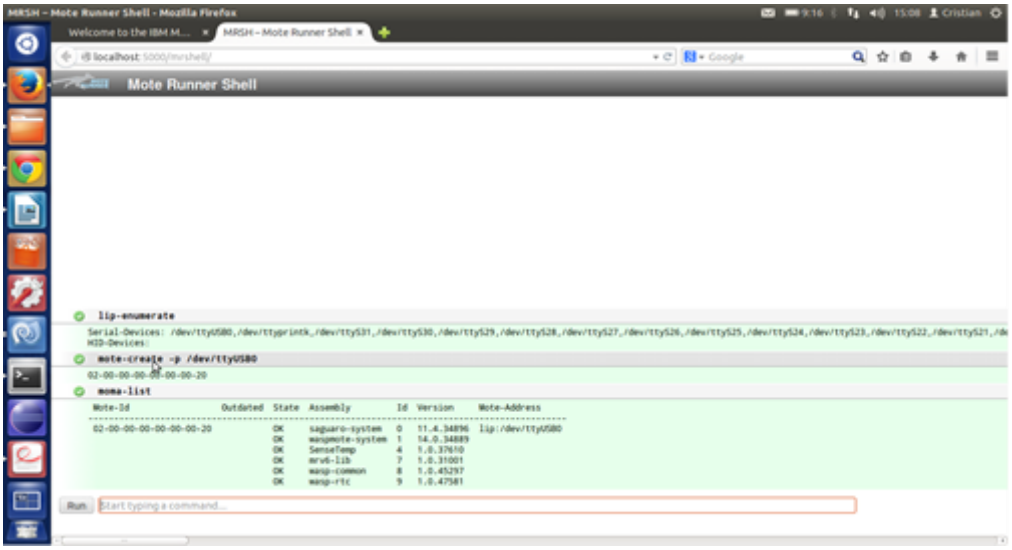

Figura A-57 Carga del programa SenseTemp

NODO DE BORDE GATEWAY

En el nodo Gateway o de borde se instalará el siguientes programa: "*mrv6-edge-1.0*" que se encuentra en la dirección (*/moterunner/examples/mrv6/src/tmp*).

Después de cargar esta librería, para poder acceder al nodo puesto el módulo LAN, se debe asignar una dirección IPv4 al nodo, en vista de que ahora se va a realizar la conexión a la PC a través del cable UTP con terminal RJ-45 y por lo tanto, no se utiliza el cable USB a la PC. Este proceso se realiza mediante el comando que se ejecuta en el servidor MRSH.

```
moma-ipv4 --ip 192.168.1.223 --subnwMask 255.255.255.0 --gateway 192.168.1.1 --udp
9999
```

Código A-30 Configuración de una dirección IPv4 al nodo Gateway

Como se puede ver en la Figura A-59, se indica que los cambios realizados serán efectuados al momento de reiniciar el nodo. Ahora bien, ya se puede desconectar el cable USB y colocar los módulos de radio frecuencia y de LAN para Gateway.
En este instante se conecta el nodo vía cable UTP a la computadora, para lo cual, a nuestra PC, se tiene que asignarle la dirección IPv4 que se utiliza como dirección *Gateway* del nodo. En nuestro caso sería:

- Dirección IP: 192.168.1.1
- Máscara: 255.255.255.0

Figura A-58 Cargando la librería mrv6-edge en el nodo Gateway

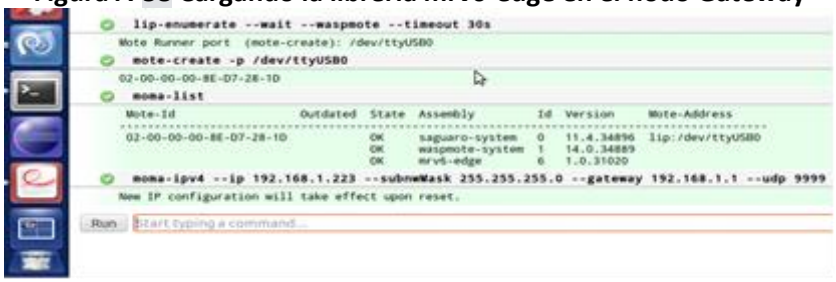

Figura A-59 Configuración de la dirección IPv4 al nodo Gateway

Figura A-60 Configuración de una dirección IPv4 en la PC

Una vez hecho esto, para que se efectúen los cambios realizados se ejecuta en una terminal el siguiente comando:

```
root@ubuntu:/home/cristian-espinosa/Escritorio# ifconfig etho down
```

Código A-31 Comando para poner shut en la interfaz eth0

Ahora se debe esperar un tiempo hasta visualizar que la interfaz cableada se haya apagado).

```
root@ubuntu:/home/cristian-espinosa/Escritorio# ifconfig etho up
```

Código A-32 Comando para levantar la interfaz eth0

Figura A-61 Red cableada desconectada

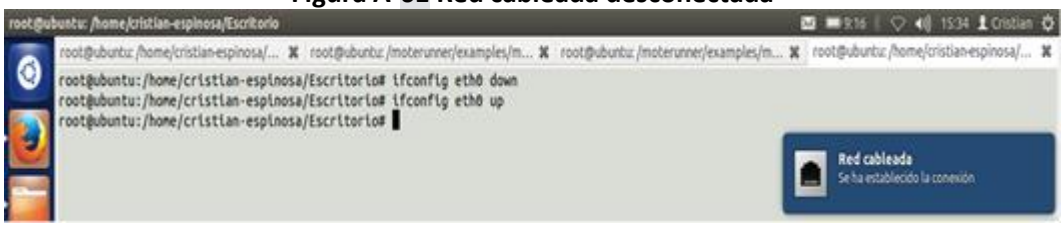

Figura A-62 Red cableada conectada

Ahora para poder visualizar que los cambios se llevaron a cabo, se efectúa en una terminal el siguiente comando:

```
root@ubuntu:/home/cristian-espinosa/Escritorio# ifconfig
```

Código A-33 Comando para verificar la configuración de red en las interfaces de la PC

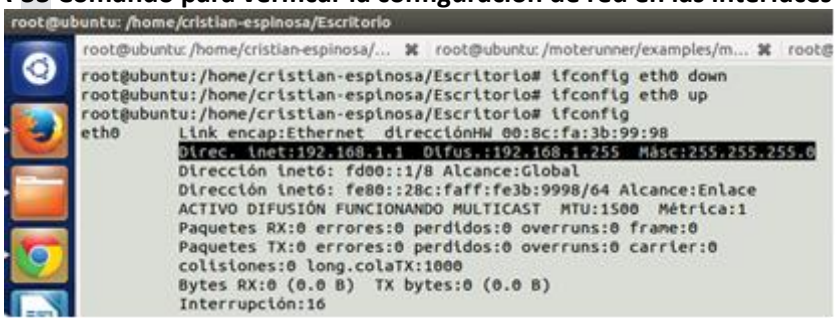

Figura A-63 Verificación de la configuración IPv4 en la red cableada

Ahora se verificará que existe conectividad con el nodo Gateway usando el comando ping en una terminal a la dirección IPV4, la cual fue asignada nodo Gateway:

```
root@ubuntu:/home/cristian-espinosa/Escritorio# ping
192.168.1.223
```

Código A-34 Verificación de conectividad con el Nodo Gateway mediante IPv4

Si el ping es exitoso se verá parpadear el led naranja del módulo LAN del nodo Gateway.

Figura A-64 Verificación de conectividad IPv4 con el nodo Gateway

Ahora se procede a conectar el nodo por medio del *Moterunner* para esto se debe tener ejecutando el servidor web y abrir un Shell:

```
1  mote-create -i 192.168.1.223
```

Código A-35 Conexión con el nodo Gateway mediante su dirección IPv4

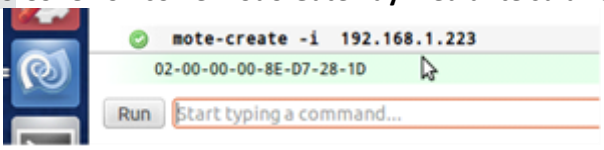

Figura A-65 Conexión al nodo Gateway mediante su dirección IPv4

Ahora se configura el túnel para esto se deberán tener abiertas tres pestañas de una terminal, en la primera pestaña debe estar ejecutándose servidor *MRSH*, en la siguiente pestaña se ejecutará la aplicación para la PC del túnel "que se encuentra en la dirección */moterunner/examples/mrv6/tunnel"* y para poder ejecutar dicho aplicativo se utiliza el comando que se muestra a continuación:

```
1  root@ubuntu:/moterunner/examples/mrv6/tunnel# ./tunnel
```

Código A-36 Ejecución del túnel

Figura A-66 Inicio del túnel

Ahora en el shell del moterunner, se debe dirigir a la siguiente dirección:

```
1  cd /moterunner/examples/mrv6/lib/js
```

Código A-37 Directorio donde se encuentra el JavaScrit

Se ejecuta el siguiente comando del Código A-38:

```
1  source mrv6.js
```

Código A-38 Carga del JavaScript

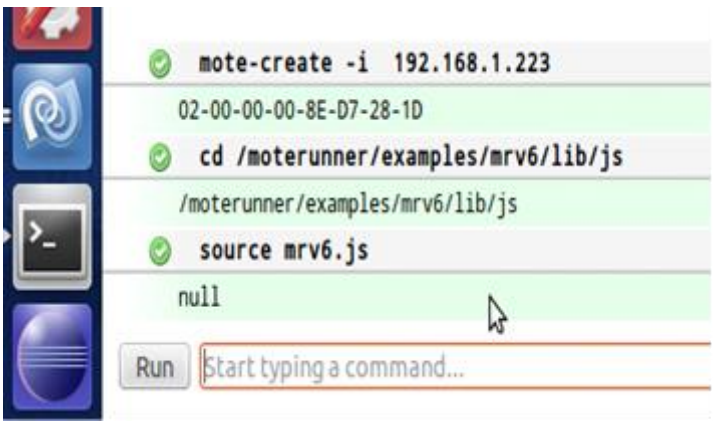

Figura A-67 Inicio del JavaScript para la configuración de la red 6LowPAN

Ahora se debe dirigir a la siguiente dirección:

"cd /moterunner/examples/mrv6/src/tmp"

Se ejecuta el siguiente comando para configurar el túnel y las direcciones de todos los nodos que conforman la red 6LowPAN:

```
1  u0 v6-setup --MAX_DEPTH=12 --NUM_CHILDREN=5 --MAX_CHILDREN=5 --MAX_MOTES=6 --
   RECV_SAFETY_MILLIS=3 --R24_SLOT_RCV_MILLIS=12 --R24_SLOT_GAP_MILLIS=15 --
   R24_BEACON_GAP_MILLIS=20 -INFO_INTERVAL_CNT=0
```

Código A-39 Comando que permitirá configurar la red 6LowPAN

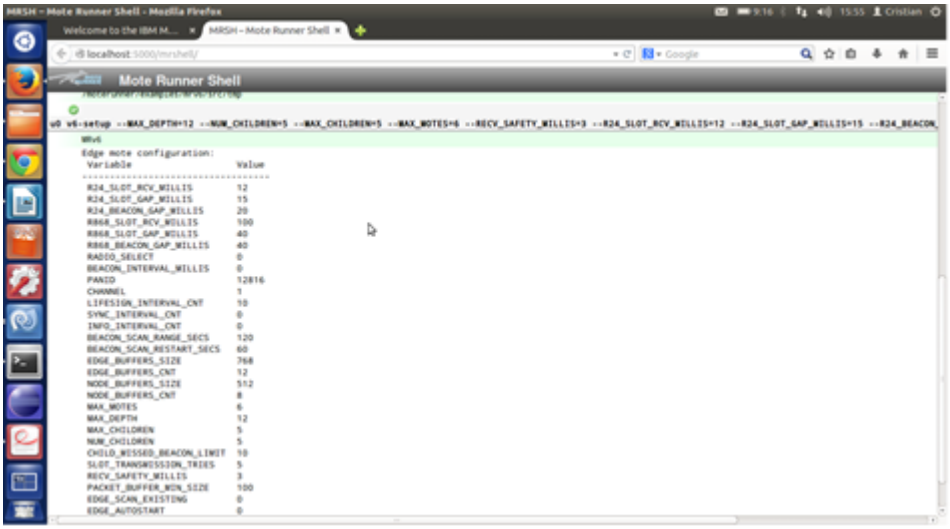

Figura A-68 Red 6LowPAN configurada con algunos parámetros

En la terminal donde está corriendo el tunnel se puede apreciar que todos los nodos han sido añadidos incluso hasta el nodo Gateway.

Figura A-69 Nodos añadiéndose a la red 6LowPAN

Para el establecimiento de las rutas y la configuración del tunnel es importante dirigirse a la siguiente dirección:

```
root@ubuntu:~# cd
/moterunner/examples/mrv6/tunnel/
```

Código A-40 Comando para cambiarme de dirección donde se encuentra el túnel

Se ejecuta el siguiente archivo de configuración:

```
root@ubuntu:/moterunner/examples/mrv6/tunnel# ./route_setup_linux_ip6.sh
```

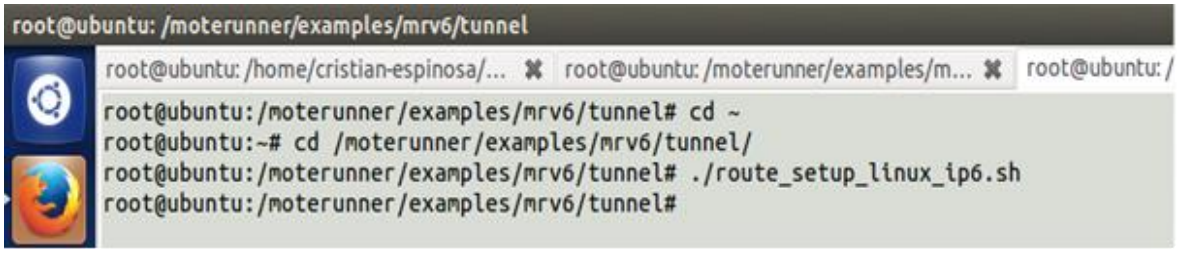

Figura A-70 Configuración del túnel el cuál va a dar el direccionamiento IPv6 a la red 6LowPAN

Con esto se puede observar en la pestaña donde se ejecuta la aplicación tunnel que las direcciones Ipv6 fueron asignadas, tanto al nodo Gateway como a los nodos inalámbricos. Para poder visualizar en el *Moterunner Shell* las direcciones asignadas se utiliza los comandos:

- network-list
- v6-connect

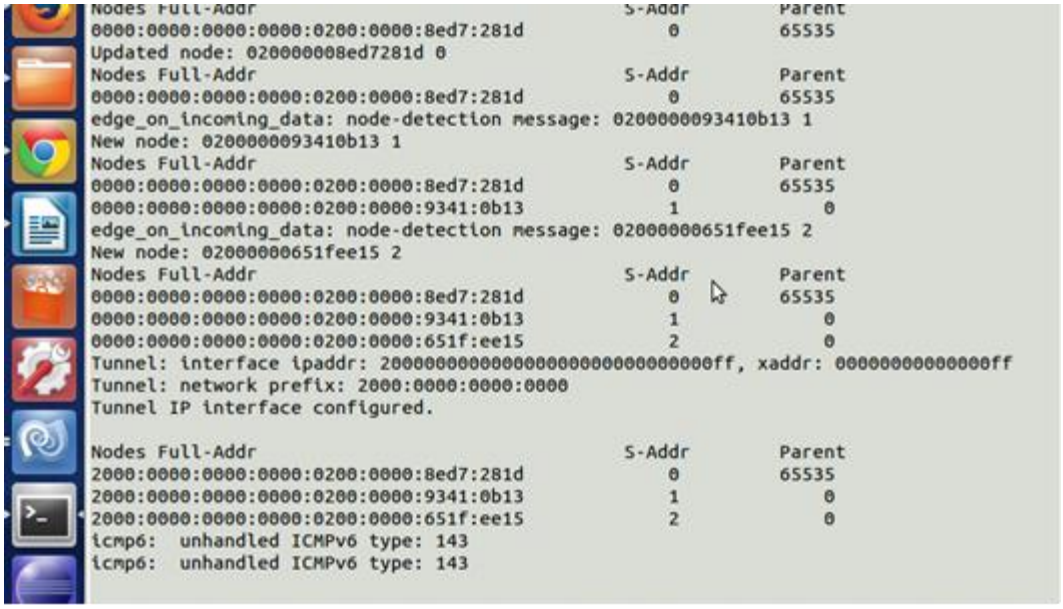

Figura A-71 Direccionamiento IPv6 a la red 6LowPAN

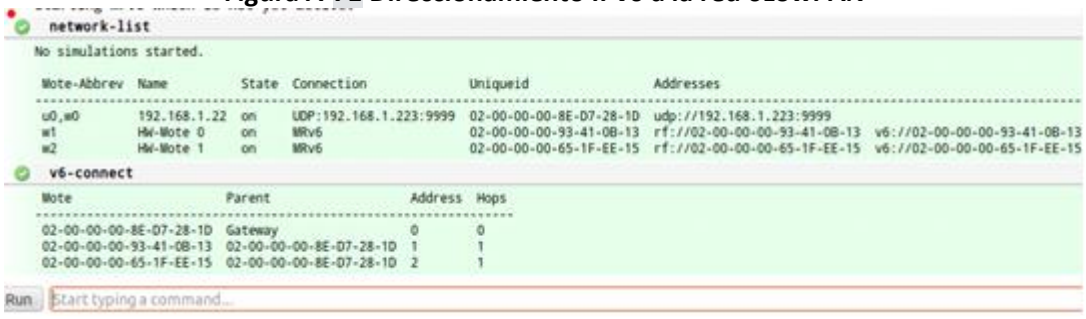

Figura A-72 Comandos "network-list" y "v6-connect"

Ahora desde la PC se puede hacer ping tanto al nodo Gateway como a los nodos inalámbricos que conforman la red 6LowPAN:

```
1   root@ubuntu:/moterunner/examples/mrv6/tunnel#ping6
    2000:0000:0000:0000:0200:0000:8ed7:281
```

Código A-41 probando conectividad IPv6 con la red 6LowPAN

```
1   root@ubuntu:/moterunner/examples/mrv6/tunnel# ping6
    2000:0000:0000:0000:0200:0000:651f:ee15
```

Código A-42 Comprobando conectividad IPv6 con la red 6LowPAN

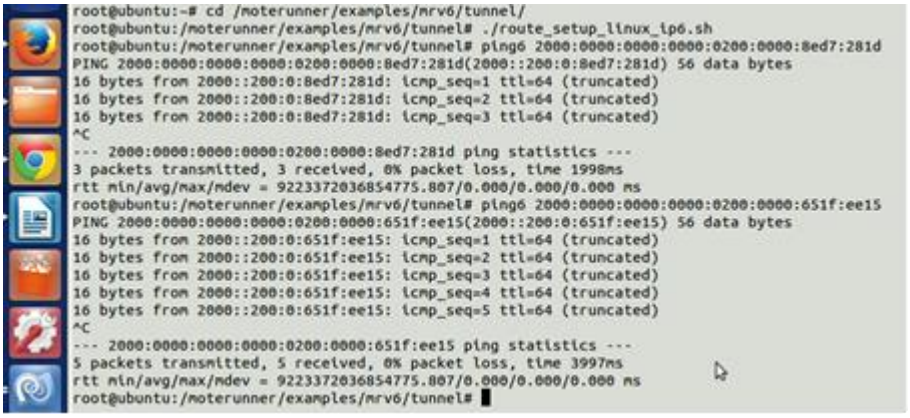

Figura A-73 Verificación de conectividad IPv6 con toda la red 6LowPAN

Cabe recalcar que el archivo route_setup_linux_ip6.sh fue modificado para obtener direcciones que puedan ser publicadas en el internet. Por defecto viene la dirección *fc00:db8:5::ff/8*, por lo cual es recomendable modificar esta dirección IPv6, específicamente en este caso se ha puesto con la dirección *2000::ff/8*, esta dirección no puede ser mayor a /48.

Posteriormente falta configurar una dirección IPv6 a la máquina donde correrá la aplicación del túnel, así como también en el lado de los computadores donde corre la aplicación diseñada.

Figura A-74 Configuración de una dirección IPv6 a la PC externa

Figura A-75 Configuración de una dirección IPv6 a la PC que ejecuta el túnel

En este caso se le asigna la dirección IPV6 *fd00:1/8* a la computadora donde corre el programa del túnel, así como la dirección IPv6 *fd00:2/8* al otro computador final. Ya establecidas las direcciones IPv6 es posible hacer una comprobación de conectividad por medio del comando “ping6”.

Ping desde aplicación hasta el Gateway:

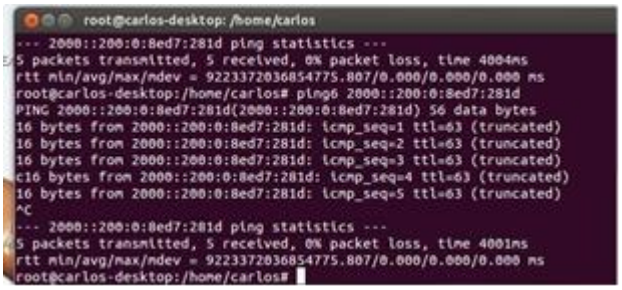

Figura A-76 Conectividad extremo a extremo

Ping desde aplicación hacia los nodos:

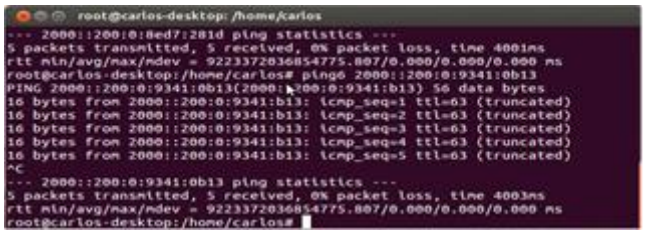

Figura A-77 Verificación de conectividad extremo a extremo

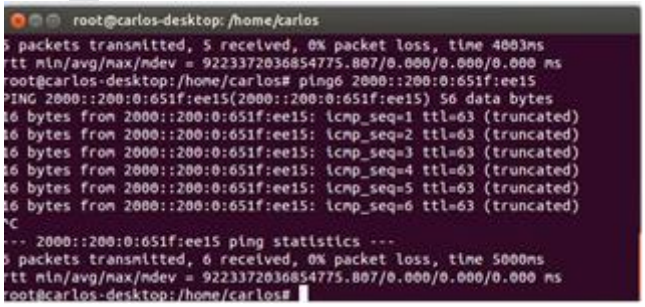

Figura A-78 Verificación de conectividad extremo a extremo

En este punto se tiene ya la conectividad tanto del lado de los nodos que conforman la red 6LowPAN hasta la PC donde se ejecuta el túnel, como de esta misma PC hasta a la otra. Sin embargo, todavía no se tiene una total comunicación total de extremo a extremo, para lo cual es necesario establecer en la misma instancia donde corre el túnel, una función de enrutamiento para poder especificar el túnel como interfaz de entrada/salida. Para ello se debe instalar la aplicación *radvd.*

```
root@ubuntu:/moterunner/examples/mrv6/tunnel# sudo apt-get install radvd
```

Código A-43 Aplicación que permitirá enrutar los datos de forma correcta

El archivo de configuración deberá ser creado con el comando *gedit /etc/radvd.conf* y deberá tener el siguiente contenido o parecido:

```
interface ath0
{
        AdvSendAdvert on;
        prefix fd00::/8
        {
                AdvOnLink on;
        };
};
```

Código A-44 Archivo de configuración radvd

En este archivo se puede configurar la dirección de red correspondiente a la configurada entre estas dos máquinas.

Con la siguiente instrucción se puede iniciar el demonio que corresponde al *radvd*. Esto es muy importante ya que una vez hecho demonio ya se ejecutará por sí solo.

```
1   root@ubuntu:/moterunner/examples/mrv6/tunnel# sudo /etc/init.d/radvd start
```

Código A-45 Comando para ejecutar como demonio la aplicación radvd

ANEXO B

ELEMENTOS DEL KIT DE DESARROLLO MENSIC Y PROCESO DE INSTALACIÓN DE LOS SENSORES PARA SU CORRECTO USO

En la Tabla B-1, se detalla cada uno de los elementos del kit de desarrollo MENSIC utilizados junto con su propósito:

N°	Elemento	Propósito	Imagen
1	Placa de programación MIB520CB	Se utiliza para programar los nodos, tiene un conector de 51 pines. Conexión basada en USB	
2	nodos sensores Iris XM2110CB	Transmiten y reciben datos por radio. Tienen un conector de 51 pines para alojar algún tipo de sensor, además utilizan el estándar IEEE 802.15.4 de 2,4 GHz	
4	Nodo Gateway MIB520CB	Es La unión de la placa programadora con un nodo, con la diferencia que viene incrustado. Se lo utiliza como nodo Gateway ya que tiene	

		alimentación USB permanente.	

En esta segunda parte del anexo vamos a mostrar el proceso de instalación que se necesita realizar antes de ocupar los nodos sensores a continuación se mostrará paso a paso la forma de instalación de los programas que debemos utilizar.

INSTALACION DE JAVA 8

Antes de instalar TinyOS se debe tener una versión de Java instalada. Para instalar Java 8 en Ubuntu 14.04, se abre inicialmente una terminal con los comandos (CTRL + ALT + T) y a continuación se siguen los 4 pasos siguientes:

1. Añadir los repositorios:

```
$ sudo add-apt-repository ppa:webupd8team/java
```

2. Actualizar los repositorios:

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

3. Instalar Java desde su página:

```
$ sudo apt-get install oracle-java8-installer
```

4. Comprobar la funcionalidad de la versión instalada:

```
$ java –versión
```

INSTALACIÓN DE TINYOS 2.1.2

A continuación, se detalla paso a paso las modificaciones hechas en Ubuntu 14.04 LTS para instalar la versión 2.1.2 de TinyOS.

PASO 1: En Ubuntu

```
$ Sudo gedit / etc / apt / sources.list
```

Primero, se accede al repositorio de Ubuntu ejecutando lo siguiente en una terminal:

Se abre el archivo y allí se copia lo siguiente:

deb http://tinyos.stanford.edu/tinyos/dists/ubuntu lucid main
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu natty main
deb http://tinyos.stanford.edu/tinyos/dists/ubuntu maverick main
Posterior al paso anterior, aparece una imagen como la que se muestra a continuación:

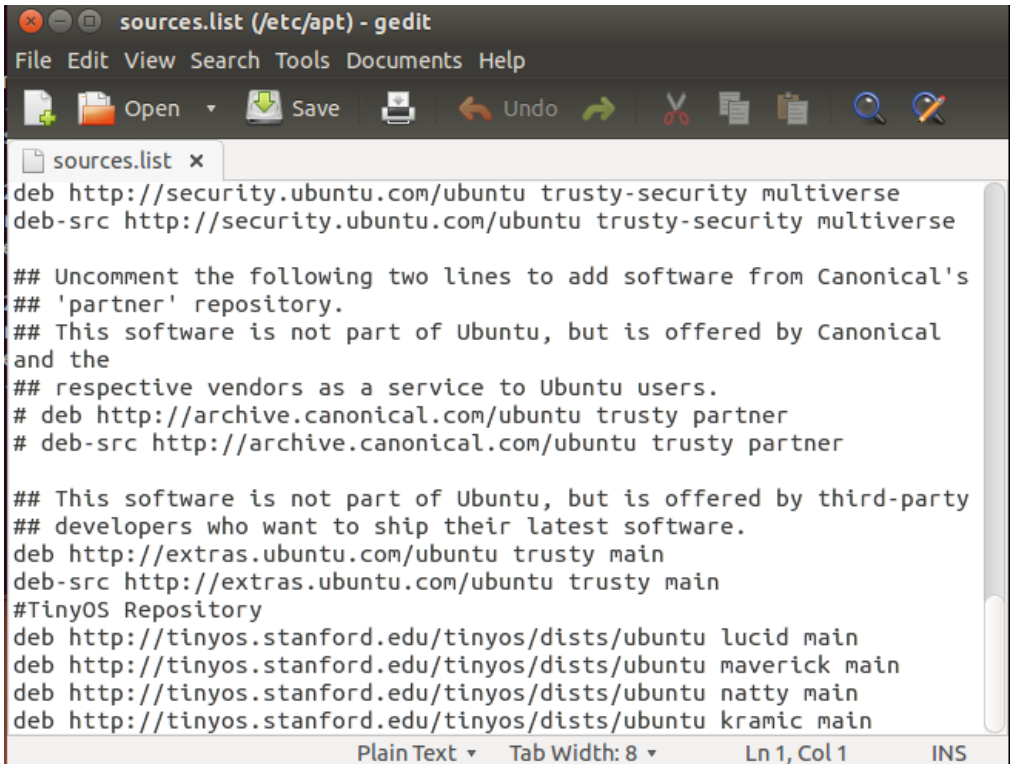

Figura. B-1 Agregación de repositorios de TinyOS

PASO 2: Actualización

Posteriormente, se actualizan las modificaciones realizadas mediante un update:

```
$ Sudo apt-get update
```

PASO 3: instalación de TinyOS

Se instala TinyOS 2.1.2 desde su página utilizando:

```
$ Sudo apt-get install TinyOS-2.1.2
```

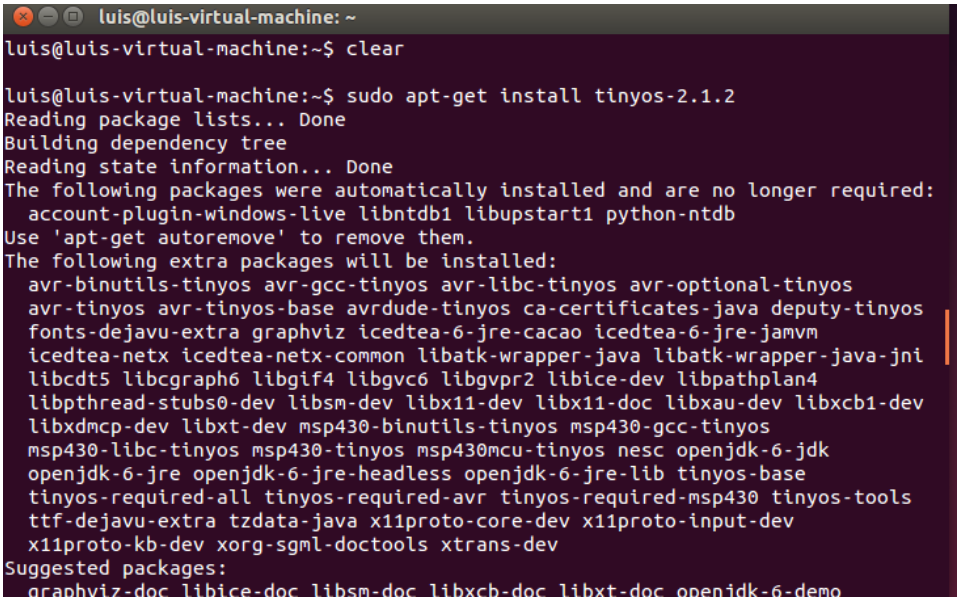

Figura. B-2 Instalación de TinyOS

Paso 4: Otorgar permisos de usuario a TinyOS

Se otorga los respectivos permisos a la carpeta creada en la instalación.

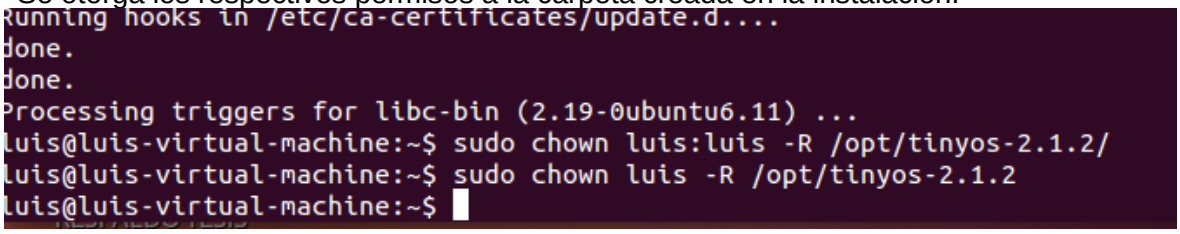

Figura. B-3 Permiso a la carpeta TinyOS

Paso 5: Situarse en la Carpeta TinyOS 2.1.2

Dentro de la carpeta, se crea un archivo “tinyos.sh” colocando el CLASSPATH correcto dentro de este, para ello se ejecuta:

```
sudo gedit / opt/tinyos-2.1.2/tinyos.sh
```

Dentro del texto debe aparecer lo siguiente:

```
#! /usr/bin/env  bash
# Here we setup the environment
# variables  needed by the tinyos
# make system

echo "Setting up for TinyOS 2.1.2 Repository Version"
export TOSROOT=
export TOSDIR=
export MAKERULES=

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


export TOSROOT
export TOSDIR
export CLASSPATH
export MAKERULES
```

Como se muestra en la siguiente figura:

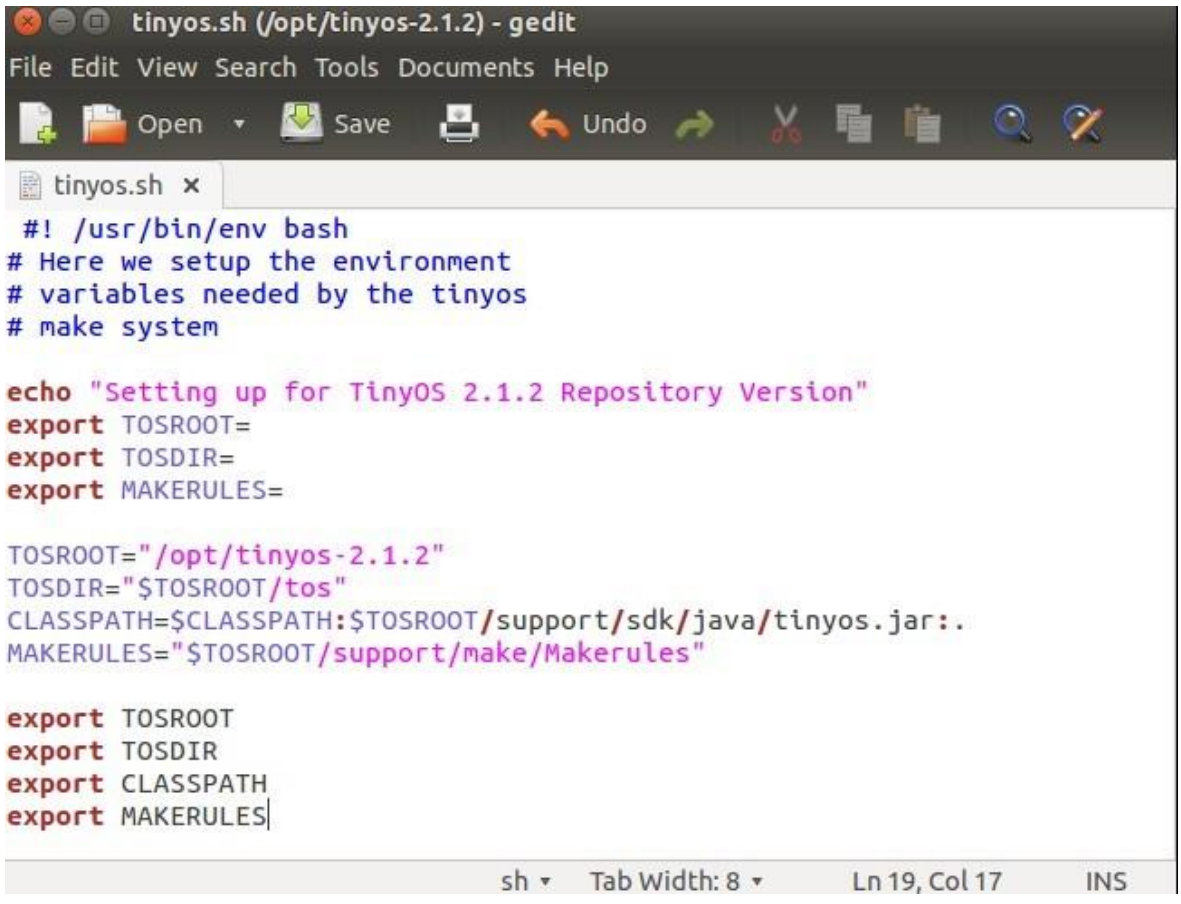

Figura. B-4 Llenado de archivo tinyos.sh

Paso 6: más modificaciones
Se modifica el bashrc para exportar todos los ejecutables de TinyOS a través del siguiente comando:

```
$ Sudo gedit ~ /. bashrc
```

Dentro del archivo se agrega lo siguiente:

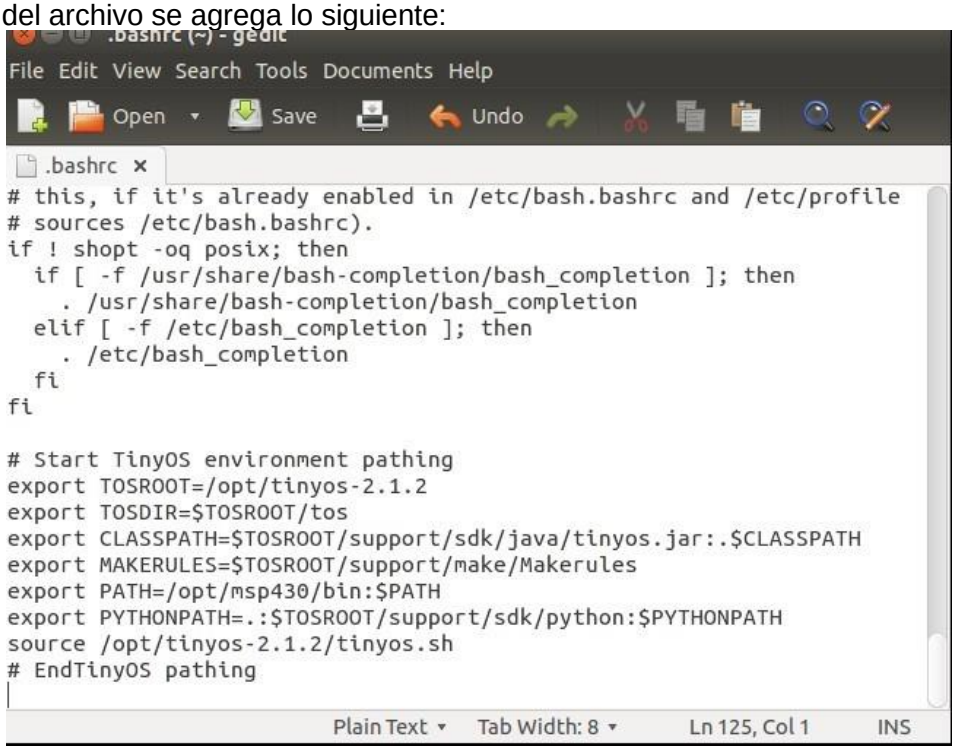

Figura. B-5 Agregación de reglas para exportar ejecutables de TinyOS

Se guarda y se aplican los cambios en el bashrc:
$ source ~/.bashrc
Para comprobar que la instalación se realizó de manera correcta, se verificará la aparición del siguiente mensaje:

```
#Sourcing the tinyos environment variable setup script
source /opt/tinyos-2.1.2/tinyos.sh
```

Paso 7: verificamos instalación

```
$ tos-check-env
```

Se observa si la instalación tiene algún error, mediante:

Dicho código debe aparecer como se muestra en la figura B-6 al ser ingresado:

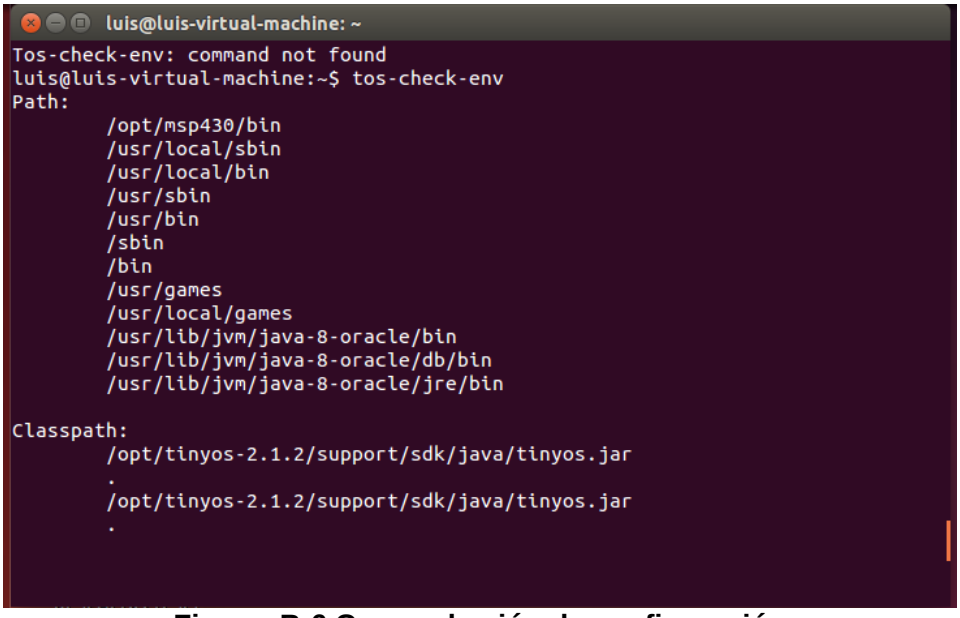

Figura. B-6 Comprobación de configuración

Pasó 8: Instalamos las herramientas de Java que vienen con TinyOS
Situarse en la carpeta de java que viene por defecto al instalar tinyos:

```
$ cd /opt/tinyos-2.1.2/support/sdk/java/
```

Se Instala la herramienta con los siguientes comandos:

```
$ sudo tos-install-jni
```

Debe aparecer el mensaje mostrado en la imagen:

Figura. B-7 Directorio java en TinyOS

Se compila la herramienta java mediante:

```
$ sudo make install
```

Paso 9: Ejecución de un ejemplo

Para ello se sitúa en la carpeta de apps de TinyOS:

```
$ cd /opt/tinyos-2.1.2/apps/
```

Se compila cualquier aplicación que esté dentro del repositorio de apps, para lo cual se escoge la aplicación Blink, que es una de las más básicas; con el siguiente comando se procede a compilar la aplicación:

```
$ cd /opt/tinyos-2.1.2/apps/Blink/ $ make iris
```

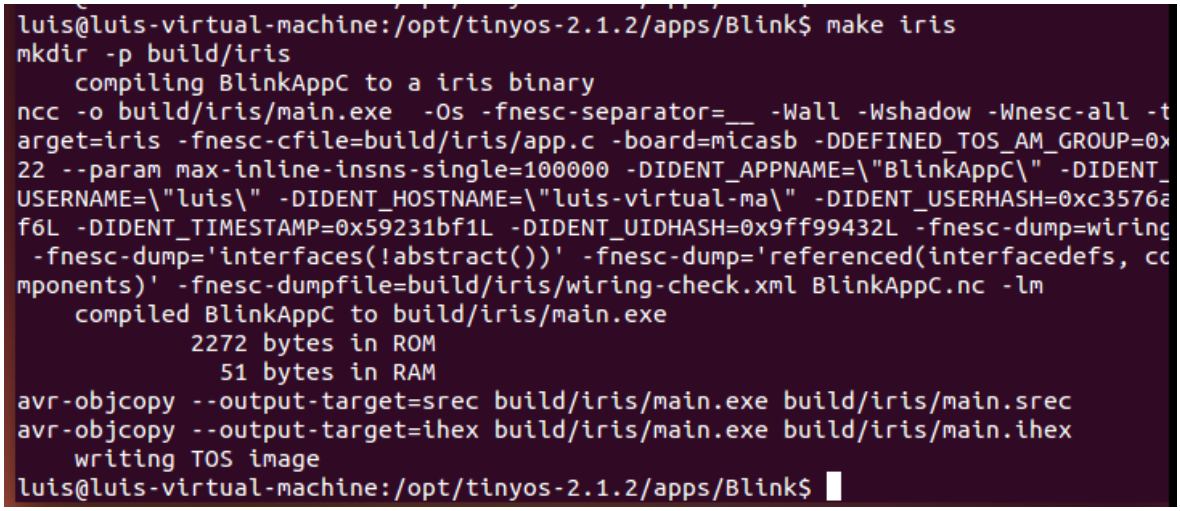

Figura. B-8 Compilación de aplicación Blink

Como se puede observar en la figura B-8, se utiliza “make iris” ya que los nodos sensores utilizados en este proyecto son iris; sin embargo, pueden ser compilados para algunos tipos de nodos que se encuentren en el repertorio de TinyOS.

INSTALACIÓN DE ECLIPSE CON YETI2 PARA DESARROLLO DE NUEVOS PROYECTOS

Paso 1: En la página http://eclipse.org/eclipse/ se puede descargar su versión para Linux (en este caso Eclipse 4.4 Luna).

Paso 2: Se extrae el contenido del archivo descargado en la propia carpeta de descargas.

Paso 3: Se sitúa la carpeta extraída en descargas a la carpeta “/opt”. Para ello, se da los respectivos permisos.

Paso 4: Dentro de esa carpeta existe un archivo ejecutable con el nombre “eclipse”. Se puede crear un enlace de este archivo en el escritorio y renombrar de acuerdo a criterios personales.

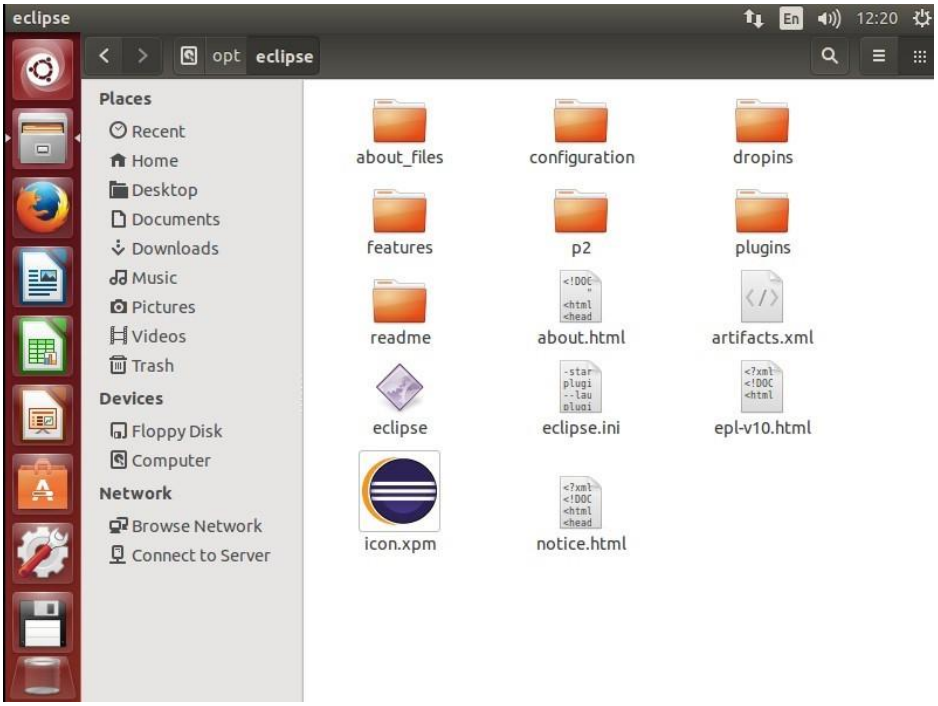

Figura. B-9 Carpeta Eclipse en TinyOS

Paso 5: Se crea un fichero desktop para tener un acceso directo para eclipse.

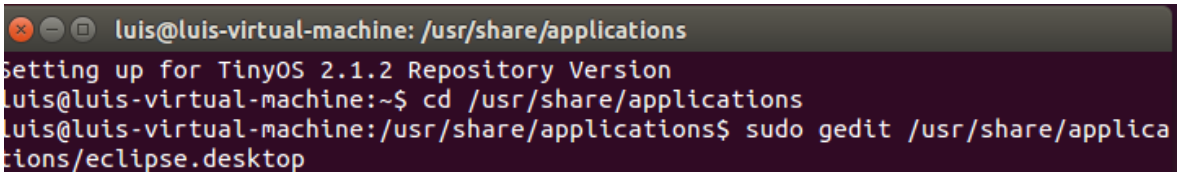

Figura. B-10 Creación de fichero eclipse.desktop

Paso 6: Posteriormente, se añade el código en el fichero que se ha creado, ejecutando las líneas anteriores:

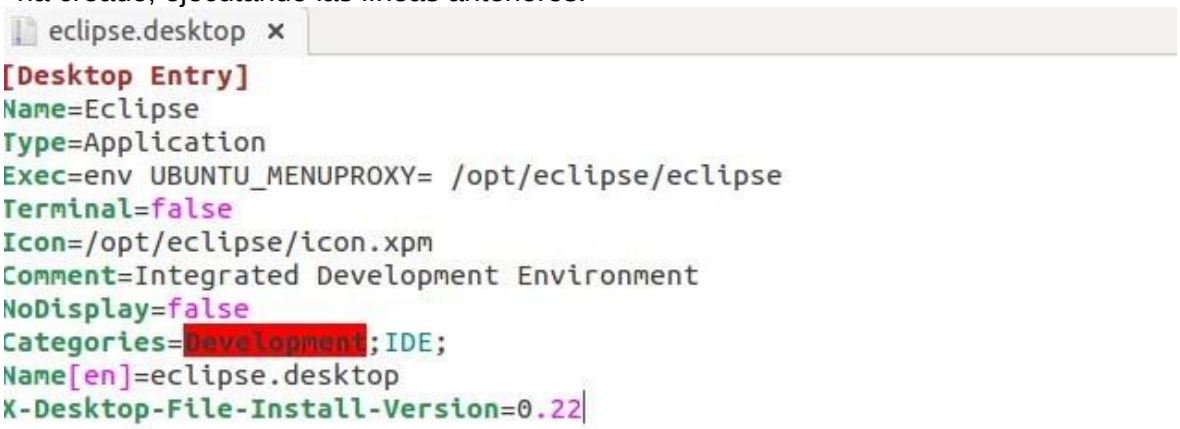

Figura. B-11 Fichero para crear icono de Eclipse

Con el siguiente comando se instala eclipse.desktop en el Unity:

```
$ sudo desktop-file-install /usr/share/applications/eclipse.desktop
```

Se crea un enlace simbólico al IDE de Eclipse con los siguientes comandos:

```
$ cd /usr/local/bin/
```

```
$ sudo ln -s /opt/eclipse/eclipse
```

Para que el ícono de Eclipse aparezca en el buscador de recursos de Ubuntu, se ejecuta el siguiente comando:

```
$ sudo cp /opt/eclipse/icon.xpm /usr/share/pixmaps/eclipse.xpm
```

Eclipse se encuentra instalado, para acceder al programa a través del ícono, se busca el mismo en aplicaciones y se lo arrastra a la barra de la izquierda.

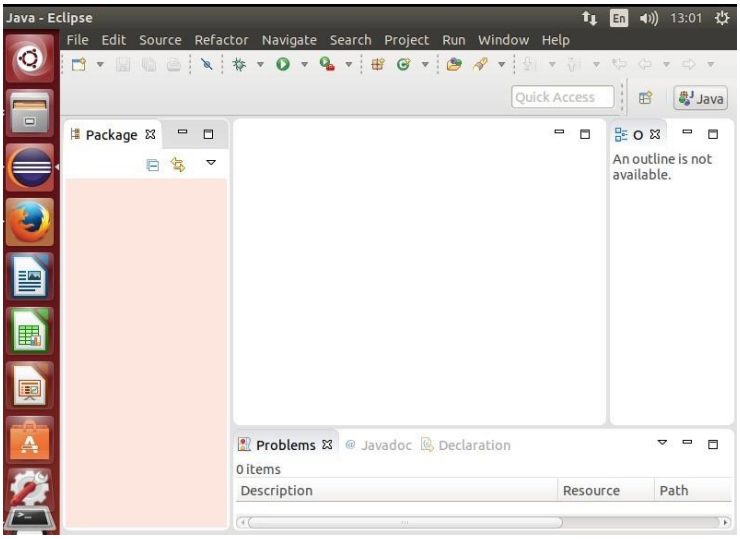

Figura. B-12 Eclipse ejecutado desde el icono

Paso 7: Instalación de yeti-2 para desarrollar aplicaciones para TinyOS. Para ello se sitúa en Help>Intall New Software y se escribe la página de descarga (ver figura B- 13).

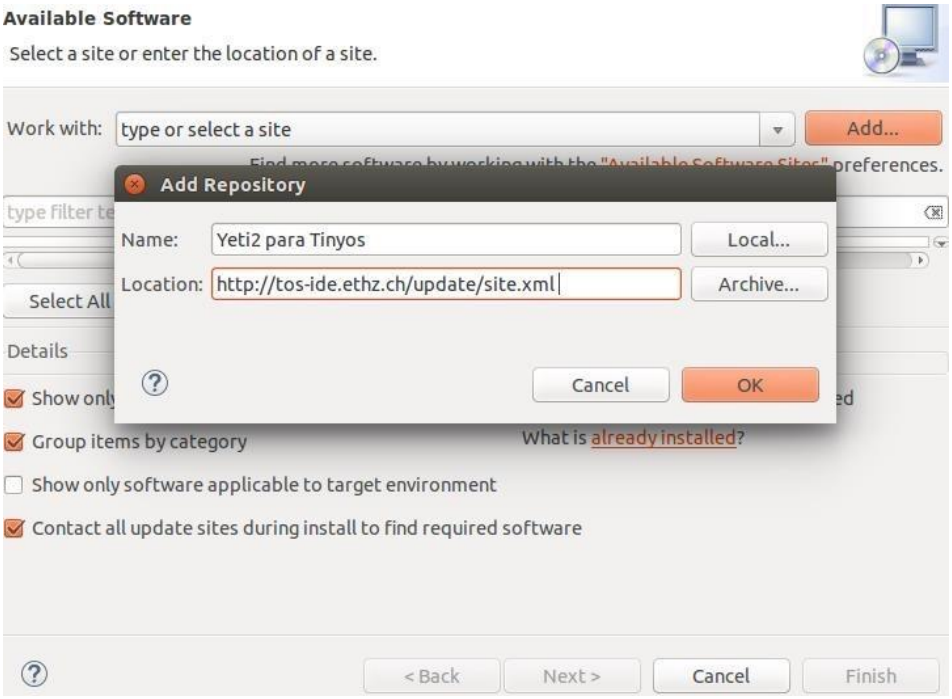

Figura. B-13 Instalación del repositorio Yeti-2

Paso 8: Una vez instalado yeti 2, se modifica la perspectiva para trabajar con ambiente TinyOS, para ello se da click en window>Open Perspective, para seleccionar la de TinyOS.

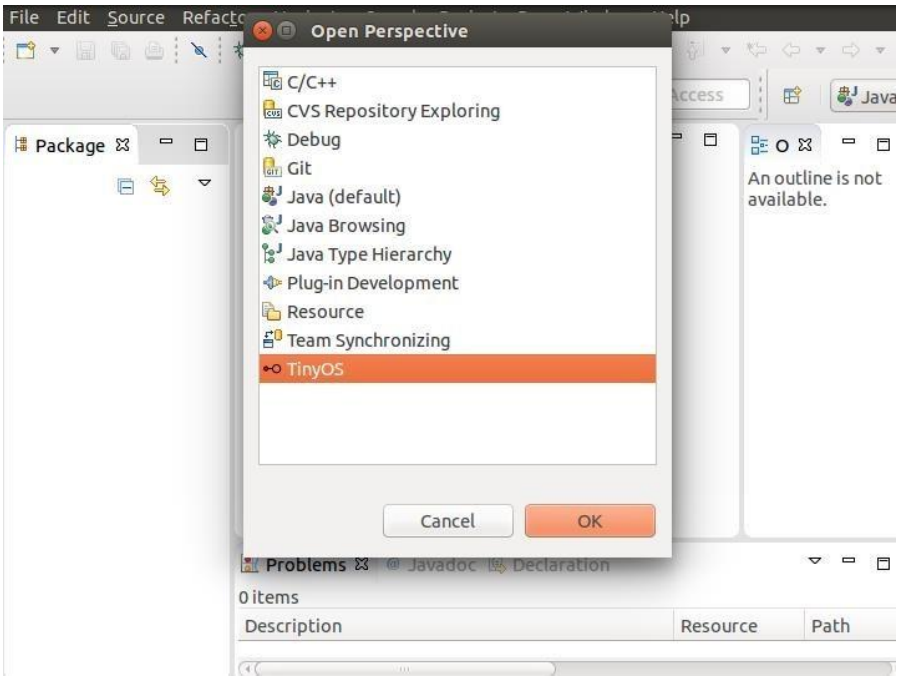

Figura. B-14 Selección de perspectiva TinyOS

Paso 9: Después de seleccionar la perspectiva, la aplicación eclipse se encuentra lista para crear nuevas aplicaciones con TinyOS.

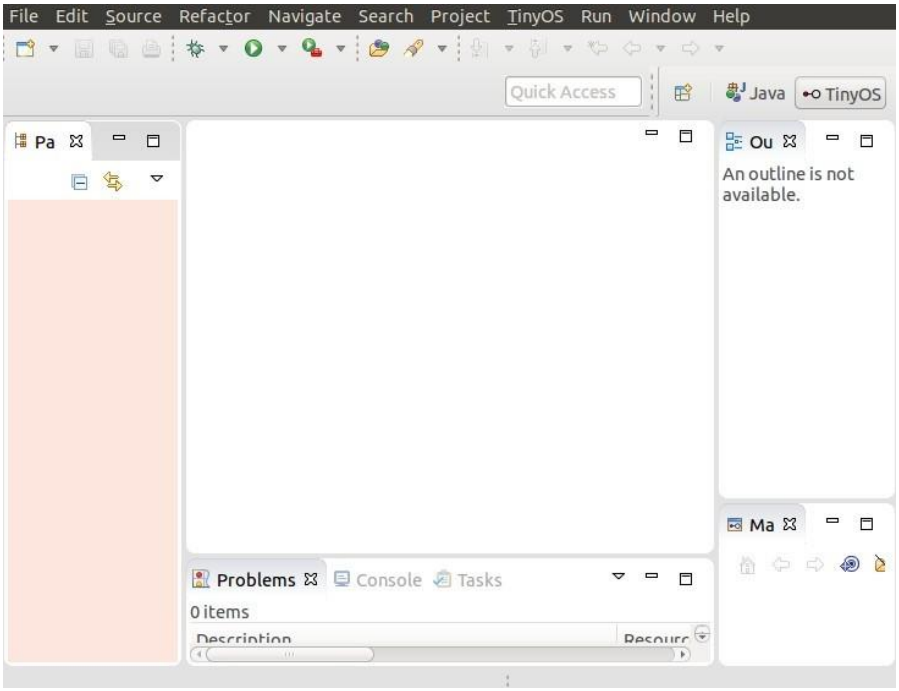

Figura. B-15 Perspectiva TinyOS

INSTALACIÓN DE UN APLICATIVO EN LOS NODOS SENSORES

Para que los nodos sensores cumplan con lo requerido se tiene que compilar la aplicación envió datos. Lo primero que se debe hacer es quitar la placa censora del nodo IRIS en estado OFF e incrustarlo en la placa programadora a través del conector de 51 pines (Ver figura A-16).

Figura. B-16 Conexión del nodo sensor a la placa programadora

Luego se ubica la aplicación a cargar dentro de la carpeta creada en eclipse que en este caso se llama envía datos (ver figura B-17)

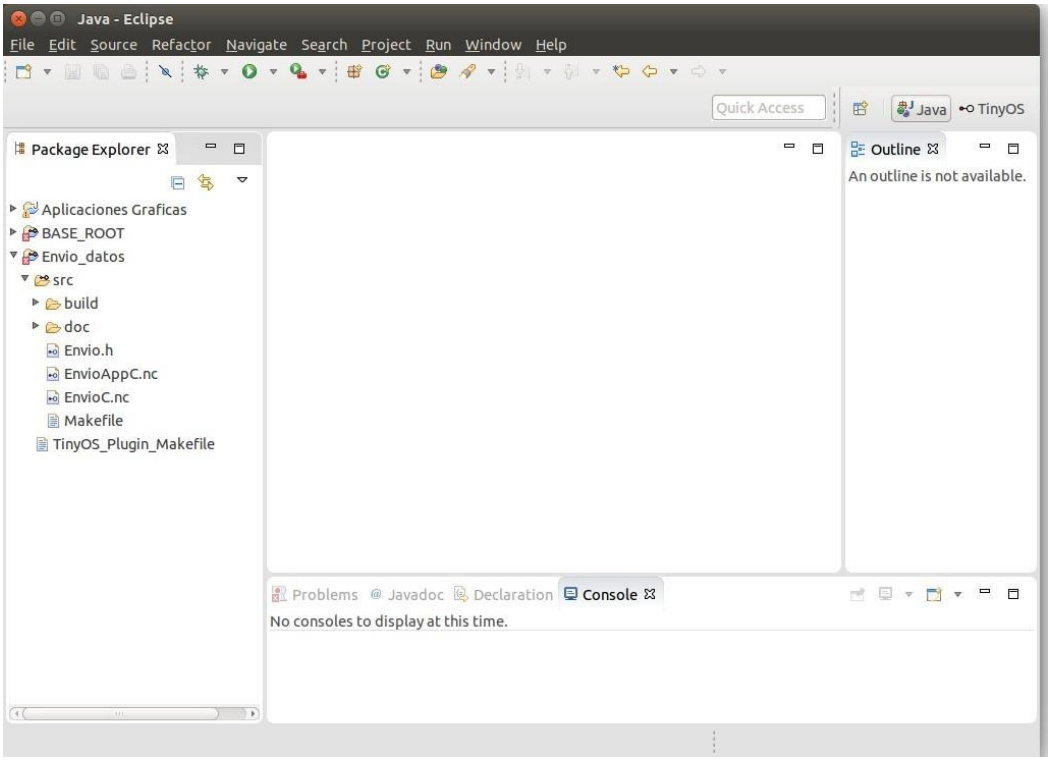

Figura. B-17 Aplicación Envía datos para nodos sensores

Para compilar la aplicación se debe dar los respectivos permisos a los puertos que tiene la placa programadora a través de:

```
$ sudo chmod 666 /dev/ttyUSB0
```

Después se instala el aplicativo mediante los respectivos comandos en una terminal:

```
$ make iris install, 16 mib520, /dev/ttyUSB0
```

Donde ID=16, representa el identificador que se da al nodo sensor, mientras que MIB520
es la placa programadora que permite transmitir el código seguido del puerto serial ttyUSB0 que es el asignado a la placa.

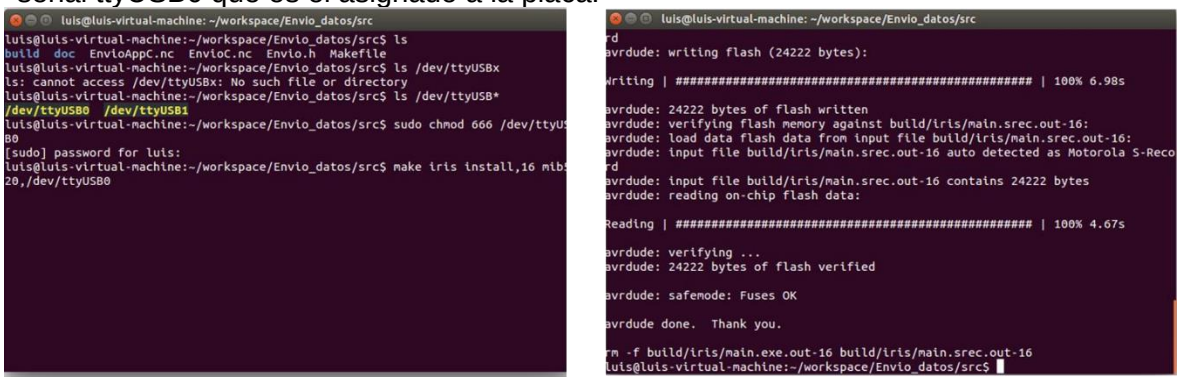

Figura. B-18 Compilación de la Aplicación Envía datos