Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Nodo de recolección y procesamiento de datos basado en GNU/Linux

Informe de Proyecto de Graduación para optar por el título de Ingeniera en Electrónica con el grado académico de Licenciatura

Carmen Chan Zheng

Cartago, Junio de 2012

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Proyecto de Graduación

Tribunal Evaluador

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniera en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Ing. William Marín M
Profesor lector

Ing. Johan Carvajal G
Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, Viernes 22 de Junio de 2012

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Cartago, 22 de Junio de 2012

Carmen Chan Zheng

Cédula: 1-1407-0838

Agradecimiento

En este espacio aprovecho para agradecer a personas que me han acompañado a través de todos estos años que he estado en el TEC para lograr los éxitos tanto académicos como personales.

A todos mis compañeros, que eventualmente se hicieron amigos de por vida, les agradezco toda la ayuda y el apoyo moral que me proporcionaron durante las noches de estudios y los días previos a las entregas de proyectos.

A mi tutor, que con su apoyo, su dedicación y su ayuda fue invaluable para alcanzar el éxito de mi proyecto de graduación.

A mis mejores amigos, que siempre me dieron su apoyo en todo.

A mi familia adoptiva, gracias a la calidez y el hogar que me brindaron desde que llegué hace más de 5 años.

En especial agradezco a mi padre, a mi madre y a mi hermana. Por todo el apoyo que me han brindado, por todas esas veces que estuvieron en apuros conmigo y por todas esas visitas para que me sintiera como en casa. Ellos fueron, son y serán mi base para alcanzar cualquier sueño y cualquier éxito, por lo tanto, les dedico este éxito a ustedes.

Como hija, como hermana y como amiga me siento dichosa de haber conocido a todas estas personas en mi vida.

Contenido

Índice de figuras 8
Índice de tablas 10
Resumen 11
Abstract 12
Capítulo I. Introducción 13
1.1 Antecedentes 13
1.2 Solución seleccionada 15
Capítulo II. Meta y Objetivos 16
2.1 Meta 16
2.2 Objetivo general 16
2.3 Objetivos específicos 16
Capítulo III. Marco teórico 17
3.1 Sistema actual 17
3.2 Módulos 18
3.2.1 Tarjeta de desarrollo 18
3.2.2 Red inalámbrica de sensores (Waspmote) 20
3.2.3 Módulo de recepción 23
3.2.4 Módulo de envío 23
3.3 Bibliotecas implementadas en el S.O Ubuntu 33
3.3.1. Biblioteca PPP 33
3.3.2. Biblioteca NTP 35
3.3.3. Bibliotecas de Python 36
3.3.4. Biblioteca Gammu 38
Capítulo IV. Procedimiento Metodológico 39
4.1 Reconocimiento y definición del problema 39
4.2 Obtención y análisis de información 39
4.3 Evaluación de las alternativas y síntesis de una solución 40
4.4 Implementación de la solución 40
4.5 Reevaluación y rediseño 42

Capítulo V. Descripción detallada de la solución 43
5.1 Análisis de soluciones y selección final 43
5.1.1 Plataforma de desarrollo 43
5.1.2 Sistema Operativo 44
5.1.3 Lenguaje de programación 44
5.1.4 Operadora de Red de telefonía celular 44
5.1.5 Redes de sensores inalámbricos 45
5.2 Descripción del hardware 45
5.2.1. Red de sensores 45
5.2.2. Gateway 46
5.2.3. BeagleBoard 47
5.2.4. Datacard 47
5.3 Descripción del software 48
5.3.1. Implementación del Sistema Operativo Ubuntu 48
5.3.2. Configuración Inicial del Sistema Operativo 49
5.3.3. Sistema principal 52
Capítulo VI. Análisis de Resultados 60
6.1. Implementación del Sistema Operativo 60
6.2. Conexión a Internet 61
6.3. Ancho de banda de la señal 63
6.4. Pruebas del sistema principal en el laboratorio 64
6.5. Pruebas del sistema principal en el puente 69
6.6. Alcance de los objetivos específicos 70
6.7. BeagleBoard xM vs. Computadora PC 71
Capítulo VII. Conclusiones y recomendaciones 72
7.1. Conclusiones 72
7.2. Recomendaciones 73
Referencias 74

Apéndices 77
A.1. Glosario y abreviaturas 77
A.2. Script de Instalación en *Bash* 78
Anexos 79
B.1. Especificaciones eléctricas de la tarjeta de desarrollo BeagleBoard [4] .. 79
B.2. Hoja de datos del módulo Waspmote [6] 81
B.3. Hoja de datos del módulo 802.15.4/ZigBee [6] 82
B.4. Hoja de datos del módulo Gateway [6] 83

Índice de figuras

Figura 1 Sistema implementado actualmente 14
Figura 2 Diagrama de la solución seleccionada 15
Figura 3 Diagrama de bloques del sistema actual 17
Figura 4 *BeagleBoard* xM [1] 18
Figura 5 POP con el procesador y las memorias flash y RAM 19
Figura 6 Dispositivo Waspmote [5] 20
Figura 7 Conectores de los dispositivos Waspmote [6] 21
Figura 8 Módulo XBee [7] 22
Figura 9 Topología P2P de los módulos XBee [2] 22
Figura 10 Waspmote Gateway [6] 23
Figura 11 Diagrama del Módulo de Envío 24
Figura 12 Operación generalizada del Protocolo Punto a Punto 25
Figura 13 Estructura del frame del Protocolo Punto a Punto [9]: 26
Figura 14 Modelo OSI del Protocolo Punto a Punto 26
Figura 15 Formato de comunicación serial RS-232 [12] 27
Figura 16 Módem 3G/HSDPA 31
Figura 17 Arquitectura Cliente/Servidor del Protocolo SFTP [22] 32
Figura 18 Enlace PPP [23] 33
Figura 19 Protocolo NTP [26] 36
Figura 20 Ventana de configuración de la biblioteca Gammu 38
Figura 21 Diagrama de flujo de la solución Implementada 41
Figura 22 Diagrama de bloques de la solución implementada 45
Figura 23 Acelerómetro [6] 46
Figura 24 Topología estrella implementado con Waspmote [6] 46
Figura 25 1: Módulos USB. 2: Módulo Ethernet. 3: Módulo RS-232. 4: Módulo microSD . 47
Figura 26 Script de configuración de fecha 51
Figura 27 Script Integración 53
Figura 28 Script Autenticación 54
Figura 29 Formato de los archivos .csv 55
Figura 30 Script de Recolección de datos 56
Figura 31 Script carga de archivo por SFTP 58
Figura 32 Script de envío de mensaje SMS de notificación 59
Figura 33 Boot BeagleBoard 60
Figura 34 Consola BeagleBoard 61
Figura 35 Captura de imagen de la conexión establecida a internet con el protocolo PPP 61
Figura 36 Carga exitosa del archivo con 40000 muestras 62
Figura 37 Archivo cargado al servidor eBridge 62
Figura 38 Aplicación móvil para medición de la velocidad de ancho de banda 63
Figura 39 Capturas de pantalla de carga exitosa y error durante la carga 64
Figura 40 Carga exitosa del archivo con advertencia 65

Figura 41 Capturas de pantalla de clave incorrecta, muestras inválidas, reinicio y apagado del sistema 65
Figura 42 Archivos cargados al servidor eBridge 67
Figura 43 Formato de los archivos cargados 68
Figura 44 Sistema diseñado realizando pruebas en el puente 69

Índice de tablas

Tabla 1 Características del procesador OMAP3530 [2] [4] 19
Tabla 2 Especificaciones técnicas del Waspmote [6] 21
Tabla 3 Resumen de Comandos Hayes 29
Tabla 4 Códigos de respuestas del módem 29
Tabla 5 Resumen de comandos para el módem GPRS utilizado [15] [16] [17] 30
Tabla 6 Ancho de banda según Operadora y Plan [39] [40] [41] 63
Tabla 7 Pruebas realizadas y porcentaje de éxito 66
Tabla 8 Estadísticas de la prueba “Inicio de recolección de datos” en el laboratorio 66
Tabla 9 Estadísticas de la prueba “Inicio de recolección de datos” en el puente 70

Resumen

En la actualidad, han surgido problemas en varias infraestructuras de obras públicas en Costa Rica, entre estos inconvenientes se encuentran los daños estructurales en puentes. Se requieren proyectos enfocados en la monitorización de estas estructuras que permitan la lectura de variables físicas por parte de los encargados, para determinar el estado actual del puente. De esta manera, permitir así la resolución de los problemas con suficiente antelación para evitar daños materiales e inclusive pérdidas humanas.

Varias escuelas del Tecnológico de Costa Rica se encuentran desarrollando en conjunto, bajo el nombre de grupo de investigación *eBridge*, el proyecto *Predicción Remota de Fallas Estructurales en Puentes.* El proyecto de *eBridge* consiste en una red de sensores colocados en varios puntos estratégicos de un puente. Consecuentemente, un sistema ubicado a no más de 100 metros del puente, se encuentra recolectando, fusionando y procesando los datos de los sensores, los cuales se envían al servidor de ese grupo. El sistema actual que realiza tales funciones es un PC, el cual desperdicia recursos, el consumo energético es alto y no es práctica la instalación cerca del puente.

Por lo tanto, el objetivo de este documento es detallar cada paso del desarrollo de un sistema empotrado para la recolección, fusión y procesamiento de datos provenientes de sensores inalámbricos colocados en un puente y el envío posterior de éstos a un servidor de internet. El sistema diseñado se le instaló un Sistema Operativo, el cual maneja los diferentes módulos que se le acopla. Además, la conexión a Internet se da por medio de la red inalámbrica de telefonía móvil

Palabras Claves: BeagleBoard, Biblioteca, Datacard, Gateway, Nodo, PPP, sensores, SFTP

Abstract

Nowadays, several problems have arisen in public works infrastructure in Costa Rica; among these problems are structural damages in bridges. Specifics systems are needed to monitor these structures so the administrators can read the physical variables obtained by these systems. Hence, it allows the early resolution of problems avoiding any kind of damage and even loss of humans lifes.

Several *Tecnológicos de Costa Rica's* schools are developing together (the investigation group is named *eBridge*) the project *Remote Prediction for Bridges's Structural Failures,* which is network of sensors placed in several strategic points under a bridge. Consequently, a system placed within 100 meters of the bridge, is collecting, joining and processing the sensors data, which are sent to the *eBridge's* server. Currently, a Personal Computer is performing these functions, which is a waste of resources, has high energy consumption and is not practical to install near the bridge.

Therefore, the purpose of this paper is to detail every step of the development of an embedded system for collection, fusion and processing of data from wireless sensors placed on a bridge and later sending them to an internet server.

Keywords: BeagleBoard, Datacard, Gateway, Library, Node, PPP, sensors, SFTP

Capítulo I. Introducción

Actualmente, han surgido problemas en varias infraestructuras de obras públicas en Costa Rica, entre estos inconvenientes se encuentran los daños estructurales en puentes. Se requieren proyectos enfocados en la monitorización de las estructuras que permitan la lectura de variables físicas por parte de los encargados, para determinar el estado actual del puente. Por lo tanto, permitir así la resolución de los problemas con suficiente antelación para evitar daños materiales e inclusive pérdidas humanas.

Varias escuelas del Tecnológico de Costa Rica se encuentran desarrollando el proyecto denominado *Predicción Remota de Fallas Estructurales en Puentes*, el cual se encuentra bajo la responsabilidad del grupo de investigación *eBridge*. El proyecto de *eBridge* consiste en una red de sensores colocados en puntos estratégicos de un puente. Por ende, se espera realizar un sistema de fusión de datos que permita la recolección de los mismos provenientes de dichos sensores (nodos), procesarlos y posteriormente enviarlos a la red de *eBridge.* El proyecto se basa en el diseño de sistemas empotrados que permitan realizar funciones dedicadas a una tarea específica, consecuentemente, estos sistemas tendrán un alto nivel de portabilidad así como un consumo menor de energía.

1.1 Antecedentes

El proyecto *Predicción Remota de Fallas Estructurales en Puentes* de eBridge de la Escuela de Ingeniería Electrónica del TEC consiste en una red inalámbrica de sensores la cual cada nodo (sensor) envía de forma inalámbrica los datos recolectados a una computadora PC ubicada a una distancia máxima de 100 metros. De esta manera, la computadora fusiona, procesa y envía los datos a un servidor. Sin embargo, al colocar una computadora PC se requiere la construcción de una casetilla cerca del puente para evitar cualquier daño o robo al equipo. Además, el uso de la computadora PC conduce a un consumo mayor de energía y un desperdicio de recursos innecesario. A raíz de esto, se requiere implementar un sistema portátil al sistema actual, de forma que se encargue de realizar las funciones de fusión, procesamiento y envío de datos. En la figura 1 se muestra el sistema que se encuentra funcionando actualmente:

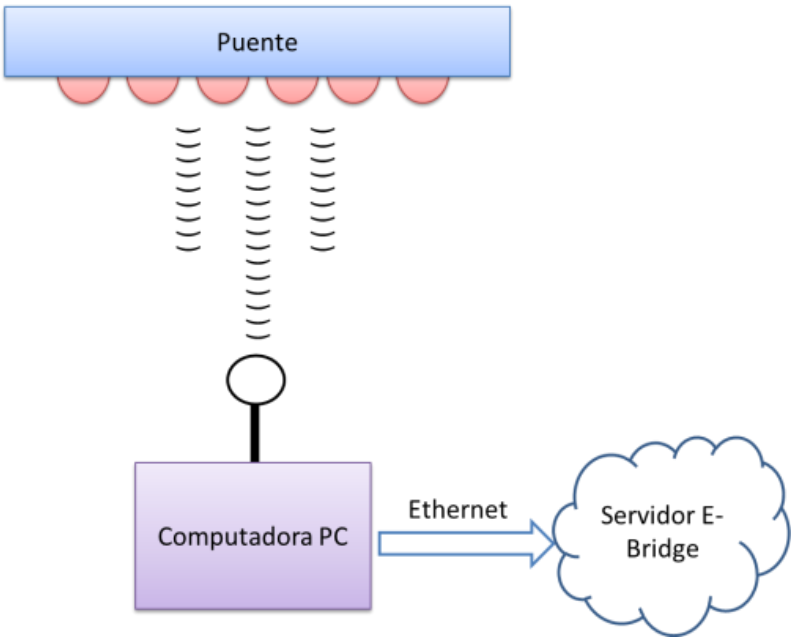


Figura 1 Sistema implementado actualmente

Como se menciona con anterioridad, el sistema actualmente implementado desperdicia recursos además el consumo energético es alto, por lo que se requiere sustituirlo por un sistema portátil, de bajo consumo energético y que realice las mismas funciones del sistema actual.

1.2 Solución seleccionada

La solución tiene un enfoque principal en el área de Sistema Empotrados de la Ingeniería Electrónica. El sistema actual se rediseñará de manera que pueda ser implementado en una tarjeta de desarrollo *BeagleBoard* xM. La misma tarjeta contiene un procesador ARM Cortex – A8, además, se le acopla un módulo para la recepción de datos provenientes de los nodos. Para el envío de datos al servidor, se acopla al sistema un módulo que permite la conexión a internet mediante la red de telefonía celular.

La implementación del sistema en una tarjeta de desarrollo disminuirá el consumo energético y el uso de la red de telefonía celular aumentará el nivel de portabilidad del mismo, de manera que el sistema empotrado tendrá la opción de ser colocado en cualquier sitio ubicado a no más de 100 metros de distancia del puente, por ende, este sistema sustituirá la computadora PC actual. En la siguiente figura se presenta el diagrama de bloques de la solución seleccionada:

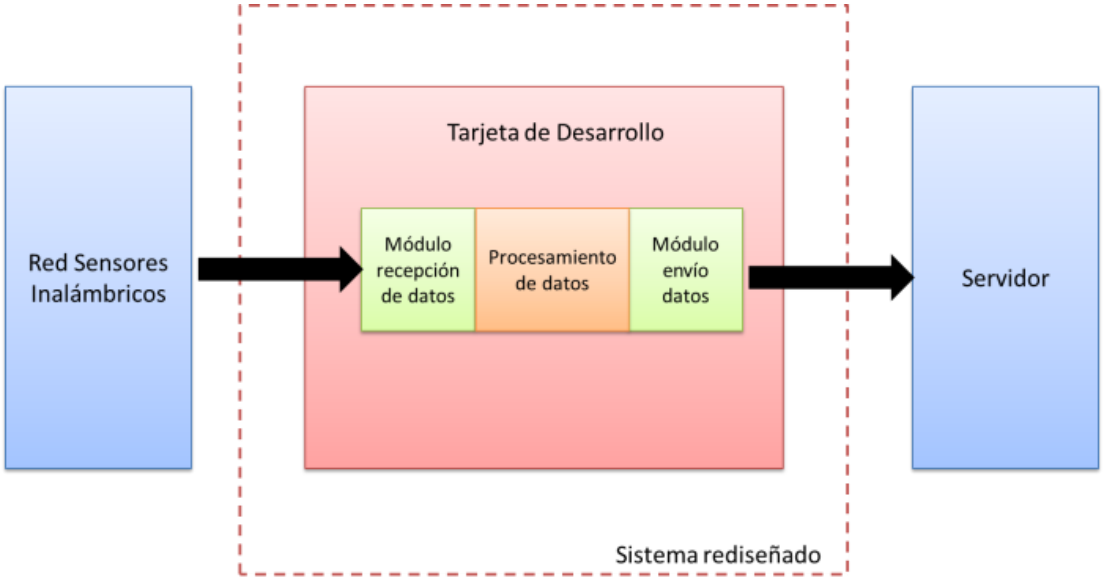


Figura 2 Diagrama de la solución seleccionada

Capítulo II. Meta y Objetivos

2.1 Meta

Rediseñar el sistema electrónico de recolección, procesamiento y envío de datos del proyecto *Predicción Remota de Fallas Estructurales en Puentes* del grupo *eBridge* en un sistema empotrado.

2.2 Objetivo general

Desarrollar un sistema empotrado para la recolección, fusión y procesamiento de datos provenientes de sensores inalámbricos colocados en un puente y el envío posterior de éstos a un servidor usando el protocolo SFTP.

2.3 Objetivos específicos

1. Seleccionar y configurar la distribución GNU/Linux adecuada a las características del hardware provisto, mediante la comparación de al menos dos implementaciones de S.O.
2. Desarrollar una rutina que permita la recolección, procesamiento y envío de los datos provenientes de una red de sensores, usando para ello los protocolos PPP y SFTP sobre una red de telefonía 3G/HSPDPA
3. Desarrollar las rutinas de software que automaticen el proceso de recolección de datos mediante mensajes de textos (SMS).

Capítulo III. Marco teórico

3.1 Sistema actual

El sistema actualmente implementado se muestra en la Figura 1. En la estructura del puente se encuentran diferentes tipos de sensores, los cuales miden la deflexión, la tensión superficial, el desplazamiento lineal y posición actual (x,y,z) del puente. Cada sensor se encuentra colocado estratégicamente en la estructura, de esta manera se forma una red inalámbrica de sensores. Cada sensor se le acopla un Waspmote, estos dispositivos serán referidos como nodos. Cada nodo se conecta de manera inalámbrica por medio del protocolo Zigbee al nodo sumidero.

En el sistema actual, el nodo sumidero se encuentra acoplado a una computadora personal que se ubica cerca del puente, esta computadora se encuentra programada para recolectar los datos de la red de sensores, procesarlos y enviarlos a un servidor específico. En este proyecto, se pretende sustituir todas las funciones que realiza la computadora personal por un sistema portátil de menor consumo de recursos así como con menor consumo de potencia. En la siguiente figura, se observa el diagrama de bloques del sistema actual:

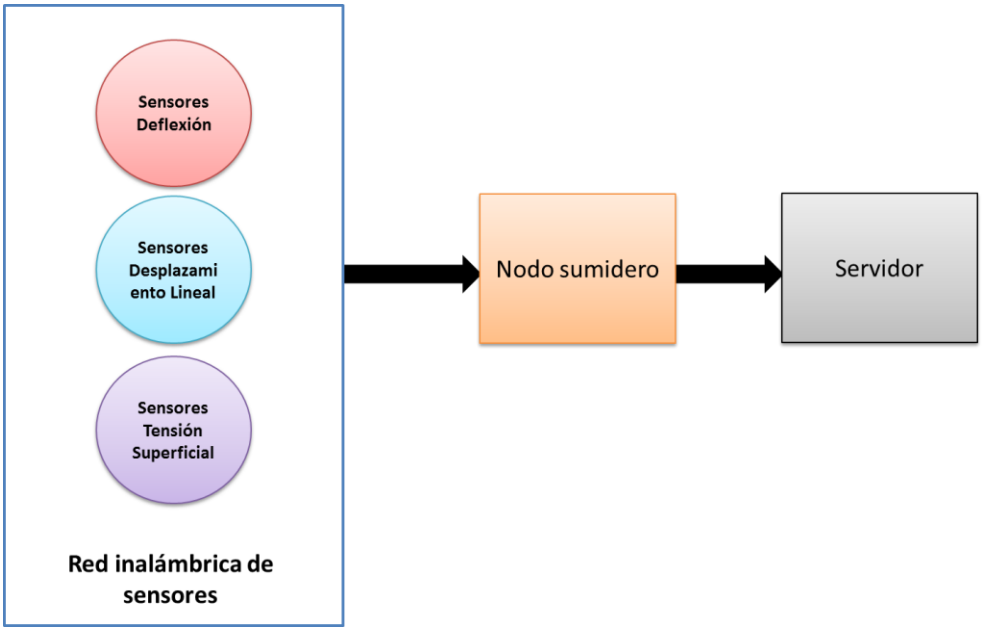


Figura 3 Diagrama de bloques del sistema actual

3.2 Módulos

Como se observa en la Figura 2, el sistema rediseñado posee diversos módulos. En este apartado se referirá con detalle cada uno de estos.

3.2.1 Tarjeta de desarrollo

La tarjeta de desarrollo utilizada es una *BeagleBoard* xM [1]:

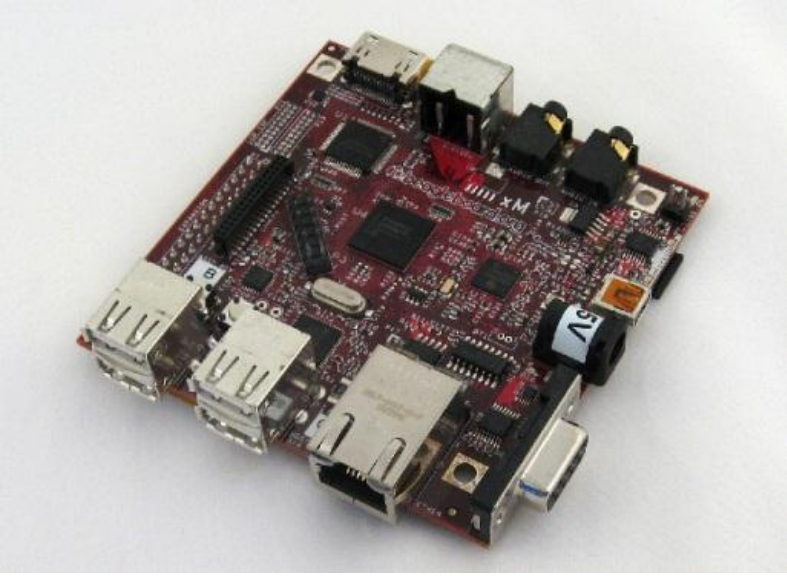


Figura 4 *BeagleBoard* xM [1]

Esta tarjeta es ideal para la implementación del proyecto ya que contiene varios módulos necesarios para el sistema además de tener el microcontrolador acoplado. La siguiente lista detalla algunos de los elementos útiles para el proyecto que se encuentra ensamblados en la tarjeta:

- Procesador OMAP3530 [2]
- Puertos USB
- Puerto Ethernet
- Puerto RS-232
- Conector SD para tarjeta MicroSD

Este procesador se encuentra empaquetado con una técnica llamada POP (*acrónimo en inglés de: Package on Package*), esta consiste en la cual la memoria se encuentra montada sobre el procesador [3]. En la siguiente figura se observa la implementación de esta técnica:

Figura 5 POP con el procesador y las memorias flash y RAM

La siguiente tabla muestra algunas especificaciones del procesador:

Tabla 1 Características del procesador OMAP3530 [2] [4]

OMAP3530	
Sistemas Operativos compatibles	Android,Neutrino,Integrity,Tornado,Windows Embedded CE,Linux,VXWorks
Procesador Digital de Señales (DSP)	C64x
Frecuencia máx DSP	520 MHZ
Unidad central del procesamiento (CPU)	Texas Instruments ARM Cortex A8 1GHz processor
Frecuencia máx CPU	1 GHz
Memoria	Micron 4Gb MDDR SDRAM (512MB) 200MHz
L1 Cache	112 KB (DSP),32 KB (ARM Cortex-A8)
L2 Cache	256 KB (ARM Cortex-A8),96 KB (DSP)
Puertos USB	4. Cada puerto puede proveer hasta 0.5 A
UART (SCI)	3
I2C	3
SPI	4
UART	1 (RS232 DB9 Connector)
Conector SD/MMC	MicroSD
Video	DVI-D ,S-VIDEO (TV-OUT)
Fuente de alimentación	DC, USB

3.2.2 Red inalámbrica de sensores (Waspmote)

La red inalámbrica de sensores consiste en varios módulos colocados en diferentes puntos estratégicos del puente. Cada módulo (nodo) recolecta datos de la deflexión, la tensión superficial, el desplazamiento lineal y posición actual (x,y,z) del punto del puente donde se encuentra colocado el nodo. Los dispositivos utilizados se denominan Waspmote [5], los cuales se basan en una arquitectura modular, la cual consiste en integrar únicamente los módulos que se requieren en cada dispositivo [6]. Esta característica le permite al dispositivo ser bastante versátil ya que se pueden agregar los módulos según las necesidades del proyecto.

Figura 6 Dispositivo Waspmote [5]

Los módulos disponibles para integrar en Waspmote se clasifican en [6]:

- Módulos ZigBee/802.15.4 (2.4GHz, 868MHz, 900MHz). Baja y alta potencia.
- Módulo GSM - 3G/GPRS (Quadband: 850MHz/900MHz/1800MHz/1900MHz)
- Módulo GPS
- Módulos Sensoriales (Placas de Sensores)
- Módulo de almacenamiento: SD Memory Card

En la siguiente figura se observa un dispositivo Waspmote con los diferentes conectores para cada módulo mencionado anteriormente:

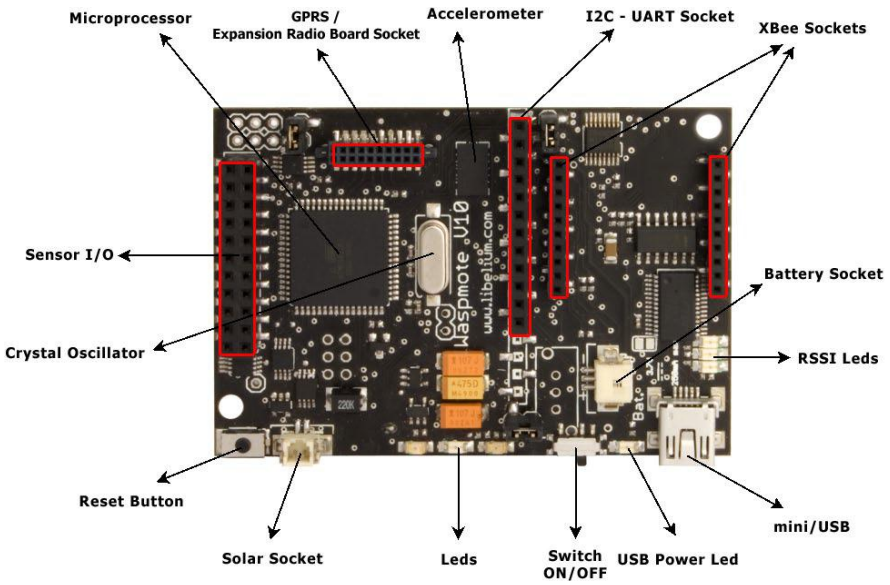


Figura 7 Conectores de los dispositivos Waspmote [6]

Las especificaciones del dispositivo se observa en la siguiente tabla:

Tabla 2 Especificaciones técnicas del Waspmote [6]

Waspmote	
Microcontrolador	ATmega1281
Frecuencia	8 MHz
SRAM	8KB
EEPROM	4KB
FLASH	128KB
SD Card	2GB

Cabe destacar que cada dispositivo cuenta con una batería la cual puede ser recargada mediante conexión USB o un módulo de tensión por placa solar que puede ser adaptado fácilmente.

Además de los módulos adicionales que pueden ser acoplados a estos dispositivos, también poseen un sensor de temperatura y un acelerómetro.

La integración del acelerómetro permite la medición de la aceleración en los 3 ejes (X,Y,Z), estableciendo 2 tipos de eventos: caída libre y detección de cambio de dirección [6].

En este proyecto, se acoplan módulos de XBee a los dispositivos Waspmote. Los módulos XBee se observan en la siguiente figura:

Figura 8 Módulo XBee [7]

Estos módulos establecen comunicación inalámbrica con el módulo de recepción de datos acoplado en la tarjeta de desarrollo. El módulo XBee utiliza el protocolo 802.15.4, la frecuencia utilizada es la banda libre de 2,4GHz, utilizando 16 canales con un ancho de banda de 5MHz por canal. Los módulos XBee 802.15.4 cumplen con el estándar IEEE 802.15.4 [6]. Estos nodos tienen la capacidad de transmitir hasta 100 m en línea vista.

La topología de este tipo de red es el P2P, ya que estos módulos establecen conexiones punto a punto con otros nodos mediante el uso de parámetros como la dirección MAC (*acrónimo en inglés de: Media Access Control*). La dirección MAC es un valor único que identifica un dispositivo específico dentro de la red [8].

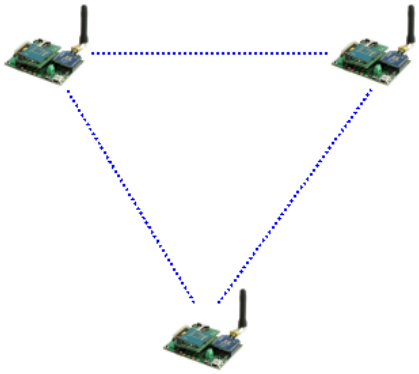


Figura 9 Topología P2P de los módulos XBee [2]

3.2.3 Módulo de recepción

Los nodos mencionados en la sección anterior se comunican con un nodo sumidero o un nodo central. Este nodo sumidero es un módulo de recepción que se conecta mediante USB a la tarjeta de desarrollo. Este dispositivo permite obtener los datos que circulan por la red sensorial, este dispositivo se denomina Waspmote Gateway. Como su nombre lo indica, este módulo actúa como un puente de datos o puerta de acceso entre la red y el equipo receptor. El receptor se encarga de almacenar o utilizar los datos de acuerdo a las necesidades de la aplicación [6]. En la siguiente figura se observa el dispositivo mencionado:

Figura 10 Waspmote Gateway [6]

Como se observa en la figura 10, el *Gateway* posee un conector USB A. El equipo receptor debe ser compatible con la conectividad estándar USB. Al instalarse correctamente el *Gateway*, éste aparece como un nuevo puerto serie de comunicación conectado directamente a la UART del módulo [6]. Para capturar los datos por el puerto serie se puede utilizar cualquier aplicación como *Minicom o Cutecom* en Linux ó *Hyperterminal* en Windows. La velocidad que se debe seleccionar es 38400 ya que es la velocidad estándar establecida para Waspmote.

3.2.4 Módulo de envío

El módulo para la transmisión de los datos procesados al servidor eBridge utiliza tres protocolos de comunicación: PPP *(acrónimo en inglés de: Point-to-Point Protocol),* HSDPA (acrónimo en inglés de: *High-Speed Downlink Packet Access*), y SFTP *(acrónimo en inglés de: SSH File Transfer Protocol ó Secure File Transfer Protocol)*.

El primer protocolo establece comunicación entre la tarjeta de desarrollo y el módem. El módem utilizado en este proyecto es un *Datacard,* el cual es un dispositivo que se conecta a la *BeagleBoard* mediante la conexión USB. El *Datacard* posee una ranura para una tarjeta SIM, por lo que utiliza la red de telefonía móvil para realizar la conexión a Internet. El segundo protocolo establece la comunicación a Internet, y por último, al establecerse la comunicación con la red, el protocolo SFTP comunica y transfiere datos del sistema al servidor eBridge. El siguiente diagrama del módulo de envío:

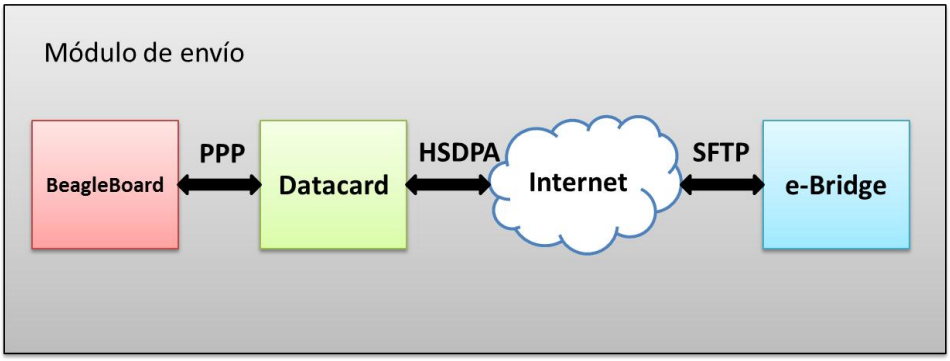


Figura 11 Diagrama del Módulo de Envío

3.2.4.1 Protocolo Punto a Punto

El PPP es un protocolo de capa de enlace de datos el cual encapsula otros protocolos de la capa de red para transmisión en líneas de comunicación sincrónica y asincrónica. La definición de "punto a punto" se debe a una configuración de red entre dos puntos, sin la presencia de una terminal intermedia entre ellos, por lo tanto, este protocolo es una estructura intermediaria que facilita la transmisión entre protocolos de alto nivel (como TCP/IP) y otros enlaces de comunicación [9].

El PPP está diseñado para enlaces simples que transmite paquetes de datos entre dos puntos. Estos enlaces permiten operaciones bidireccionales full-duplex (el envío y recepción de datos es simultáneo) [10].

Este protocolo fue propuesto a inicios de los 90's con un propósito original permitir un estándar para encapsular el protocolo IP para comunicaciones de punto a punto en un entorno mixto de proveedores diferentes y dispositivos. Actualmente, el protocolo tiene un amplio uso en conexiones asíncronas y síncronas entre computadoras, routers y otros dispositivos intermediarios (como el módem HSDPA) [9].

La encapsulación PPP prevé la multiplexación de diferentes protocolos de red de forma simultánea sobre el mismo enlace. El diseño de la encapsulación permite mantener la compatibilidad entre los hardware [10].

Con el fin de ser lo suficientemente versátil dentro de una gran variedad de entornos (distintos proveedores y dispositivos), el protocolo proporciona un Protocolo de Control de Enlace (en inglés: *Link Control Protocol,* acrónimo: *LCP).* El LCP se utiliza de acuerdo al formato de encapsulación, este protocolo maneja paquetes de datos de tamaños predeterminados, también permite la detección de errores durante la transmisión de datos. Además posee facilidades opcionales como la autenticación de la identidad de un punto del enlace y determina cuándo el enlace se encuentra funcionando correctamente o no [10].

Por otro lado, los enlaces punto a punto tienen a muchos problemas con la actual familia de protocolos de red, por ejemplo, la asignación y manejo de direcciones IP. Estos

tipos de problemas son controlados por una familia de Protocolos de Control de Red (en inglés: *Network Control Protocols*, acrónimo: *NCP).* Cada protocolo de control maneja las necesidades específicas requeridas por sus respectivos protocolos de red [10].

Desde un punto de vista general, al establecer una comunicación a través de un enlace punto a punto, primero el dispositivo maestro envía un paquete de datos, denominado *frame,* de LCP para configurar y probar el enlace de datos. Una vez establecido el enlace de datos así como otros datos requerido para el LCP, el dispositivo maestro envía un *frame* de NCP para escoger y configurar uno o más protocolos de capa de red. Cuando cada capa ha sido configurada, los paquetes de cada capa ya están listos para utilizar el enlace principal. El enlace quedará configurado hasta que algún frame de LCP o NCP indique que cierre el enlace, o algún evento externo ocurra como la inactividad por un periodo de tiempo o la desconexión por parte del usuario [11]. En la siguiente figura se muestra un diagrama de la operación generalizada del protocolo:

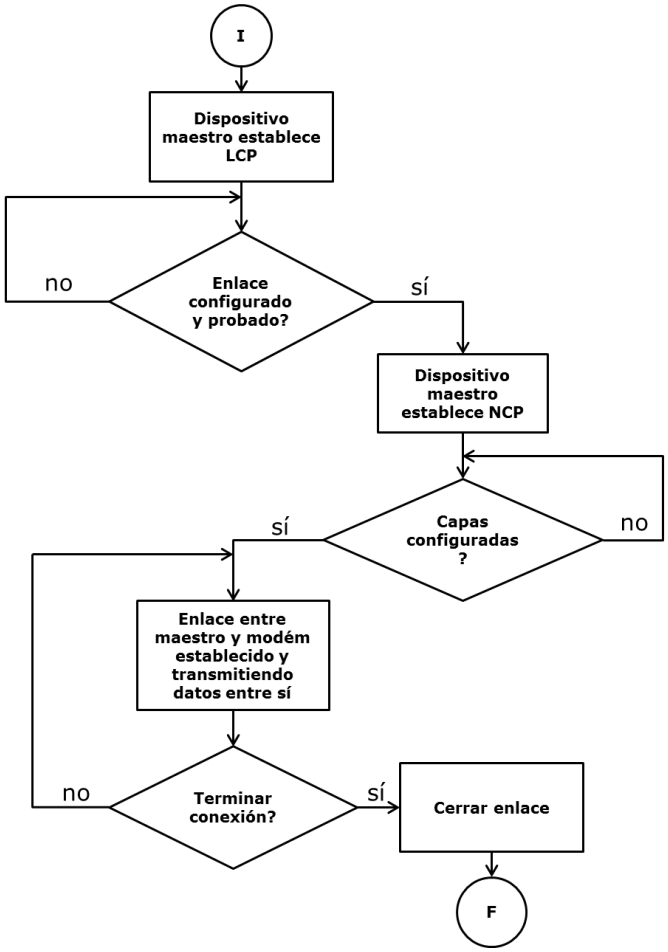


Figura 12 Operación generalizada del Protocolo Punto a Punto

La estructura del frame del protocolo se muestra en la siguiente figura:

flag	address	control	protocol	information	fcs (crc)	flag
1 byte	1 byte	1 byte	2 bytes	up to 1500 bytes	2 bytes	1 byte

Figura 13 Estructura del frame del Protocolo Punto a Punto [9]:

La Organización Internacional de Estándares (ISO) establece el Modelo de Interconexión de Sistemas Abiertos (OSI) para dividir el protocolo en una estructura que consiste en siete capas. Cada capa determina un set de funciones comunes en el entorno de red para la transferencia de datos de un punto a otro [12]. En la siguiente figura se observa la división de las capas:

Figura 14 Modelo OSI del Protocolo Punto a Punto

En la primera capa de abajo hacia arriba (Ver figura anterior) corresponde a la capa física la cual es utilizada para el transporte de datos. ~~La~~ segunda capa, se encarga de la transferencia de datos de una forma fiable a través del medio físico. La capa de red maneja las direcciones de las fuentes destinos de los datos a través de las redes intercomunicadas. En la capa de transporte se asegura tanto los datos recibidos como los transmitidos sean idénticos, esta etapa se responsabiliza de la detección y corrección de errores, posteriormente en la retransmisión de datos. En la quinta capa, se gestiona o distingue una o más conexiones entre aplicaciones. En la siguiente capa se da formato a los datos a una forma estandarizada para el uso de las aplicaciones. Por último, la capa de aplicación consiste en programas que interactúa con el usuario [12].

La configuración del PPP para varios propósitos, requiere de conocimientos de interfaces y conexiones seriales, tecnologías de módems así como diferentes protocolos de red existentes (direccionamientos y enrutamientos). En el entorno del PPP, la conexión más común en la interfaz RS-232 que conecta la computadora a una línea telefónica normal. En la siguiente figura se observa el formato de transmisión de datos de un enlace serial RS-232:

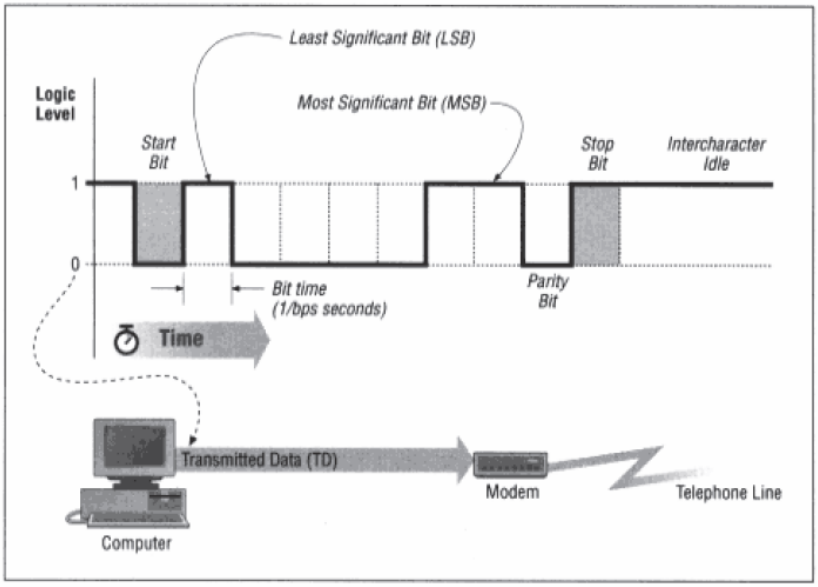


Figura 15 Formato de comunicación serial RS-232 [12]

De acuerdo a la figura anterior, el formato tiene las siguientes características:

- Bit de inicio: bandera que indica al receptor la llegada de un carácter.
- Velocidad: determina la velocidad del envío de datos a través de la interfaz serial. Los valores más comunes son (en bps) 9600, 19200, 38400, 57600, 115200. Para el PPP, se prefieren tasas de envío más altas ya que permiten mayor envío de datos por unidad de tiempo, sin embargo, el usuario debe especificar la tasa más alta según las limitaciones del hardware.
- Bits de datos: para el PPP se utiliza 8 bits.
- Bit de paridad: Para la detección de errores por parte del receptor, aunque el mismo protocolo contiene su propio método de detección, por lo tanto, no se utiliza.
- Bit de parada: indica el fin de un caracter y proporciona un separador antes de un bit de inicio del siguiente caracter.

Esta interfaz incluye diferentes señales que son usadas para conocer el estado de la conexión y el control del enlace [12]:

- Data Terminal Ready (DTR): Cuando esta señal se encuentra activada, la computadora le notifica al módem que la conexión telefónica si existe. En caso de que no se encuentre establecida ninguna conexión, el módem acepta o inicia una nueva conexión telefónica. Si la señal se encuentra desactivada, se desconecta la conexión telefónica, y no acepta comandos de control ni tampoco llamadas telefónicas.

- Data Set Ready (DSR): Cuando se encuentra activado, indica a la computadora que está listo para operación.
- Carrier Detect (CD): Conocido también como detector de señal de la línea de recepción. Cuando la seña se encuentra habilitada indica que un módem remoto se encuentra activado y puede recibir o enviar datos. Cuando la señal se encuentra desactivada indica que se terminó la conexión serial.
- Ring Indicator (RI): Se activa cuando existe una llamada entrante.

Una vez establecido el enlace con la computadora, el protocolo PPP se comunica con el módem mediante comandos especiales llamados *Comandos Hayes.*

La función principal del módem es la conversión de señales digitales a una forma análoga para la transmisión de la misma a través de una línea telefónica. Esta conversión requiere modulación. Una señal digital crea las fluctuaciones en la señal enviada a otro módem. Estas fluctuaciones contiene la información. El módem receptor debe llevar a cabo un proceso de demodulación para recobrar la información original [12]. Se inicia la configuración del módem con la velocidad de envío de datos, como se mencionó anteriormente los más comunes son (en bps) 9600, 19200, 38400, 57600, 115200. Posteriormente, se requiere enviar comandos a través de la interfaz serial, cabe destacar que los módems reconocen comandos solo cuando se encuentra en modo desconectado o en modo comando [12].

Para comunicación de un módem con una computadora Linux utiliza aplicaciones terminales como *Minicom o Cutecom*. Sin embargo, para el proyecto actual no se requiere de esas aplicaciones, ya que el módem establece comunicación con la computadora a través del protocolo PPP.

Los comandos utilizados se denominan Comandos Hayes AT, en la siguiente tabla se muestra los comandos comúnmente utilizados [13] [14] [15]:

Tabla 3 Resumen de Comandos Hayes

Comando	Descripción
ATZ	Resetea el módem a la configuración de fábrica
AT&F	Resetea el módem a la configuración por defecto
AT	Obtiene la atención del módem
ATQ	Habilita el modo de transmisión de códigos de resultados del módem. 0: Habilitado. 1: Deshabilitado
ATS0	Habilita el modo de contestado automático del módem. 0: Deshabilitado. 1-255: Habilitado
+++	Cambio de modo datos a modo comando
AT 0	Cambio de modo comando a modo datos
AT DT número	Marcar usando tonos el número especificado y conectar
AT DP número	Marcar usando pulsos el número especificado y conectar
ATD	Llama a un número para inicializar la conexión a internet (Ej: *99#)
AT H	Colgar
A/	Ejecutar el último comando insertado
A	Contestar
AT+CGATT	Si el resultado devuelto es 1 indica que el dispositivo se encuentra acoplado
AT+FCLASS	Indica el modo de operación del módem. 0: Data, 2: Fax

Por otro lado, los posibles códigos de respuestas del módem se presentan en la siguiente tabla [15]:

Tabla 4 Códigos de respuestas del módem

Respuesta	Código	Descripción
OK	0	Comando ejecutado sin error
CONNECT	1	Conexión establecida
RING	2	Tono de llamada está presente
NO CARRIER	3	Llamada fallida al conectarse o desconectarse
ERROR	4	Comando inválida o commando muy largo
BUSY	7	El modem actualmente está realizando otra llamada
NO ANSWER	8	Tiempo de espera agotado al ejecutar una llamada

En el proyecto actual se utilizó un módem HSDPA. El dispositivo contiene una ranura para insertar una tarjeta SIM. Los comandos específicos para estos módems se muestran en la siguiente tabla:

Tabla 5 Resumen de comandos para el módem GPRS utilizado [15] [16] [17]

Tipo	Comando	Descripción
Comandos generales	AT+CGMI	Identificación del fabricante
	AT+CGSN	Obtener número de serie
	AT+CIMI	Obtener el IMSI.
	AT+CPAS	Leer estado del modem
Comandos del servicio de red	AT+CSQ	Obtener calidad de la señal
	AT+COPS	Selección de un operador
	AT+CREG	Registrarse en una red
	AT+WOPN	Leer nombre del operador
	AT+CGDCONT	Especifica el protocolo del paquete de datos, el APN (Access Point Name)
Comandos de seguridad	AT+CPIN	Introducir el PIN
	AT+CPINC	Obtener el número de reintentos que quedan
	AT+CPWD	Cambiar password
Comandos para la agenda de teléfonos	AT+CPBR	Leer todas las entradas
	AT+CPBF	Encontrar una entrada
	AT+CPBW	Almacenar una entrada|
	AT+CPBS	Buscar una entrada
Comandos para SMS	AT+CPMS	Seleccionar lugar de almacenamiento de los SMS
	AT+CMGF	Seleccionar formato de los mensajes SMS
	AT+CMGR	Leer un mensaje SMS almacenado
	AT+CMGL	Listar los mensajes almacenados
	AT+CMGS	Enviar mensaje SMS
	AT+CMGW	Almacenar mensaje en memoria
	AT+CMSS	Enviar mensaje almacenado
	AT+CSCA	Establecer el Centro de mensajes a usar
	AT+ WMSC	Modificar el estado de un mensaje

3.2.4.2 Protocolo HSDPA

Los módems HSDPA son dispositivos que se conectan por puerto USB de una computadora para proporcionarle acceso a Internet. En la siguiente figura se observa un ejemplo del dispositivo mencionado:

Figura 16 Módem 3G/HSDPA

Estos dispositivos utilizan el protocolo PPP. Este protocolo es utilizado para establecer un enlace a nivel de la capa TCP/IP entre el dispositivo y una computadora y asigna dinámicamente una dirección IP al módem [11].

Como se mencionó anteriormente, la conexión a Internet se da mediante el protocolo de comunicación de telefonía móvil HSDPA, también conocido como red 3.5G. Esta tecnología está basada en el esquema de codificación WCDMA (*acrónimo en inglés de: Wideband code division multiple Access*.). Este protocolo es una técnica mejorada del UMTS (*acrónimo en inglés de: Universal Mobile Telecommunications System.* También conocido como red 3G) [18]. Actualmente, este protocolo permite tasas de transmisiones de 4 a 5 veces más rápidas que la generación UMTS [19]. La transmisión de datos por HSDPA puede llegar hasta 14 Mbps [20]. Sin embargo, el desempeño real de este protocolo es de 2-3 Mbps debido a las pérdidas durante la transmisión [21].

3.2.4.3 Protocolo SFTP

La transferencia de un archivo a un servidor específico se da mediante el protocolo SFTP (*acrónimo en inglés de: SSH File Transfer Protocol ó Secure File Transfer Protocol*).

Este protocolo es una herramienta poderosa y popular con un enfoque de seguridad en la transferencia de archivos dentro de una red, siempre que se envíe datos a una red específica, este protocolo encripta dichos datos de manera automática y cuando los datos llega al receptor, el mismo protocolo los desencripta [22].

La arquitectura de este protocolo es de servidor/cliente. En el servidor se ejecuta una aplicación de servidor de SSH que acepta o rechaza comunicaciones entrantes.

Una aplicación SSH del servidor es manejado por un sistema administrador, este sistema acepta o rechaza comunicaciones entrantes. Por otro lado, si otros sistemas desean comunicarse con el servidor (dichos sistemas son conocidos como clientes), éstos deben autenticarse ante el servidor, si dicha autenticación es correcta, el servidor establece comunicación con el cliente. El cliente, por lo general, realiza peticiones como envío de archivo, descarga de un archivo o ejecutar comandos específicos dentro del servidor desde un sistema cliente [22]. Todas las comunicaciones entre servidor-clientes se encuentran encriptadas y protegidas de modificaciones durante la transferencia. Descripción de los principales principios físicos y/o electrónicos relacionados con la solución del problema. En el siguiente diagrama se muestra la arquitectura de cliente/servidor del protocolo SFTP:

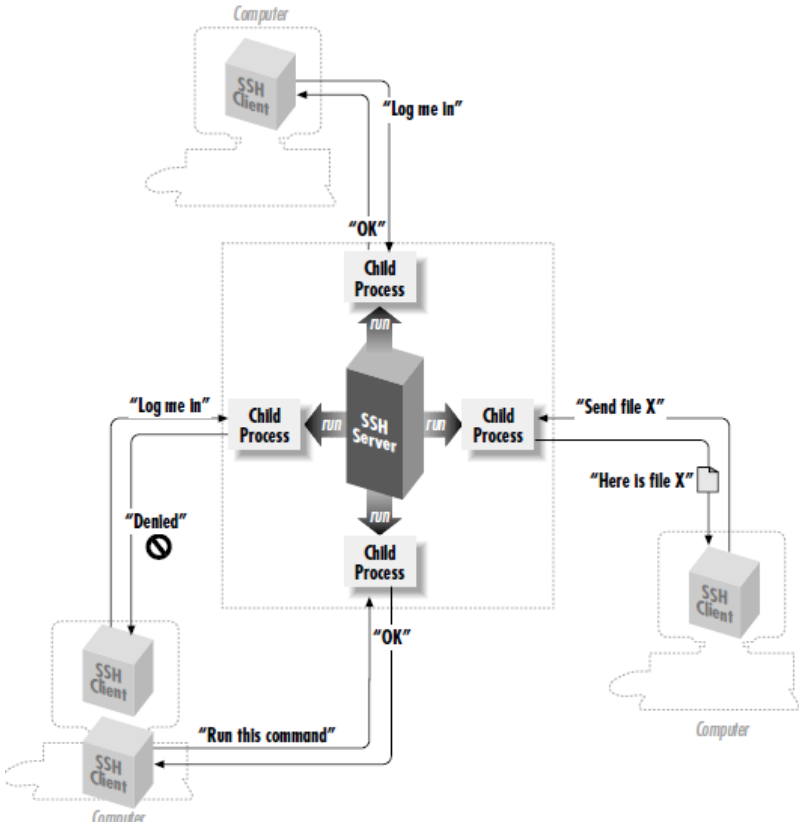


Figura 17 Arquitectura Cliente/Servidor del Protocolo SFTP [22]

3.3 Bibliotecas implementadas en el S.O Ubuntu

3.3.1. Biblioteca PPP

Por lo general, el protocolo PPP es utilizado para conectar una máquina sin una conexión IP directa con una segunda máquina (servidor PPP) que sí posee una. Cuando la primera máquina se registra y se autentica ante un servidor PPP, el servidor le provee una dirección IP. Después de establecer la comunicación entre máquina y el servidor PPP, la misma biblioteca realiza la conexión a internet [23].

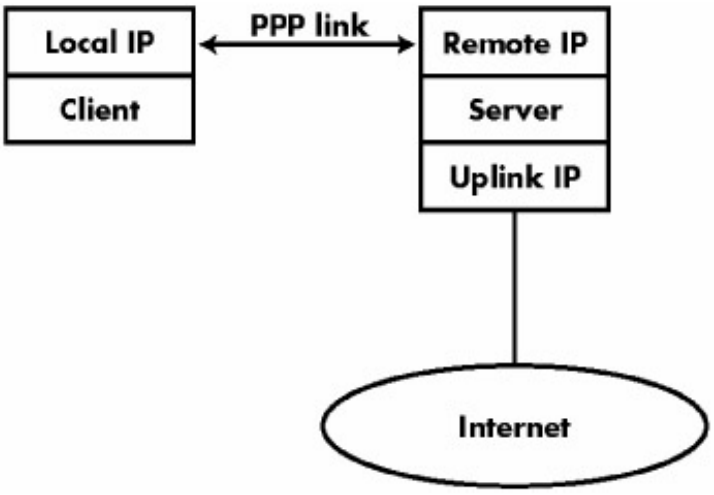


Figura 18 Enlace PPP [23]

La implementación del protocolo PPP en Linux se da mediante la biblioteca *pppd* [24], la cual junto con el controlador PPP del kernel de Linux, establece y mantiene un enlace PPP entre la computadora con otro sistema (en el caso de este proyecto: un módem HSDPA). Según el manual oficial de esta biblioteca, el usuario debe crear dos archivos: un script (archivo que contiene órdenes de ejecución en texto plano que son interpretados por la biblioteca pppd) con opciones de configuración del protocolo y otro script de comunicación con el módem.

El script de configuración de protocolo se encuentra en el directorio /etc/ppp/peers/*nombre_script.* Las opciones utilizadas frecuentemente son [24]:

- *ttyname:* Utiliza el puerto serial *ttyname* para establecer comunicación con el módem. El directorio común es /dev/*ttyname*
- *speed*: Establece el *baudrate* para el dispositivo serial. Ej: 115200, 38400.
- *noauth:* No requiere autenticación de la máquina antes del envío o recepción de paquetes.

- *connect script: Utiliza el programa chat para comunicar con el módem, se ejecuta el programa y se escribe el directorio del script de comunicación con el módem. Ej: connect "/usr/sbin/chat -v -f /etc/ppp/nombreCHAT"*
- *crtscts: Especifica al pppd debe utilizar el control del flujo del hardware del Puerto serial (usando las señales RTS y CRS del RS-232).*
- *defaultroute: Agrega una ruta por defecto a las tablas de ruteo del sistema. De esta manera, la conexión por defecto del sistema será el protocolo PPP.*
- *lock : Bloquea el módem para asegurar que sólo el pppd lo pueda utilizar.*
- *noipdefault: Deshabilita el comportamiento por defecto cuando no existe una dirección IP local.*
- *modem –detach: Deshabilita las líneas de control del módem, de esta manera, se puede ejecutar el chat.*
- *usepeerdns: Solicita dos direcciones DNS (acrónimo en inglés de: Domain Name System)*

A continuación, se muestra un ejemplo de un archivo de configuración obtenido de [25]:

```
/dev/ttyS1        - Directorio donde se encuentra acoplado el dispositivo
115200             - Baud rate
modem -detach
crtscts
asyncmap 0
defaultroute
noipdefault
lock
connect "/usr/sbin/chat -v -t 240 -f /etc/ppp/chat-isp"
```

El script de comunicación con el módem se coloca en el directorio */etc/ppp/nombreCHAT.* Los comandos de este archivo se observan en la Tabla 3 y Tabla 5. El siguiente se muestra un ejemplo de un script [24]:

```
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
ABORT "BUSY"
ABORT "Username/Password Incorrect"
"" "at"
OK "at&d0&c1"
OK "atdt2468135"
"name:" "^Umyuserid"
"word:" "\qmypassword"
"ispts" "\q^Uppp"
```

3.3.2. Biblioteca NTP

La sincronización de la fecha del sistema se da mediante el protocolo NTP (*acrónimo en ingles de: Network Time Protocol*). El protocolo NTP se organiza en un modelo de estructura jerárquica cliente-servidor. En la parte superior de esta jerarquía hay una pequeña cantidad de máquinas que son conocidos como relojes de referencia. Por lo general, estos relojes son relojes de celsio o GPS (*acrónimo en ingles de*: *Global Positioning System)* que recibe el tiempo desde satélites espaciales. A estas pequeñas máquinas se le acoplan servidores que son los que se utilizan para sincronizar la hora y fecha de las máquinas de los clientes. En la Figura 19 se observa un diagrama del protocolo.

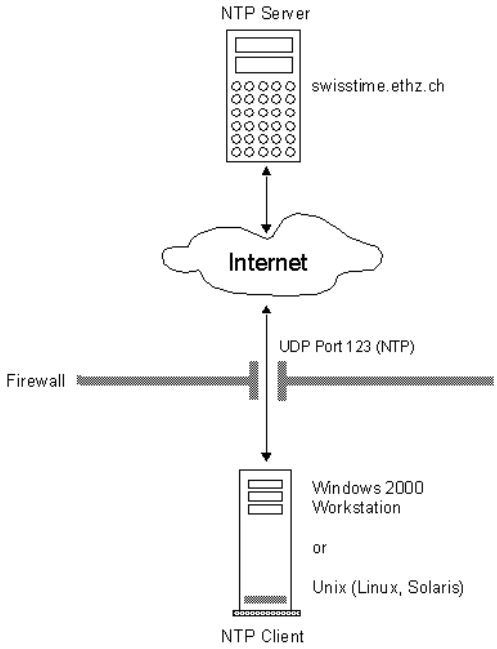


Figura 19 Protocolo NTP [26]

La implementación del protocolo en Ubuntu se da mediante el comando *ntpdate.* Según la documentación oficial de la plataforma Ubuntu, este S.O contiene por defecto esta biblioteca. Cada vez que existe un reseteo de sistema, el reloj puede desconfigurarse con respecto al servidor, por lo tanto, es recomendable sincronizar el sistema al menos una vez al día [27]. El comando de sincronización de la hora y fecha del sistema es:

ntpdate servidor

Los servidores disponibles se encuentran la página http://www.pool.ntp.org/es/. Sin embargo, en este proyecto se seleccionó un servidor local:

ntpdate cr.pool.ntp.org

3.3.3. Bibliotecas de Python

Además de las bibliotecas básicas, las cuales por defecto vienen en Python, se debe instalar tres bibliotecas adicionales: *serial, gammu y paramiko.*

3.3.3.1. Python-serial

Este módulo permite el acceso hacia el puerto serial. Ejemplo [28]:

```
>>> ser = serial.Serial('/dev/ttyS1', 19200)
>>> x = ser.read()          # read one byte
>>> line = ser.readline()   # read a '\n' terminated line
>>> ser.close()
```

El código anterior, lee el puerto ttyS1 (a una velocidad de 19200) y guarda los datos en un *char* o en un *string*.

3.3.3.2. Python-gammu

Este modulo permite accede fácilmente al *Datacard*. De esta manera, se puede utilizar el dispositivo para el envío y recepción de mensajes. Esta biblioteca requiere un archivo de configuración generalmente almacenada en el directorio ~/.gammurc (ver sección 3.3.4 para archivo de configuración). El siguiente código se utiliza para inicializar el módulo [29]:

```
import gammu
import sys

# Create state machine object
sm = gammu.StateMachine()

# Read ~/.gammurc
sm.ReadConfig()

# Connect to phone
sm.Init()
```

3.3.3.3. Python-paramiko

El módulo *paramiko* de *Python* implementa tanto el protocolo SSH como el protocolo SFTP. El siguiente código conecta la máquina con un servidor mediante el protocolo SFTP y carga un archivo al servidor [30]:

```
import paramiko
ssh = paramiko.SSHClient()
ssh.connect('servidor', username='nombreusuario', password='*****')

ftp = ssh.open_sftp()
ftp.put(remotepath, localpath)
ftp.close()
```

3.3.4. Biblioteca Gammu

La biblioteca Gammu es un proyecto que permite controlar un teléfono celular o un *Datacard* desde la línea de comando de Linux, la biblioteca se encuentra escrita en el lenguaje de programación C. Esta biblioteca permite monitorear el buzón de entrada de mensajes SMS, envío y recepción de mensajes SMS o MMS, exportar e importar listas de contactos telefónicos, entre otros [31].

Antes de utilizar esta biblioteca y su correspondiente en *Python* (ver sección 3.3.3.2), se debe ejecutar la orden gammu-config en la línea de comandos de Linux, consecuentemente, se despliega la siguiente ventana de configuración:

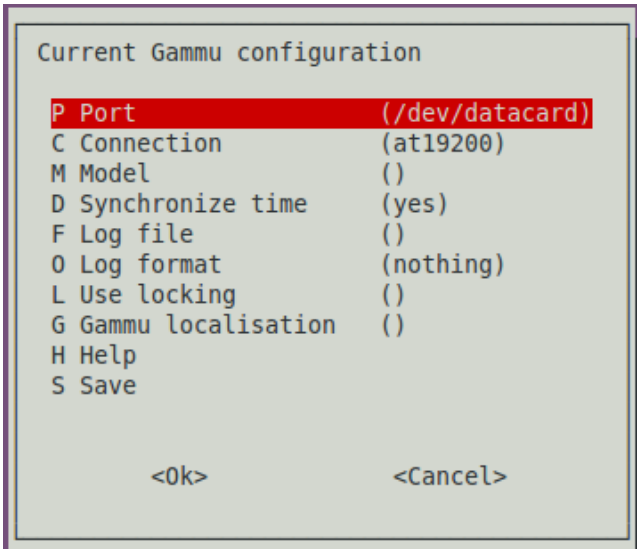


Figura 20 Ventana de configuración de la biblioteca Gammu

Esta pantalla permite crear un archivo de configuración en el directorio ~/.gammurc, el cual permite la biblioteca tanto en Python como en Ubuntu de utilizar el dispositivo (*teléfono celular o Datacard*) para enviar o recibir mensajes SMS [32].

Capítulo IV. Procedimiento Metodológico

4.1 Reconocimiento y definición del problema

El desarrollo del proyecto *Predicción Remota de Fallas Estructurales en Puentes* surge de la necesidad latente del país de mejorar la infraestructura de las obras públicas. Algunos de los inconvenientes presentados dentro de las obras son los daños estructurales en los puentes, los cuales no tienen el mantenimiento adecuado. La falta de atención de estas estructuras ha provocado múltiples accidentes en varios sitios del país, por lo que, surge la necesidad de desarrollar proyectos enfocados en la monitorización de los puentes que permitan la lectura de variables físicas trascendentales para determinar el estado actual de la estructura, consecuentemente, esto permite resolver algún problema con suficiente antelación y así anticipar cualquier accidente o incidente de seguridad evitando daños materiales e inclusive pérdidas de vidas humanas.

El monitoreo de la estructura se da mediante unos sensores colocados en puntos estratégicos definidos por la coordinación del proyecto, sin embargo, no existe un sistema portátil que permita la recolección, procesamiento y envío de datos hacia algún servidor de Internet que permita a algún encargado de interpretar los resultados recolectados por los sensores. Por lo tanto, se propuso como meta diseñar un producto final con las características de ser portátil, de bajo costo económico y de bajo consumo energético que permita realizar esas tareas mencionadas,

4.2 Obtención y análisis de información

Se realizó una consulta preliminar a los encargados de *eBridge* acerca del estado actual del proyecto *Predicción Remota de Fallas Estructurales en Puentes*. Los encargados indicaron que existían sensores ya caracterizados así como una simulación en una computadora PC de la recolección, fusión, procesamiento y envío de datos. A partir de esta información, se utilizó la investigación en internet para encontrar una plataforma de desarrollo viable para la realización de este proyecto. Además, se consideró durante el análisis de la selección de la plataforma, el costo económico de cualquier solución, ya que debería estar acorde a la situación económica actual del país. El último factor por considerar fue encontrar una plataforma que permita configurarse para ser autónoma (no requiere mantenimiento constante) ya que se sitúa cerca del puente, por lo que debe ser capaz de soportar condiciones al aire libre como la oxidación o cualquier daños ocasionados por cambios de temperatura, humedad, calor, entre otros.

4.3 Evaluación de las alternativas y síntesis de una solución

Antes del planteamiento de las alternativas para la solución, se realizó consultas con el grupo *eBridge* acerca de los equipos disponibles para la ejecución de este proyecto. El grupo tenía a su disposición módulos acoplados a los sensores que ya enviaban los datos obtenidos.

A partir de esta información preliminar, se prosiguió plantear algunas soluciones. Independientemente de la plataforma que se escogería, las soluciones tenían que tener las siguientes características:

- Recolección de los datos de los módulos acoplados a los sensores
- Fusión y procesamiento de los datos
- Envío de los datos al servidor de eBridge mediante la red telefónica algún operador
- Automatizar todos los procesos anteriores.

Seguidamente, con la investigación en Internet así como con recomendaciones del asesor, se escogió la plataforma de desarrollo *BeagleBoard*, ya que contiene los módulos necesarios para la implementación del proyecto. Los factores que se consideraron fueron la capacidad del procesador de la plataforma (ARM Cortex A8 1GHz), bajo costo económico, bajo consumo energético y el tamaño de ésta. Cabe destacar, que esta plataforma es muy versátil, permitiendo realizar cambios o mejoras de manera sencilla.

4.4 Implementación de la solución

Como se mencionó en el apartado anterior, el sistema debe recolectar, fusionar, procesar y enviar los datos a un servidor, además, automatizar todos estos procesos.

Para el manejo de los módulos, se implementó un sistema operativo en la plataforma de desarrollo *BeagleBoard*. El sistema operativo (S.O) facilita el manejo de las distintas tareas que debe realizar el sistema. El S.O implementado es la distribución (distro) Ubuntu de Linux. Este distro posee los módulos básicos implementados: USB, Ethernet y UART (este último módulo se utiliza para monitorizar las pruebas), además, facilita la instalación de bibliotecas necesarias para el proyecto.

Después de la instalación del S.O, se procedió a implementar el módulo de envío, el cual es un Datacard conectado vía USB a la *BeagleBoard*. Este dispositivo permite la conexión del sistema a Internet mediante la red de telefonía. En la implementación de este módulo se escribió rutinas en lenguaje *bash* para el manejo de este dispositivo.

Luego de establecer la conexión a internet, se escribió una rutina en *bash* que permite la conexión al servidor de eBridge mediante el protocolo de transferencia de archivo SFTP.

A continuación, se procedió con la implementación del módulo de recepción. La información recolectada por los sensores es recibida por un *Gateway* conectado mediante USB a la plataforma. Este dispositivo recolecta los datos en formato RS-232.

Seguidamente, se escribieron rutinas en lenguaje *Python* para la recolección y fusión de los datos recogidos. La escritura de las rutinas permite guardar y ordenar los datos en un archivo temporal con extensión .csv.

Posteriormente, se le agregó una mejora al proyecto. Aprovechando la funcionalidad del *Datacard*, se instalaron bibliotecas en Ubuntu para el envío y recepción de mensajes SMS. De esta manera, el usuario puede iniciar la recolección de datos mediante un mensaje de texto que contiene una clave y la cantidad de muestras que desea recolectar y el sistema envía mensajes de confirmación al usuario. Estas rutinas se escribieron en el lenguaje *bash*.

Por último, se escribió un script en lenguaje *bash* que integraba todas las funciones anteriores, cabe destacar que este script integraba rutinas escritas en *bash* y *Python*. En el siguiente diagrama muestra la integración de las rutinas:

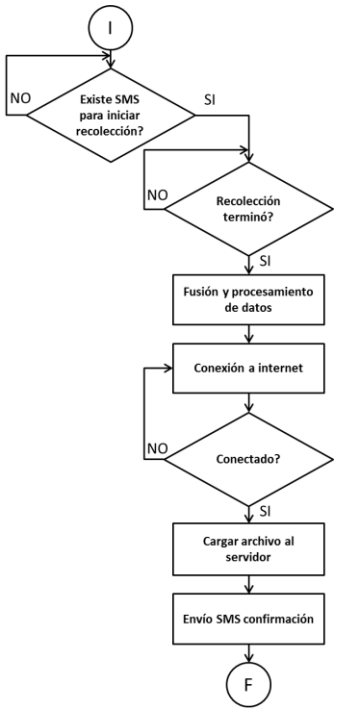


Figura 21 Diagrama de flujo de la solución Implementada

4.5 Reevaluación y rediseño

Para mejorar el sistema desarrollado, se rediseñó parte de las rutinas escritas, se propuso migrar algunas de las rutinas en lenguaje de programación *bash* al lenguaje de programación *Python*, ya que *Python* es bastante versátil y fácil de implementar, además, existe mayor compatibilidad entre las diferentes rutinas. Las rutinas migradas a *Python* fueron: la conexión al servidor de eBridge, el envío de SMS de confirmación y la recepción de mensajes SMS.

El diseño implementado es bastante versátil, ya que en un futuro si se requiere la implementación de más nodos, sólo se necesita modificar un bloque de código en la rutina recolección de datos agregando otro nodo más. Además, el código de recepción y envío de SMS se puede modificar fácilmente de manera que el usuario no sólo indique cuándo empezar la recolección de muestras sino realizar otras aplicaciones más.

Por otro lado, la implementación de Ubuntu permite diseñar un amplio rango de aplicaciones útiles para el monitoreo estructural, unas de las aplicaciones propuestas es monitorear en tiempo real en el sitio los datos recogidos y observar el comportamiento de éstos en una gráfica.

Capítulo V. Descripción detallada de la solución

La implementación de la solución propuesta se explicará detalladamente en este capítulo. Primeramente, se realiza un análisis de las soluciones propuestas y posteriormente, los factores que llevaron a la selección de la solución final.

Seguidamente, se detalla el funcionamiento y el diseño de la estructura de cada uno de los módulos físicos que conforman la solución final. Finalmente, se realiza una descripción detallada de las diferentes rutinas de software desarrolladas para lograr el funcionamiento adecuado de cada módulo físico descrito. Además, en cada sección se presenta diagramas de bloques o diagramas de flujos para entender mejor el concepto de cada etapa de la solución.

5.1 Análisis de soluciones y selección final

En este apartado se detalla las soluciones planteadas para implementar el proyecto. Además, se presenta los factores considerados para llevar a cabo la elección de la solución final.

5.1.1 Plataforma de desarrollo

El criterio principal para la selección de la plataforma es que contengan los siguientes módulos:

- Puertos USB : Acople del Gateway y Datacard
- Puerto Ethernet: Actualización de las bibliotecas del S.O
- Puerto RS-232: Puerto de monitoreo para las pruebas de la tarjeta
- Conector SD para tarjeta MicroSD: Almacena el S.O

Existen diversas plataformas que cumplen con estos requisitos, como las tarjetas de desarrollo de la empresa Microchip o las plataformas de desarrollo de hardware libre Arduino, sin embargo, se seleccionó la tarjeta de desarrollo *BeagleBoard* xM ya que contiene un procesador bastante potente (ARM Cortex A8 1GHz) que permite ejecutar un sistema operativo de manera óptima. La selección de la *BeagleBoard* xM facilita el proceso de programación de las rutinas de software ya que las ejecuta de forma más rápida por el potente procesador de 1 GHz.

5.1.2 Sistema Operativo

Se implementó un sistema operativo en la tarjeta de desarrollo *BeagleBoard* para facilitar el manejo de los diferentes módulos que se acoplan a la tarjeta.

Los sistemas operativos considerados fueron: Android, Angstrom y Ubuntu. Todos los sistemas operativos considerados son *distros* de Linux, ya que uno de los criterios basados en la selección de S.O es que sea software libre.

Según investigaciones en internet, la plataforma más desarrollada para la *BeagleBoard* es el Angstrom, sin embargo, esta plataforma tenía problemas de compatibilidad con la biblioteca del protocolo *PPP*.

Por otro lado, Android es una plataforma muy poco desarrollada para *BeagleBoard* y dificulta la ejecución de las rutinas para la solución final, además, aún no se ha desarrollado en esa plataforma las bibliotecas necesarias para la implementación de este proyecto.

Se eligió Ubuntu como la plataforma de desarrollo para la *BeagleBoard*. Esta plataforma era compatible con todas las bibliotecas necesarias para el desarrollo de la solución propuesta, además, a nivel general, es un sistema operativo bastante desarrollado por lo que facilita la implementación de diversas aplicaciones en un futuro.

5.1.3 Lenguaje de programación

Las diferentes rutinas de software para los módulos acoplados en la *BeagleBoard* se escribieron con los lenguajes de programación *Python*[1] *y Bash*.

El lenguaje de programación *Python* es muy versátil, robusto y fácil de usar. Además, contiene las bibliotecas para la lectura por puerto serial RS-232, las bibliotecas para envío y recepción de SMS y creación de archivos .csv, así como las bibliotecas SFTP.

El lenguaje *Bash,* propio de la plataforma de Linux, se utilizó para la integración de todas las rutinas escritas en *Python.*

5.1.4 Operadora de Red de telefonía celular

La conexión a Internet del sistema propuesto se da mediante la red de telefonía celular, por tal razón, la selección de la operadora depende del ancho de banda de la señal que se recibe en el lugar donde se colocará el sistema.

Las posibles operadoras son Movistar y Kolbi. Sin embargo, no se puede anticipar la selección de la operadora hasta que se mida la señal del ancho de banda en el lugar de pruebas.

---

[1] Documentación oficial de Python disponible en http://docs.python.org/

5.1.5 Redes de sensores inalámbricos

Durante la consulta preliminar con los encargados del grupo *eBridge* se dio a conocer que tenían a su disposición los módulos de recolección de datos de los sensores. La coordinación del proyecto seleccionó la red de sensores Waspmote para la implementación del sistema de monitorización. Estos dispositivos tienen un microcontrolador ATmega1281 que se conecta por medio de UART al módulo XBee 802.15.4, el cual se encarga de la comunicación inalámbrica. La transmisión inalámbrica de los datos de los sensores utiliza la frecuencia de banda libre de 2.4GHz y un ancho de banda de 5 MHz. Con esta información obtenida sólo se requería escribir rutinas para acoplar el módulo de recepción de los datos a la *BeagleBoard*.

5.2 Descripción del hardware

En esta sección, se detalla el hardware de cada uno de los módulos que componen el sistema. En la siguiente figura, se muestra el diagrama de bloques de los módulos de la implementación:

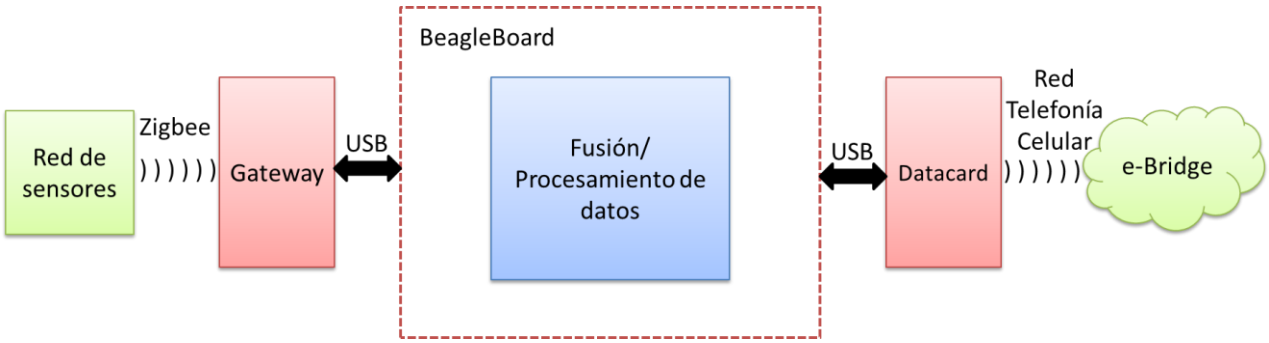


Figura 22 Diagrama de bloques de la solución implementada

5.2.1. Red de sensores

Los dispositivos utilizados son los Waspmote (Ver Figura 6), como se menciona en el apartado del Marco Teórico, estos dispositivos son de arquitectura modular, los cuales se les acoplan los módulos necesarios para una aplicación específica. Los Waspmote poseen un microcontrolador ATmega1281, el cual trabaja a una frecuencia de 8 MHz [6]. Al dispositivo se le acopló un módulo XBee 802.15.4, el cual se conecta con el microcontrolador por medio del puerto UART0 a una velocidad de 38400 Baud [6]. Este módulo transmite de manera inalámbrica los datos recolectados por los sensores.

En este proyecto sólo se implementaron sensores de aceleración (acelerómetros). Estos sensores se encuentran acoplados en los dispositivos Waspmote. Los acelerómetros utilizados pueden detectar dos cambios: caída libre y cambio de dirección. Entonces, se fijan esos sensores en puntos estratégicos del puente para medir la vibración de la estructura.

En la siguiente imagen se muestra el acelerómetro implementado en el Waspmote:

Figura 23 Acelerómetro [6]

5.2.2. Gateway

El módulo de recepción *Gateway* de la empresa *Libelium* es un dispositivo que se conecta vía USB a la *BeagleBoard*. Este módulo (ver Figura 10) emplea el protocolo XBee 802.15.4 para la recepción de los datos. Cada nodo puede estar a una distancia máxima de 100 metros del *Gateway* para que se logre la comunicación entre ambos. El *Gateway* solo recibe información de los nodos y se comunica vía serial con el sistema al que se encuentre acoplado (en este caso la *BeagleBoard*). Cabe destacar que en esta configuración de red, se utiliza una topología en estrella, en donde cada nodo se comunica solamente con el nodo central.

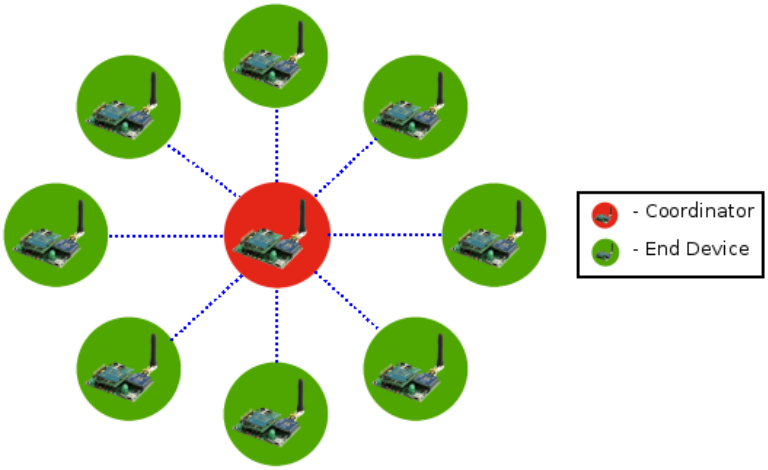


Figura 24 Topología estrella implementado con Waspmote [6]

5.2.3. BeagleBoard

La tarjeta de desarrollo escogida es la *BeagleBoard* (ver Figura 4). Este contiene los módulos necesarios que permiten acoplar el *Gateway* y el *Datacard* mediante la conexión USB. Además, posee el módulo para insertar una tarjeta microSD que almacena el S.O Ubuntu y un conector Ethernet para descargar las bibliotecas necesarias para la ejecución de este proyecto. Por último, para realizar diferentes pruebas, se encuentra acoplado el módulo serial RS-232. En esta tarjeta se programa las rutinas de recolección, fusión, procesamiento y envío de datos de forma automatizada. En la siguiente figura se encuentran señalados los módulos utilizados de la *BeagleBoard*:

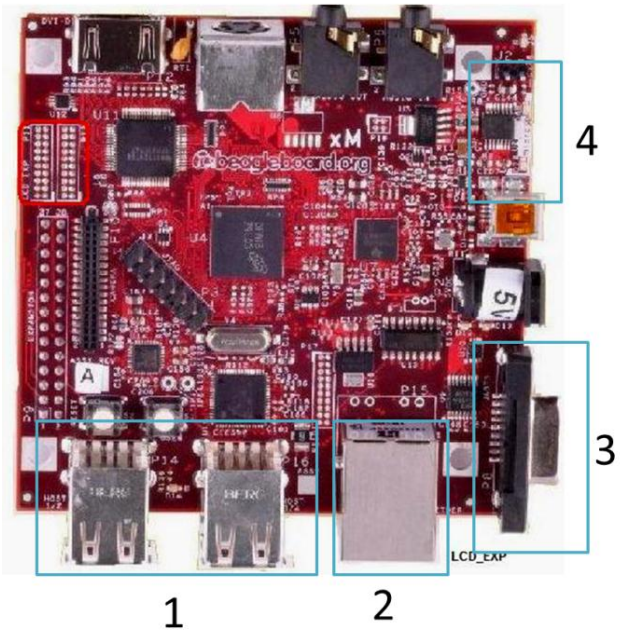


Figura 25 1: Módulos USB. 2: Módulo Ethernet. 3: Módulo RS-232. 4: Módulo microSD

5.2.4. Datacard

Este dispositivo se utiliza como módem HSDPA para la conexión a Internet mediante la red de telefonía móvil. Además, se emplea para recibir mensajes SMS para el inicio de recolección de datos y enviar mensajes SMS de confirmación o de error de carga de archivo al usuario. El modelo del dispositivo empleado es Huawei E166 (ver Figura 16) y al igual que el *Gateway*, se acopla a la *BeagleBoard* mediante la conexión USB.

5.3 Descripción del software

En este apartado se detalla cada una de las rutinas de software implementadas en las diferentes etapas del sistema. Para mejorar el entendimiento de cada algoritmo se utiliza diagramas de flujo.

Inicialmente, se explica la implementación del S.O a la tarjeta, seguidamente se explica la configuración inicial del S.O que debe realizarse solamente la primera vez que se utiliza el sistema desarrollado.

Posteriormente, se da una explicación desde un punto vista general del sistema, y por último, se divide el sistema en apartados. De esta manera, se explicará cada uno de forma más detallada.

5.3.1. Implementación del Sistema Operativo Ubuntu

Para almacenar el sistema operativo se utilizó una tarjeta microSD de 2GB. La tarjeta se prepara en una PC con Ubuntu instalado. Los siguientes pasos se deben ejecutar en la línea de comandos para preparar dicha tarjeta [33]:

a) Se obtiene la imagen del sistema operativo de la siguiente página, se destaca que la versión utilizada de Ubuntu corresponde a la 11.04:

wget *http://rcn-ee.net/deb/rootfs/natty/ubuntu-11.04-r7-minimal-armel.tar.xz*.

b) Posteriormente se descomprime la imagen:

tar xJf ubuntu-11.04-r7-minimal-armel.tar.xz

cd ubuntu-11.04-r7-minimal-armel.tar.xz

c) Se prepara la tarjeta con el script que viene incorporado en la imagen:

*sudo ./setup_sdcard.sh --mmc /dev/*** --uboot beagle_xm*

****Ubicación de la tarjeta, en algunas computadoras es mmcblk0 y en otras es sdb0. Para verificar la ubicación, ejecutar antes de este paso: sudo ./setup_sdcard.sh --probe-mmc*

Se utilizó un cable de pruebas Serial-USB para la comunicación entre la tarjeta y la computadora para el monitoreo de la primera. En la línea de comandos se ejecuta la siguiente orden:

sudo screen /dev/ttyUSB 115200*

*:0-9 el que se encuentre acoplado el cable serial

Con la orden anterior, se visualiza la consola instalada en la *BeagleBoard.*

El nombre de usuario corresponde a *ubuntu* y la clave *temppwd.*

5.3.2. Configuración Inicial del Sistema Operativo

Al inicializar por primera vez el Ubuntu, el usuario debe configurar algunos archivos para realizar las siguientes tareas:

- Automatización de los procesos
- Conexión de los módulos USB al directorio /dev/
- Ajuste zona horaria
- Ajuste de la fecha y hora al inicio del sistema

Durante el diseño de este proyecto, se creó todos los archivos para ejecutar las tareas mencionadas anteriormente y se integró todos en un script ejecutable para evitar la configuración manual. El script se encuentra en el apartado de Apéndices.

La configuración inicial se requiere de Internet, por lo que se requiere un cable Ethernet. En la consola de la tarjeta se debe introducir el comando para habilitar la conexión:

sudo dhclient eth0

Con la conexión establecida, se procede a ejecutar el script.

5.3.2.1. Automatización de los procesos

Para automatizar los procesos de la sección 5.3.3, se edita el archivo /etc/crontab. El crontab es un simple archivo de texto con una lista de comandos que se deben ejecutar en un tiempo específico [33]. Para ejecutar un comando o una tarea a la hora específica se agrega la siguiente línea en el archivo:

minutos hora dia mes diasemana tarea

minutos(0-59), hora (0-23, 0 = medianoche), día (1-31), mes (1-12), diasemana (0-6, 0 = Domingo), comando

Se puede escribir un asterisco (*), en caso de que se necesita ejecutar una tarea en cada momento (cada hora, cada minuto, etc.) dentro de un periodo de tiempo. Ejemplo:

01 04 * * * tarea ##Se ejecuta la tarea a la 1:04 AM durante todos los días

Este archivo se configura para ejecutar el script principal de la sección 5.3.3 cada tres minutos.

5.3.2.2. Conexión de los módulos USB al directorio /dev/

Al conectar el *Gateway* y el *Datacard* a la *BeagleBoard,* el sistema los acopla al directorio /dev/ttyUSB0, /dev/ttyUSB1 ó /dev/ttyUSB2 de manera aleatoria. Esta forma crea problemas al ejecutar los scripts que ocupan acceso hacia estos dispositivos. Por lo tanto, se escribe unas reglas de manera que el *Gateway* y el *Datacard* se acople al directorio */dev/gateway y /dev/datacard,* respectivamente.

Como se menciona anteriormente, se crea una regla llamada 75-tty-description en el directorio /etc/udev/rules.d [34] con la siguiente información:

```
ACTION=="remove", GOTO="tty_end"

SUBSYSTEM!="tty", GOTO="tty_end"

SUBSYSTEMS=="usb", ENV{ID_MODEL}=="", IMPORT{program}="usb_id --export
%p"

SUBSYSTEMS=="usb", ENV{ID_MODEL_FROM_DATABASE}=="", IMPORT{program}="usb-
db %p"

SUBSYSTEMS=="usb", ATTRS{idVendor}!="", ATTRS{idProduct}!="",
ENV{ID_VENDOR_ID}="$attr{idVendor}", ENV{ID_MODEL_ID}="$attr{idProduct}"

SUBSYSTEMS=="usb", DRIVERS=="usb",
ATTRS{idVendor}=="12d1",ATTRS{idProduct}=="1001",
ENV{ID_VENDOR_ID}="$attr{idVendor}", ENV{ID_MODEL_ID}="$attr{idProduct}",
SYMLINK+="datacard"

SUBSYSTEMS=="usb", DRIVERS=="usb",
ATTRS{idVendor}=="0403",ATTRS{idProduct}=="6001",
ENV{ID_VENDOR_ID}="$attr{idVendor}", ENV{ID_MODEL_ID}="$attr{idProduct}",
SYMLINK+="gateway"

SUBSYSTEMS=="usb", GOTO="tty_end"

SUBSYSTEMS=="pci", ENV{ID_MODEL_FROM_DATABASE}=="", IMPORT{program}="pci-
db %p"

SUBSYSTEMS=="pci", ENV{ID_BUS}="pci", ENV{ID_VENDOR_ID}="$attr{vendor}",
ENV{ID_MODEL_ID}="$attr{device}"

LABEL="tty_end"
```

El código anterior, al insertarle el código del producto y del fabricante de cada dispositivo y los acopla al directorio */dev/gateway y /dev/datacard*.

Para observar la información de los códigos se ejecuta el comando *lsusb.*

5.3.2.3. Ajuste zona horaria

Se ejecuta el siguiente comando [35]:

ln -s /usr/share/zoneinfo/America/Costa_Rica /etc/localtime.

El comando anterior, ajusta la zona horaria local a la zona horaria de Costa Rica. En el directorio */usr/share/zoneinfo* se encuentran todas las zonas horarias alrededor del mundo.

5.3.2.4. Ajuste de la fecha y hora al inicio del sistema

Al inicio del sistema se ajusta automáticamente el reloj del sistema. Se ejecuta un comando que conecta el sistema a internet (ver sección 5.3.3.4) y posteriormente sincroniza la fecha del sistema a un servidor.

En la siguiente figura se observa el diagrama del script que se ejecuta al inicializar el sistema:

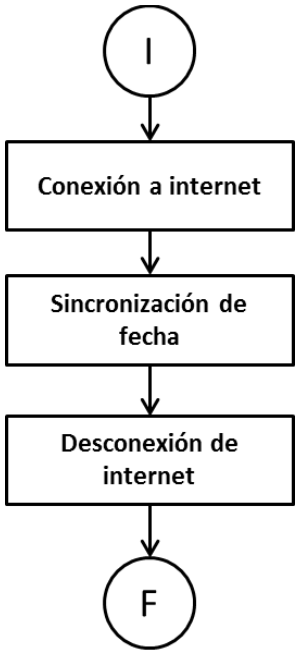


Figura 26 Script de configuración de fecha

El script de configuración se crea en el directorio /etc/init.d, para ejecutarlo al inicio del sistema se debe ejecutar el siguiente comando [36]:

update-rc.d NombreDelScript defaults 90

5.3.3. Sistema principal

La integración de todos los bloques de código se realiza con un script, el cual se implementó con el lenguaje *Bash*.

Inicialmente, el código verifica si actualmente existe un proceso activo, en caso afirmativo, el código se interrumpe. Si no existe proceso activo, el código ejecuta en orden los siguientes bloques:

- Autenticación (ver sección 5.3.3.1)
- Borrado de mensajes SMS (ver sección 5.3.3.2)
- Recolección de datos (ver sección 5.3.3.3)
- Conexión a Internet (ver sección 5.3.3.4)
- Conexión al servidor eBridge por SFTP (ver sección 5.3.3.5)
- Envío de notificación SMS (ver sección 5.3.3.6)

En la siguiente imagen se observa un diagrama de flujo detallado acerca del funcionamiento del sistema:

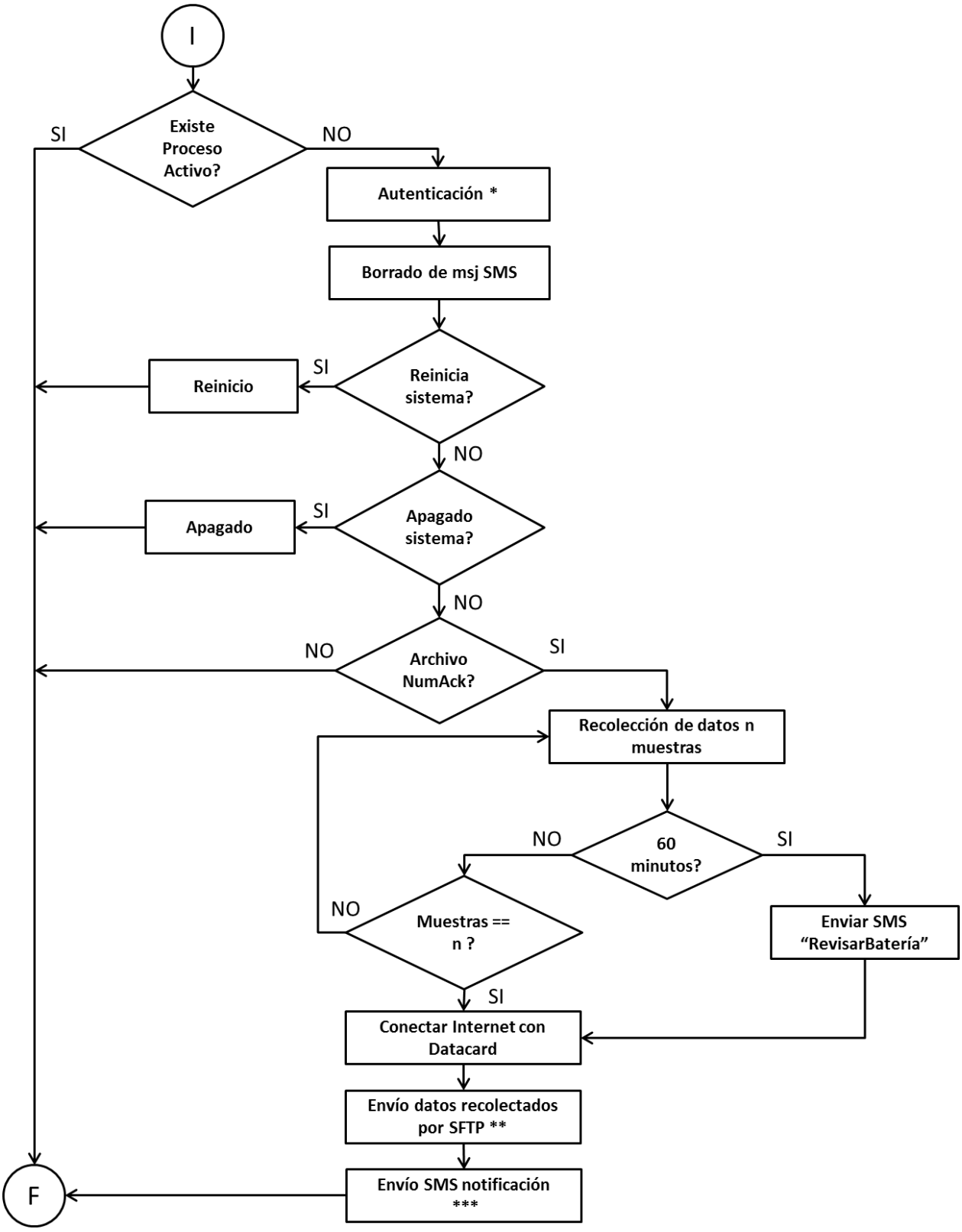


Figura 27 Script Integración

5.3.3.1. Autenticación

Las rutinas de software comienzan cuando un usuario envía un mensaje SMS al sistema. El formato del mensaje SMS recibido es: *contraseña acción.* En *acción* el usuario indica la cantidad muestras, apagado o reinicio.

El bloque de Autenticación se implementó con *Python*, el cual obtiene el primer mensaje SMS del buzón de la memoria de la tarjeta SIM (el código se interrumpe en caso de no existir mensajes).

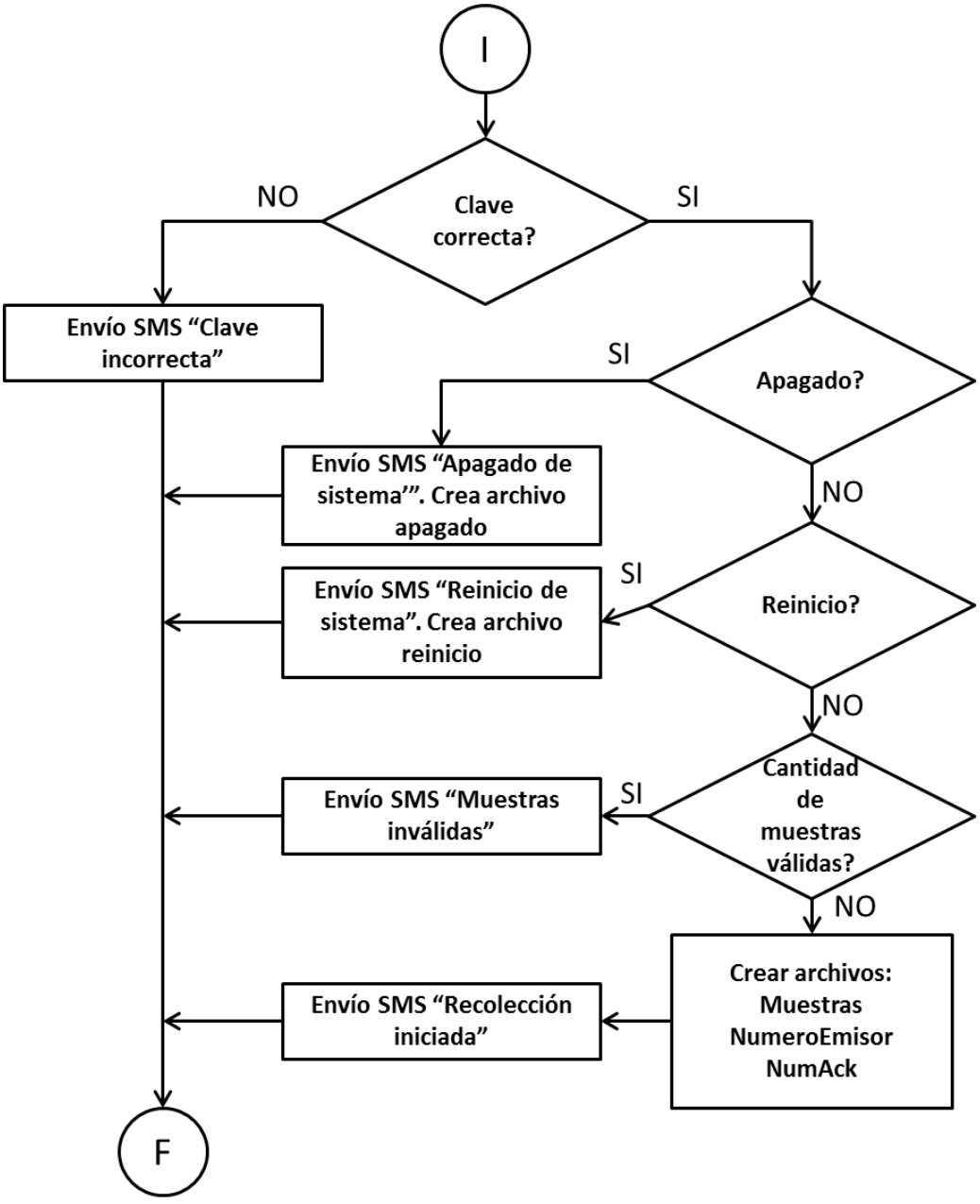


Figura 28 Script Autenticación

Según el diagrama de la Figura 28, inicialmente, el código verifica la contraseña enviada, en caso de ser incorrecta, envía al usuario un SMS indicando la invalidez de la misma. Sin embargo, si la contraseña es correcta, el código procede a ejecutar la acción que el usuario desea. Si la acción corresponde al reinicio o apagado, el sistema envía un SMS al usuario con el mensaje de confirmación de reinicio o apagado, respectivamente y crea un archivo temporal que se utiliza para indicar al script principal la acción a ejecutar (ver sección 5.3.3). Si la acción contiene letras o caracteres el sistema envía un SMS indicando invalidez de la muestra.

Por último, si la acción indica la cantidad de muestra, el sistema envía un SMS indicando el inicio de recolección de datos del sistema y crea tres archivos temporales*: NumeroEmisor* (número telefónico del usuario)*, Muestras* (cantidad de muestras que el usuario indica) *y NumAck* (indica que existe un proceso activo y evita que se ejecute el script principal en caso de un mensaje nuevo), estos tres archivos se utilizan en los bloques de códigos siguientes.

5.3.3.2. Borrado de mensajes SMS

Este bloque de código implementado en *Python*¸ borra todos los mensajes SMS del buzón de la memoria de la tarjeta SIM.

5.3.3.3. Recolección de datos

Este bloque contiene dos scripts escritos en el lenguaje de programación *Python.* El primer script abre el puerto serial del *Gateway* y luego comienza un ciclo de recolección de datos provenientes de los *Waspmote*. Este proceso recolecta la cantidad de muestra que el archivo temporal *Muestras* indica (ver Sección 5.3.3.1).

En caso de que algún nodo se queda sin batería, el código crea un archivo temporal (*baterias*) que lo indica. En caso de que no exista respuesta por parte de los nodos, este script se interrumpe en 60 minutos. El segundo script le da formato a los datos recolectados y crea un archivo .csv. El formato de los archivos se muestra en la Figura 29.

Hora	NodoID	X	Y	Z
12:54:16	A	-1	-179	1071
12:54:16	B	35	-104	1023
12:54:16	A	-1	-180	1080
12:54:16	B	36	-104	1029
12:54:16	A	0	-179	1080
12:54:16	B	37	-102	1027
12:54:16	A	-1	-180	1075
12:54:16	B	36	-105	1021
12:54:16	A	-2	-179	1079
12:54:16	A	0	-182	1084
12:54:16	B	36	-103	1026

Figura 29 Formato de los archivos .csv

La rutina de recolección de datos se muestra en el siguiente diagrama de flujo:

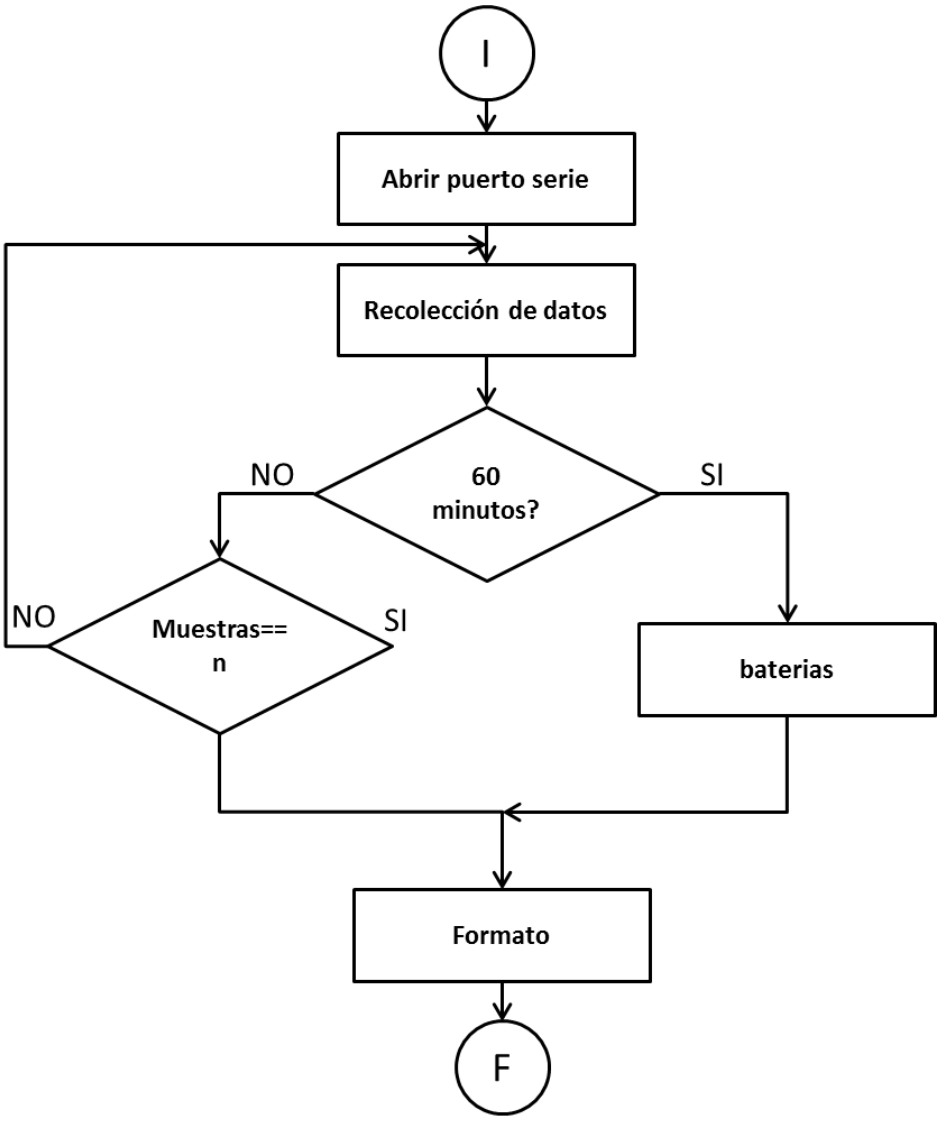


Figura 30 Script de Recolección de datos

5.3.3.4. Conexión a internet

Como se menciona en el apartado del Marco teórico, se debe establecer el protocolo de comunicación PPP entre el *Datacard* y la tarjeta de desarrollo *BeagleBoard.*

Se implementó la biblioteca PPP en Ubuntu. Como se menciona en la sección 3.3.1, se debe crear dos archivos: archivo de configuración y archivo de comunicación con el módem.

El archivo de configuración creado en el directorio /etc/ppp/peers se denomina *conexionHSDPA* y posee la siguiente configuración:

```
/dev/datacard
115200
noauth
usepeerdns
modem -detach
crtscts
asyncmap 0
defaultroute
noipdefault
lock
connect "/usr/sbin/chat -v -f /etc/ppp/chatICE"
```

El archivo de comunicación *chatHSDPA* creado en el directorio /etc/ppp contiene la siguiente información:

```
ABORT BUSY
ABORT 'NO CARRIER'
ABORT VOICE
ABORT 'NO DIALTONE'
ABORT 'NO DIAL TONE'
ABORT 'NO ANSWER'
ABORT DELAYED
""    'ATZ'
OK    'AT&F'
OK    'ATQ0 V1 E1'
OK    'AT&D2 &C1'
OK    'AT+FCLASS=0'
OK    'ATS0=0'
OK    AT+CGDCONT=1,"IP","APN de la OPERADORA"
OK    AT+CGATT=1
OK    ATD*99***1#
CONNECT '\c'
```

De acuerdo al código anterior, el APN de la operadora Kolbi es *kolbi3g,*y de la operadora Movistar es *internet.movistar.cr.*

Con los archivos correctamente creados, se ejecuta el comando *pon conexionHSDPA* para iniciar la conexión a Internet y el comando *poff* para finalizar la conexión [37].

5.3.3.5. Conexión al servidor eBridge por SFTP

Este script, escrito en Python, implementa las bibliotecas SFTP para la conexión del sistema al servidor de eBridge. La rutina se muestra en la Figura 31.

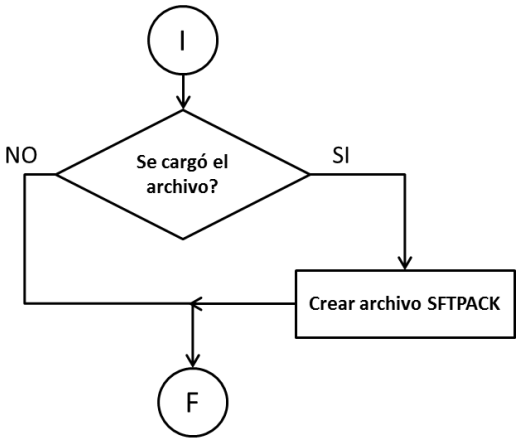


Figura 31 Script carga de archivo por SFTP

En caso de que la carga del archivo al servidor fue exitosa, este código crea un archivo temporal de confirmación *SFTPACK,* el cual se utiliza en el siguiente bloque.

5.3.3.6. Envío de notificación SMS

Este último bloque notifica al usuario el resultado de la carga del archivo al servidor. En caso de que exista el archivo temporal *SFTPACK* (ver sección 5.3.3.5). El sistema envía un SMS con el enlace del sitio web del archivo cargado. En caso de que exista el archivo *baterias* (ver sección 5.3.3.3), el sistema envía una advertencia al usuario, indicándole el estado de la batería de algún nodo. En caso de error, el sistema envía el mensaje notificando al usuario el error de carga de archivo. La Figura 32 muestra el proceso que ejecuta este script.

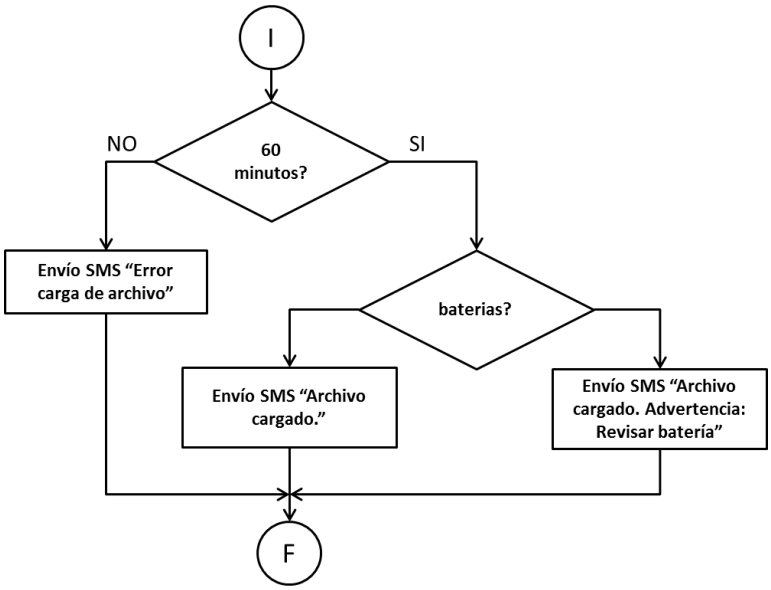


Figura 32 Script de envío de mensaje SMS de notificación

Capítulo VI. Análisis de Resultados

En este capítulo se presentan los resultados alcanzados junto con un análisis detallado en las diferentes pruebas desarrolladas con los módulos implementados. Cabe destacar, que para monitorear las pruebas se utilizó un cable serial-USB para conectar la *BeagleBoard* con una computadora PC.

El puente de pruebas se ubica en San Isidro, El Guarco de la provincia de Cartago. Se escogió esta estructura debido a que se encuentra sobre la carretera Interamericana, por lo que es un puente muy transitado.

6.1. Implementación del Sistema Operativo

Con la tarjeta debidamente preparada (ver sección 5.3.1), se inserta en la ranura microSD de la tarjeta *BeagleBoard* (ver Figura 25). Al encender la *BeagleBoard* se observa en la consola el *booteo* de la tarjeta:

```
U-Boot 2012.04.01-00005-g45939b4 (Apr 30 2012 - 09:34:17)

OMAP3630/3730-GP ES1.1, CPU-OPP2, L3-165MHz, Max CPU Clock 1 Ghz
OMAP3 Beagle board + LPDDR/NAND
I2C:   ready
DRAM:  512 MiB
NAND:  0 MiB
MMC:   OMAP SD/MMC: 0
*** Warning - readenv() failed, using default environment

In:    serial
Out:   serial
Err:   serial
Beagle xM Rev A
No EEPROM on expansion board
No EEPROM on expansion board
Die ID #5d5e00211ff00000015739eb0802e00b
Net:   Net Initialization Skipped
No ethernet found.
Hit any key to stop autoboot:  2
```

Figura 33 Boot BeagleBoard

Al terminar la inicialización de la tarjeta, ésta logra cargar el sistema operativo:

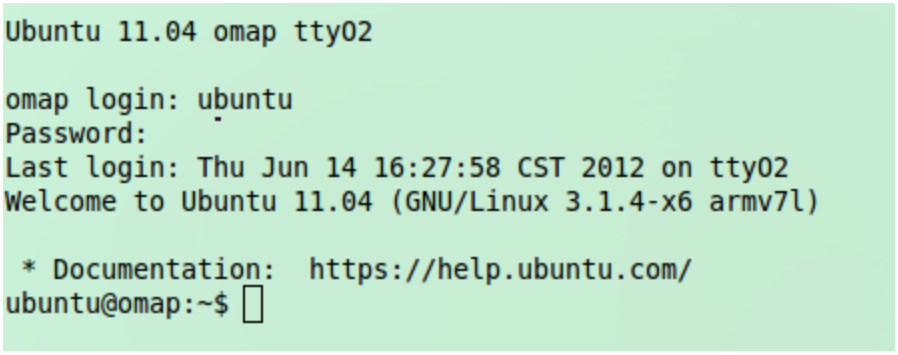


Figura 34 Consola BeagleBoard

6.2. Conexión a Internet

A partir del protocolo de enlace PPP entre el módem y la BeagleBoard, se logró la conexión a Internet. En la línea de comandos se ejecutó la orden *pon conexionHSDPA*. El resultado obtenido se muestra en la siguiente figura:

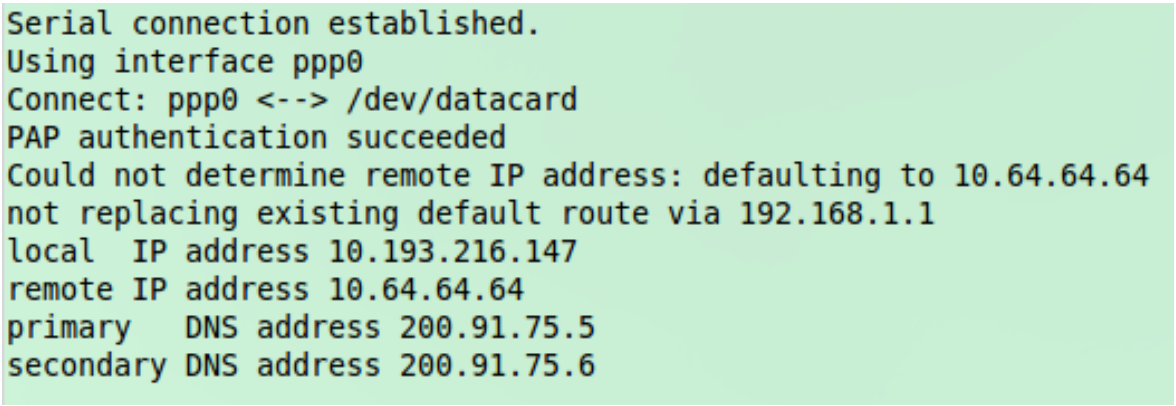


Figura 35 Captura de imagen de la conexión establecida a internet con el protocolo PPP

Una vez establecida la conexión a Internet a través del *Datacard,* se realizaron distintas pruebas de confiabilidad del enlace. El archivo de datos recolectados de mayor tamaño cargado en el servidor es de 977kB con 40000 muestras adquiridas.

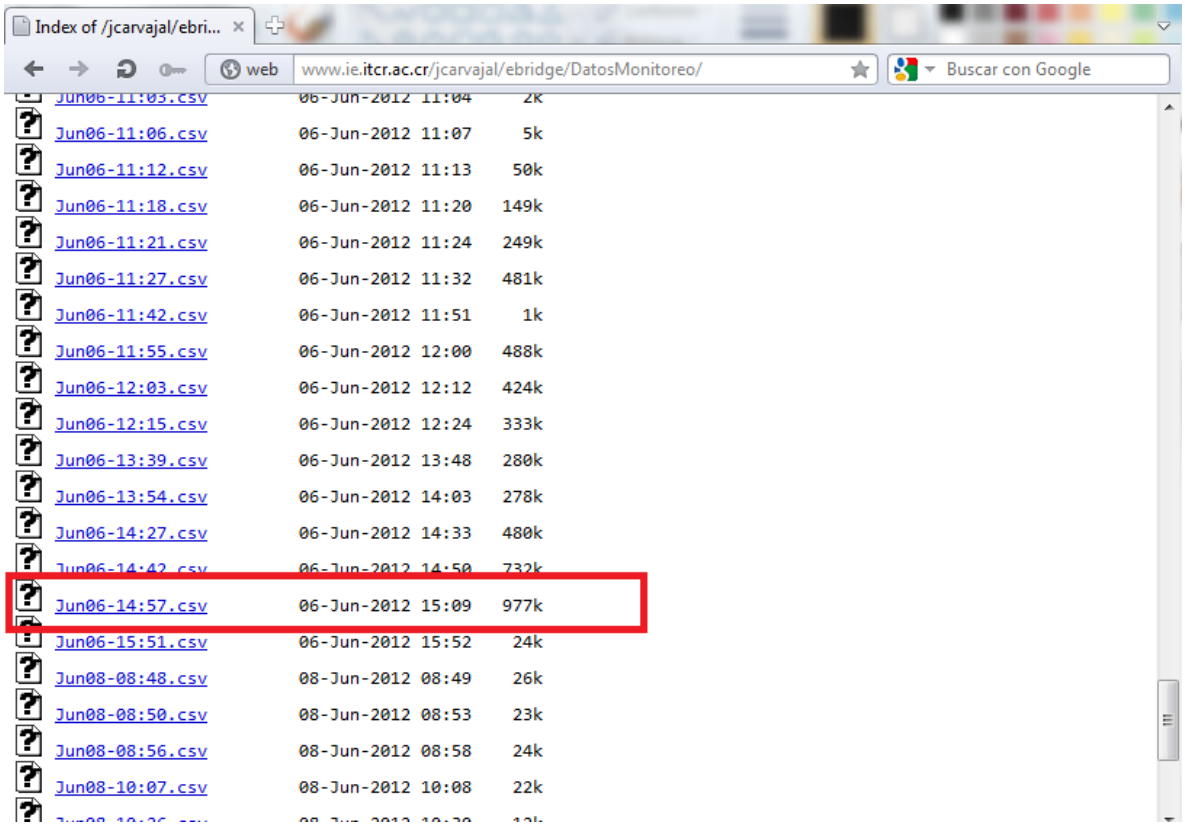


Figura 36 Carga exitosa del archivo con 40000 muestras

Además, se cargó exitosamente otro tipo de archivo con un tamaño aproximado de 5MB.

```
sftp> put Pumped\ Up\ Kicks.mp3
Uploading Pumped Up Kicks.mp3 to /home/jcarvajal/web/ebridge/Pumped Up Kicks.mp3
Pumped Up Kicks.mp3                           100% 4947KB   3.7KB/s   22:12
sftp>
```

Figura 37 Archivo cargado al servidor eBridge

Como se observa en la figura Figura 37, el archivo se cargó a una velocidad promedio de 3.7 KB/s (29.6kbps). La velocidad de carga del archivo depende de la señal de la operadora y la ubicación del sistema.

En distintos sitios ubicados alrededor del laboratorio se cargaron distintos archivos con velocidades entre 24kbps (3 KB/s) a 80kbps (10 KB/s) con la operadora Kolbi y con velocidades entre 12kbps (1.5 KB/s) a 120kbps (15 KB/s) con la operadora Movistar.

6.3. Ancho de banda de la señal

Como se menciona en la sección 5.1.4, la selección de la operadora depende del ancho de banda de la señal del puente. Se utilizó una aplicación para móviles, desarrollado por el Grupo ADSLZone, para medir la velocidad del ancho de banda de la señal telefónica [38]:

Figura 38 Aplicación móvil para medición de la velocidad de ancho de banda

Se midió la velocidad en el puente y los resultados obtenidos se observan en la siguiente tabla:

Tabla 6 Ancho de banda según Operadora y Plan [39] [40] [41]

Operadora	Plan	Velocidad de descarga esperado (kbps)	Velocidad de carga esperado (kbps)	Velocidad de descarga medido(kbps)	Velocidad de carga medido(kbps)
Kolbi	Internet por minuto	512	256	498	282
Kolbi	Internet por mes	1024	512	976	388
Movistar	Internet por Dia	1024	512	1018	393

Como se observa en la Tabla 6, las velocidades tanto de descarga como carga de ambas operadoras para el plan de 1Mbps son bastantes parecidas, por lo que, al cargar un archivo de 40000 muestras se requiere menos de 20 segundos. Mientras que utilizando el plan de 512 de Kolbi, cargar el mismo archivo duraría aproximadamente 27 segundos.

Por lo tanto, la selección de la operadora depende del costo económico de cada plan y de la cantidad del uso del sistema por parte de la coordinación de eBridge.

6.4. Pruebas del sistema principal en el laboratorio

Se realizaron diversas pruebas para determinar la confiabilidad del sistema. Como se menciona en la sección 5.3.3, el sistema puede enviar los siguientes mensajes SMS de notificación al usuario:

- Clave enviada incorrecta
- Cantidad de muestras inválidas
- Reinicio del sistema
- Apagado del sistema
- Recolección de datos iniciados
- Error en la carga de archivos
- Carga exitosa del archivo
- Carga exitosa del archivo con advertencia: Revisar baterías de los nodos

En las siguientes figuras se observa capturas de pantalla realizadas desde un teléfono celular con los mensajes de notificación del sistema:

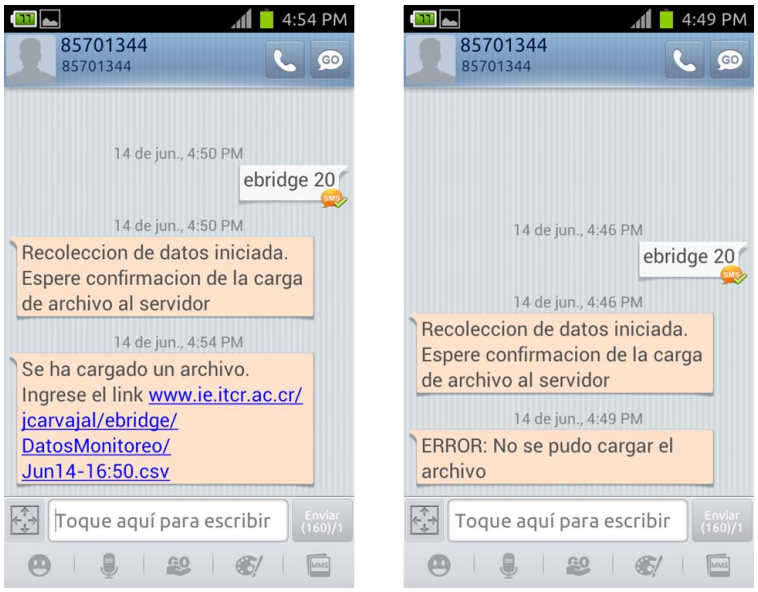


Figura 39 Capturas de pantalla de carga exitosa y error durante la carga

Figura 40 Carga exitosa del archivo con advertencia

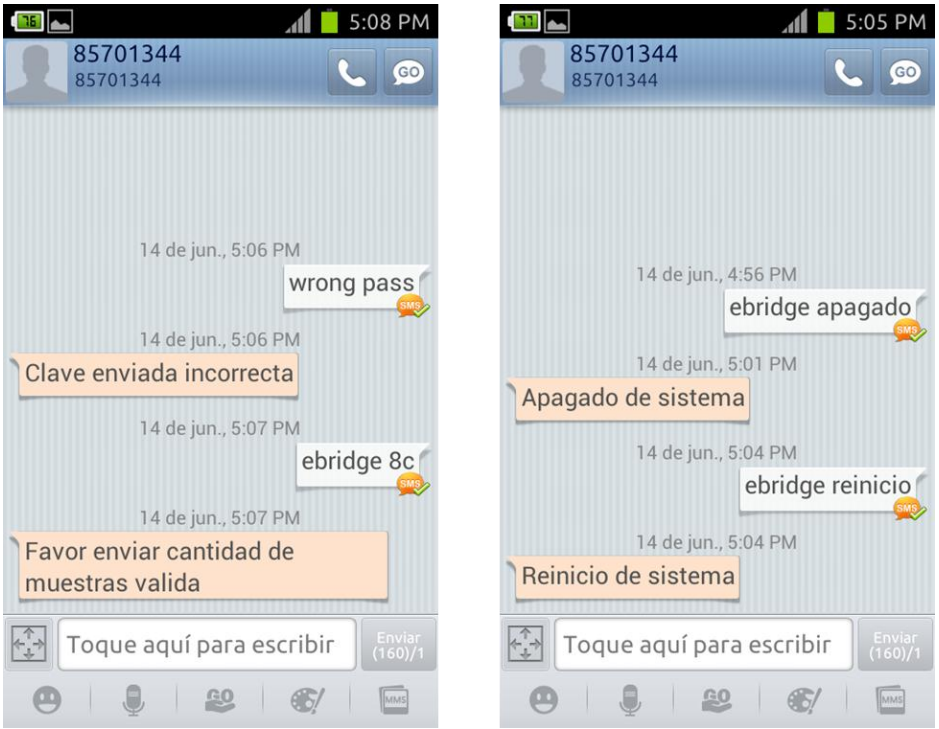


Figura 41 Capturas de pantalla de clave incorrecta, muestras inválidas, reinicio y apagado del sistema

En la siguiente tabla se muestra los resultados de diferentes pruebas realizadas al sistema principal:

Tabla 7 Pruebas realizadas y porcentaje de éxito

Prueba realizada	Cantidad de pruebas	Porcentaje de pruebas exitosas
Envío de Clave incorrecta	15	100%
Envío de apagado y reinicio del sistema	15	100%
Envío de muestras inválidas	15	100%
Inicio recolección de datos	130	95.4%
Pruebas totales	**175**	**96.57%**

En la prueba de inicio de recolección de datos puede presentarse tres situaciones: Carga exitosa del archivo, error en la carga de archivo o el sistema no contesta al usuario. En la siguiente tabla muestra las estadísticas de las pruebas:

Tabla 8 Estadísticas de la prueba "Inicio de recolección de datos" en el laboratorio

	SMS Archivo cargado	SMS Error de carga	SMS no contestados
Cantidad	**110**	**14**	**6**
Porcentaje	**84.6%**	**10.8%**	**4.6%**

Los archivos cargados se observan en la página del servidor de eBridge[2]:

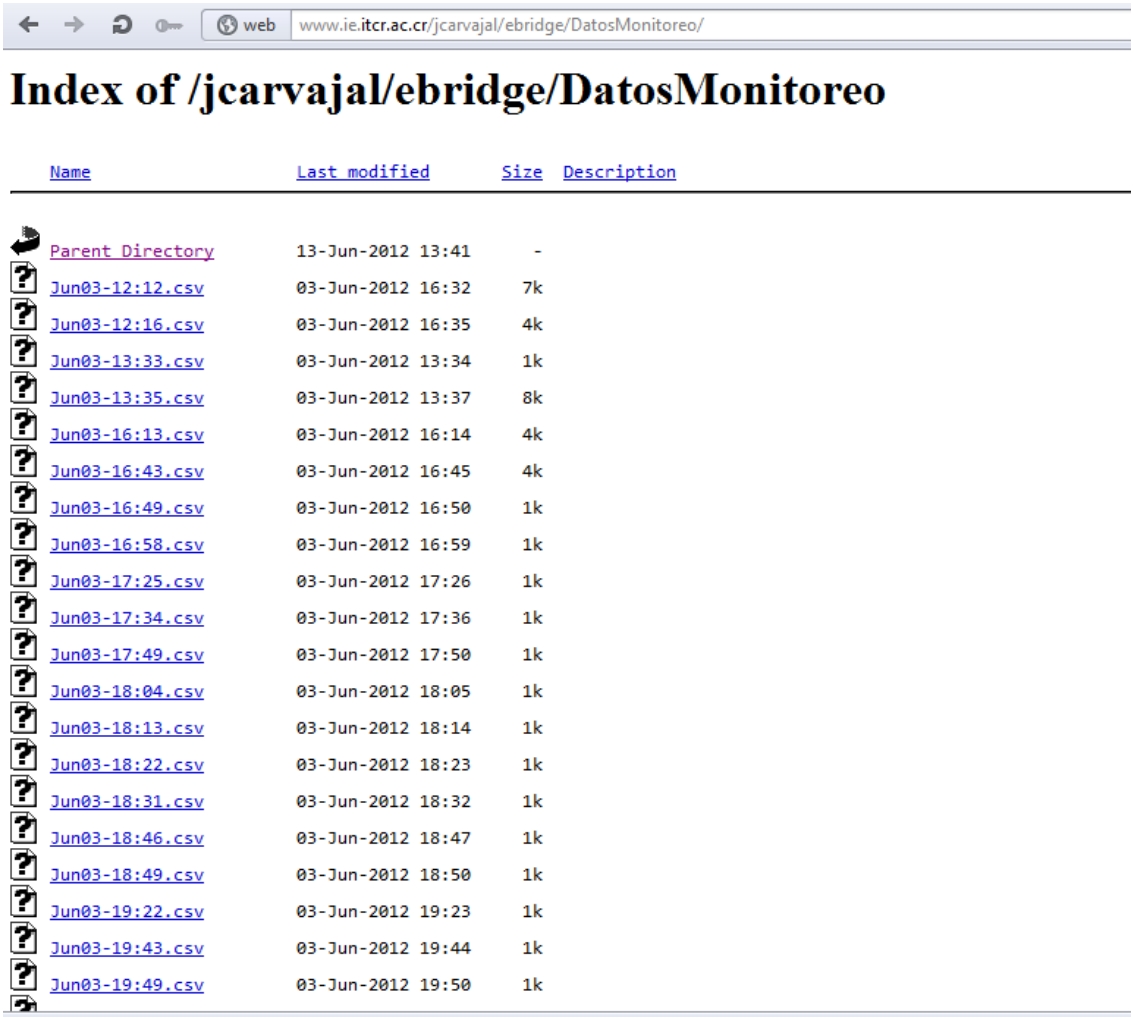
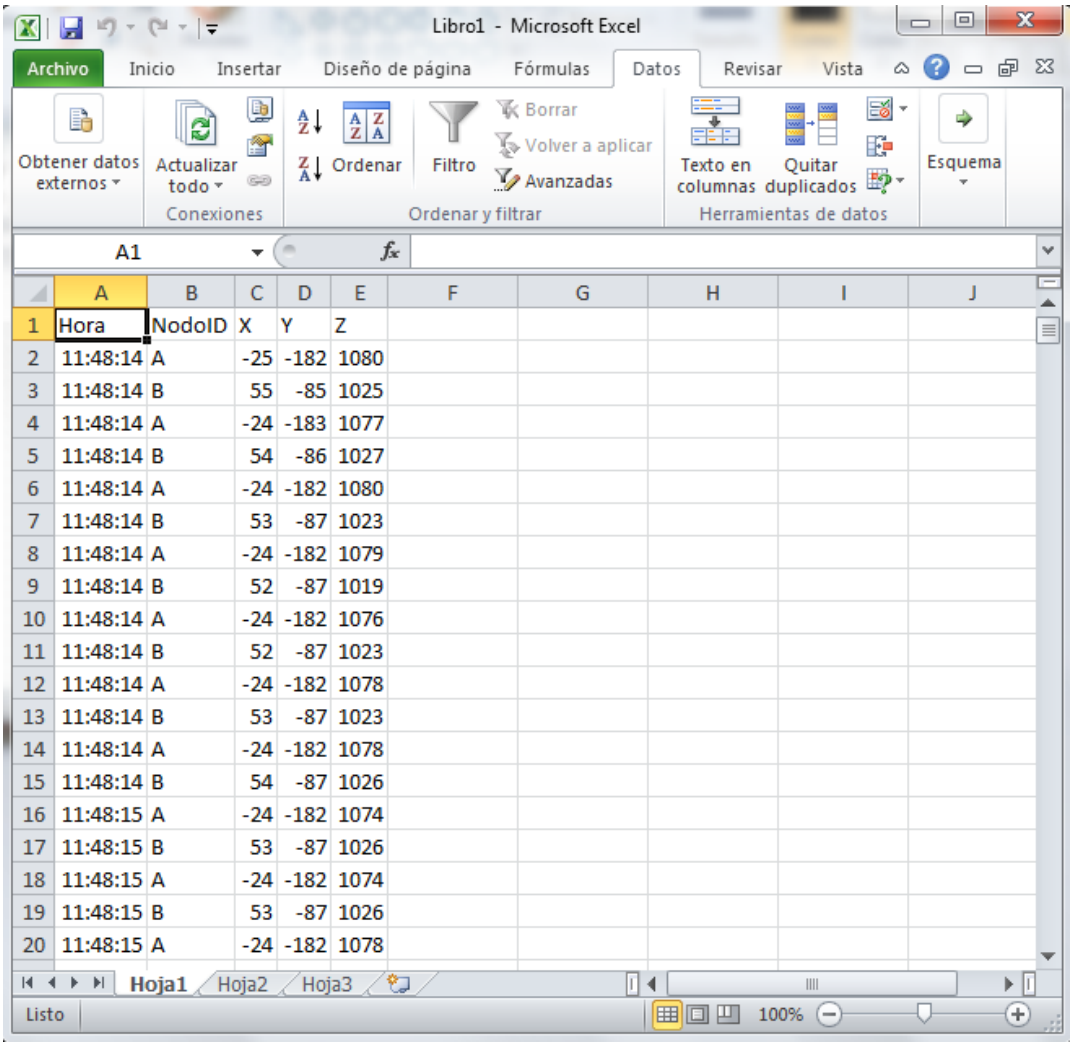


Figura 42 Archivos cargados al servidor eBridge

El tipo de archivo cargado es en .csv, el formato se encuentra en la siguiente figura:

[2] http://www.ie.itcr.ac.cr/jcarvajal/ebridge/DatosMonitoreo/



	A	B	C	D	E
1	Hora	NodoID	X	Y	Z
2	11:48:14	A	-25	-182	1080
3	11:48:14	B	55	-85	1025
4	11:48:14	A	-24	-183	1077
5	11:48:14	B	54	-86	1027
6	11:48:14	A	-24	-182	1080
7	11:48:14	B	53	-87	1023
8	11:48:14	A	-24	-182	1079
9	11:48:14	B	52	-87	1019
10	11:48:14	A	-24	-182	1076
11	11:48:14	B	52	-87	1023
12	11:48:14	A	-24	-182	1078
13	11:48:14	B	53	-87	1023
14	11:48:14	A	-24	-182	1078
15	11:48:14	B	54	-87	1026
16	11:48:15	A	-24	-182	1074
17	11:48:15	B	53	-87	1026
18	11:48:15	A	-24	-182	1074
19	11:48:15	B	53	-87	1026
20	11:48:15	A	-24	-182	1078

Figura 43 Formato de los archivos cargados

En la Tabla 7 se observa que se realizó 175 pruebas, de las cuales indica que la confiabilidad del sistema en el laboratorio supera los 96%. Las pruebas fallidas son de la prueba de “Inicio de recolección de datos”. En la Tabla 8, se muestra con detalles los resultados de esa prueba.

En esta tabla, se observa un pequeño porcentaje de mensajes no contestados (4.6%), del cual se ocasiona una situación específica. Como se menciona en la sección 5.3.3.1, el sistema sólo obtiene el primer mensaje de texto del buzón, por lo que, este problema lo ocasiona cuando dos o más usuarios envían simultáneamente los mensajes de textos. Por ejemplo, si el usuario A y el usuario B envían los mensajes al mismo tiempo, el sistema procesa el mensaje del usuario A, pero el mensaje del usuario B nunca llega a procesarse, ya que según la Figura 27, hay un bloque de Borrado SMS, por lo que, ambos mensajes se borran.

Por lo tanto, se tiene que esperar alrededor de 15-20 segundos (dependiendo del centro de mensajes de la operadora) entre un mensaje y otro, esta cantidad de tiempo

permite al sistema autenticar el usuario y posteriormente, borrar todos los mensajes de textos del sistema. Siguiendo el ejemplo anterior, si el usuario B envía el mensaje después de la ventana de tiempo de 15-20 segundos, el sistema procesará dicho mensaje hasta que se termine la recolección y carga del archivo del usuario A.

El 10.8% de los archivos no cargados no se consideran como error de prueba, ya que no se lograron cargar los archivos debido a la conexión de Internet. En la sección 6.2 se explica que el éxito de carga del archivo se relaciona con la ubicación del sistema y la operadora de telefonía. En la Figura 39, se observa los mensajes de confirmación del sistema de la carga.

6.5. Pruebas del sistema principal en el puente

Se instaló la *BeagleBoard* en una caja de metal para protegerla de cualquier factor externo como golpes, humedad, etc. El sistema se colocó cerca del puente y se realizaron las pruebas, las cuales principalmente consisten en el inicio de recolección de datos.

Figura 44 Sistema diseñado realizando pruebas en el puente

Se realizaron veinte pruebas de las cuales se dividen en doce pruebas de recolección con la operadora Kolbi, y ocho pruebas con la operadora Movistar, las estadísticas se muestran en la siguiente tabla:

Tabla 9 Estadísticas de la prueba "Inicio de recolección de datos" en el puente

Operadora	SMS Archivo cargado	SMS Error de carga	SMS no contestados
Kolbi	7	4	1
Movistar	7	1	0
% Exito	**70%**	**25%**	**5%**

Según las estadísticas de la Tabla 9, 14 archivos se cargaron exitosamente al servidor (70%), y el 25% corresponde el error de carga y solo un mensaje no se llegó a contestar. El mensaje no contestado es el mensaje que se envió dentro de la ventana de 15-20 segundos comentado en la sección 6.4. Además, como se comentó en la misma sección, el error de carga no se considera como error de prueba ya que por motivos de la señal de la operadora no se lograron cargar correctamente los archivos, en la Figura 39 se observa los mensajes de carga exitosa y error de carga, respectivamente.

Por lo tanto, según los resultados de la Tabla 9, la confiabilidad del sistema alcanza un 95%.

Como se muestra en la Tabla 8 y en la Tabla 9, la única limitación del sistema es la ventana del tiempo de 15-20 segundos, de manera que, si se reduce este tiempo los errores durante la prueba se disminuye.

6.6. Alcance de los objetivos específicos

El objetivo específico 1 se alcanzó con la solución descrita en la sección 5.3.1, la razón por la cual se implementó la plataforma Ubuntu es facilitar el manejo de todos los módulos utilizados en el proyecto.

Seguidamente, el objetivo específico 2 se logró con las soluciones descritas en la sección 5.3.3.3, se recolectaron la cantidad de datos especificadas por el usuario, además se le dio el formato indicado por la coordinación de eBridge. En seguida, los protocolos de transferencia de datos, se implementaron en las secciones 5.3.3.4 y 5.3.3.5. Además, se realizaron pruebas de confiabilidad del enlace del sistema, las cuales sobrepasan el 95% de éxito.

Por último, las rutinas de integración y automatización de los procesos se implementaron con las soluciones descritas en las secciones 5.3.3 y 5.3.2.1.

6.7. BeagleBoard xM vs. Computadora PC

A partir de toda la descripción detallada de la solución y del análisis de resultados, se observa que la tarjeta *BeagleBoard* realiza las mismas funciones de la PC. Por lo tanto, se deben analizar los factores que intervienen en el proyecto, para determinar si una tarjeta de desarrollo es mejor que una computadora.

1) Costo económico: La tarjeta cuesta 150$, mientras que la PC supera los $500. Además, la conexión a Internet implementado en la tarjeta tiene menor costo que el internet por *Ethernet* de la computadora PC.

2) Sistema optimizado: La plataforma Ubuntu implementada en la tarjeta es una versión mínima que posee los módulos y las bibliotecas más básicas para inicializar el sistema, mientras que la plataforma para el PC ya poseen controladores como el ratón, la pantalla, impresora, etc. En la tarjeta se instala las bibliotecas y los módulos necesarios que requiere el sistema diseñado, de esta manera, se optimiza el uso de la memoria del Sistema Operativo. Además, la optimización del sistema conlleva a que la tarjeta sólo realice solamente funciones específicas, sin embargo, esto permite ahorrar recursos.

3) Portabilidad: En la sección 6.5 se menciona que se instaló la tarjeta dentro de una carcasa negra de metal. Las dimensiones de esta caja negra es 3.13" x 3.25" x 1.3"H, las cuales permiten colocar el sistema en cualquier lugar no mayor a 100 metros del puente (Por ejemplo: Poste de luz, o debajo del puente). Por otro lado, si se coloca una computadora cerca del puente se requiere construir una casetilla para evitar el robo del aparato. Además, la conexión a internet en la computadora PC se estableció mediante el *Ethernet*, lo cual no es práctico en el puente puesto que se requiere implementar un módem aparte y conectar diversos cables para establecer la conexión. En la tarjeta se le acopló mediante USB un *Datacard*¸ el cual utiliza la red de telefonía móvil (Red inalámbrica) para establecer la conexión a internet.

Capítulo VII. Conclusiones y recomendaciones

7.1. Conclusiones

1. El procesador de 1GHz logró ejecutar el S.O Ubuntu 11.04 y todas las rutinas implementadas en el proyecto.
2. La implementación de las rutinas con el lenguaje *Python* facilita la integración de nodos adicionales a la red de sensores inalámbricos.
3. Se debe implementar el protocolo de comunicación PPP para comunicar cualquier módem telefónico (dispositivo esclavo) con un dispositivo maestro (*BeagleBoard*)
4. La confiabilidad de la conexión a internet así como el éxito de la carga de los archivos depende de la operadora de telefonía celular y la ubicación del sistema.
5. La confiabilidad del sistema se puede aumentar reduciendo la ventana de tiempo de 15-20 segundos.
6. Se incorporó una función que permite al usuario controlar la cantidad de muestras a recolectar, esto mediante el envío de mensajes de texto SMS
7. La sustitución de la PC por una tarjeta de desarrollo disminuye costos económicos.
8. La implementación de la tarjeta corresponde a una versión optimizada del sistema actual, ya que consume menos recursos de memoria y de energía.
9. El sistema diseñado es portátil y de fácil instalación por sus reducidas dimensiones físicas.
10. El sistema diseñado es más práctico por la selección del *Datacard* como medio de conexión a Internet.

7.2. Recomendaciones

1. Reducir la ventana de tiempo de 15-20 segundos implementando un ciclo de búsqueda de mensajes en todo el buzón de mensajes de la memoria SIM.

2. Realizar más pruebas con otras operadoras para establecer la mejor velocidad de carga: *Claro, FullMóvil, Tuyo y Telefónica.*

3. Adquirir una batería portátil para alimentación eléctrica del sistema.

4. Diseñar una rutina que monitoree el estado de batería de los nodos y de la batería portátil de la tarjeta. De esta manera, si llega a un límite establecido, se envía una notificación a la coordinación de eBridge

5. Desarrollar un código que coloque la tarjeta en modo de bajo consumo mientras no se procese ningún mensaje de texto.

6. Desarrollar una aplicación para la *BeagleBoard* de manera que se pueda desplegar los datos recolectados en tiempo real y observar el comportamiento de los mismos datos con respecto al tiempo en una gráfica desde una aplicación web o desde una aplicación móvil.

7. Implementar y probar el diseño final en otros tipos de monitorización. Ejemplo: Invernadero, bosques, estructuras en los estadios, entre otros.

8. Agregar una rutina del reinicio del sistema cada 8 horas.

9. Agregar una rutina de verificación de archivo mediante el programa md5sum. Esta rutina consiste en que el sistema envía el valor md5sum al usuario para determinar si el archivo descargado es el correcto.

Referencias

[1] BeagleBoard.org, «Product Details (for the original BeagleBoard),» [En línea]. Available: http://beagleboard.org/hardware. [Último acceso: 6 Febrero 2012].

[2] Texas Instruments. «OMAP3530/25 Applications Processor,» 2009. [En línea]. Available: http://www.ti.com/lit/ds/symlink/omap3530.pdf. [Último acceso: 6 Febrero 2012].

[3] Linux Devices, «$150 board sports Cortex-A8,» 2009. [En línea]. Available: http://www.linuxfordevices.com/c/a/News/150-board-sports-CortexA8/. [Último acceso: 6 Febrero 2012].

[4] G. Coley, «BeagleBoard-xM Rev C System Reference Manual,» 2010. [En línea]. Available: http://beagleboard.org/static/BBxMSRM_latest.pdf. [Último acceso: 6 Febrero 2012].

[5] Libelium, «Waspmote,» [En línea]. Available: http://www.libelium.com/products/waspmote. [Último acceso: 2 Abril 2012].

[6] Libelium, «Waspmote Guía Técnica,» 2011. [En línea]. Available: http://www.libelium.com/documentation/waspmote/waspmote-technical_guide_esp.pdf. [Último acceso: 2 Abril 2012].

[7] Libelium, «ZigBee,» [En línea]. Available: http://www.libelium.com/products/waspmote/hardware#xbee. [Último acceso: 2 Abril 2012].

[8] B. Mitchell, «The MAC Address,» [En línea]. Available: http://compnetworking.about.com/od/networkprotocolsip/l/aa062202a.htm. [Último acceso: 2 Abril 2012].

[9] T. Spencer, «The Point-to-Point Protocol (PPP): An Overview,» [En línea]. Available: http://technet.microsoft.com/en-us/library/cc768082.aspx. [Último acceso: 5 Marzo 2012].

[10] W. Simpson, «The Point-to-Point Protocol (PPP),» 1994. [En línea]. Available: http://www.rfc-archive.org/getrfc.php?rfc=1661. [Último acceso: 12 Marzo 2012].

[11] Cisco Systems Inc. Internetworking Technologies Handbook, Cuarta ed., Cisco Press, 2003.

[12] A. Sun, Using and Managing PPP, Primera ed., O'Reilly & Associates, Inc., 1999.

[13] Cermetek Microelectronics. «AT Commands and S Registers.Reference Guide for CH1786, CH1794, CH1798 and CH1799,» [En línea]. Available: http://www.iclinks.com/public_ftp/DocRelease/icl4300/ModemATCmdRef.pdf. [Último acceso: 5 Marzo 2012].

[14] Autor desconocido «Basic Hayes Modem AT strings,» [En línea]. Available: http://www.computerhope.com/atcom.htm. [Último acceso: 5 Marzo 2012].

[15] UbiNetics, «GSM AT Command Set,» 2001. [En línea]. Available: http://www.zeeman.de/wp-content/uploads/2007/09/ubinetics-at-command-set.pdf. [Último acceso: 5 Marzo 2012].

[16] Autor desconocido. «Comandos modems GSM/GPRS,» 2006. [En línea]. Available: http://www.pcdemano.com/phpBB2/phpBBToGo/thread.php?topic_id=15344. [Último acceso: 6 Marzo 2012].

[17] Airlink Communications Inc. Redwing GPRS. User Guide, 2006.

[18] K. D. Singh, G. Rubino y C. Viho, « Chapter 5: Packet Scheduling Principles and Algorithms for Downlink,» de *HSDPA/HSUPA Handbook*, Taylor & Francis Group, 2011.

[19] TelecomSpace, «High Speed Downlink Packet Access (HSDPA),» [En línea]. Available: http://www.telecomspace.com/latesttrends-hsdpa.html. [Último acceso: 5 Marzo 2012].

[20] C. Johnson, Radio Access Networks for UMTS: Principles and Practices, John Wiley & Sons Ltd., 2008.

[21] H. Holma y A. Toskala, HSDPA HSUPA for UMTS, John Wiley & Sons, Ltd., 2006.

[22] D. J. Barrett y R. E. Silverman, SSH, the Secure Shell: The Definitive Guide, O'Reilly & Associates, Inc., 2001.

[23] etutorials.org, «PPP Connections,» [En línea]. Available: http://etutorials.org/Linux+systems/how+linux+works/Chapter+5+Configuring+Your+Network/5.8+PPP+Connections/. [Último acceso: 18 Marzo 2012].

[24] Linux Man Page. «Point-to-Point Protocol Daemon,» [En línea]. Available: http://linux.die.net/man/8/pppd. [Último acceso: 21 Marzo 2012].

[25] YoLinux, «Using PPP,» [En línea]. Available: http://www.yolinux.com/TUTORIALS/LinuxTutorialPPP.html. [Último acceso: 18 Marzo 2012].

[26] Autor Desconocido. «Introduction to NTP,» [En línea]. Available: http://www.akadia.com/services/ntp_synchronize.html. [Último acceso: 18 Abril 2012].

[27] Ubuntu Official Documentation. «Time Synchronisation with NTP,» [En línea]. Available: https://help.ubuntu.com/11.04/serverguide/NTP.html. [Último acceso: 18 Abril 2012].

[28] C. Liechti, «pySerial,» 2010. [En línea]. Available: http://pyserial.sourceforge.net/pyserial.html. [Último acceso: 27 Marzo 2012].

[29] M. Čihař, «python-gammu API,» 2011. [En línea]. Available: http://wammu.eu/docs/manual/python/. [Último acceso: 4 Abril 2012].

[30] J. Noller, «SSH Programming with Paramiko,» 2008. [En línea]. Available: http://jessenoller.com/2009/02/05/ssh-programming-with-paramiko-completely-different/. [Último acceso: 25 Abril 2012].

[31] M. Čihař, «Gammu,» 2012. [En línea]. Available: http://wammu.eu/gammu/. [Último acceso: 25 Abril 2012].

[32] M. Čihař, «The Gammu Manual,» 2012. [En línea]. Available: http://wammu.eu/docs/manual/. [Último acceso: 26 Abril 2012].

[33] elinux.org, «BeagleBoardUbuntu,» [En línea]. Available: http://elinux.org/BeagleBoardUbuntu. [Último acceso: 20 Febrero 2012].

[34] Ubuntu Official Documentation. «CronHowto,» 2011. [En línea]. Available: https://help.ubuntu.com/community/CronHowto. [Último acceso: 2 Mayo 2012].

[35] D. Drake, «Writing udev rules,» 2006. [En línea]. Available: http://www.reactivated.net/writing_udev_rules.html. [Último acceso: 25 Mayo 2012].

[36] Autor Desconocido. «How to change the timezone in linux,» 2007. [En línea]. Available: http://www.r71.nl/kb/technical/89-how-to-change-the-timezone-in-linux. [Último acceso: 30 Marzo 2012].

[37] Ubuntu ManPage Repository. «ubuntu Manuals-update-rc.d,» 2010. [En línea]. Available: http://manpages.ubuntu.com/manpages/hardy/es/man8/update-rc.d.8.html. [Último acceso: 12 Abril 2012].

[38] Ubuntu ManPage Repository. «ubuntu Manuals,» 2012. [En línea]. Available: http://manpages.ubuntu.com/manpages/hardy/man1/pon.1.html. [Último acceso: 14 Marzo 2012].

[39] Grupo ADSLZone. «Test de velocidad,» 2012. [En línea]. Available: http://www.testdevelocidad.es/. [Último acceso: 8 Junio 2012].

[40] Kolbi, «Internet por minuto,» [En línea]. Available: http://www.grupoice.com/wps/portal/!ut/p/c4/04_SB8K8xLLM9MSSzPy8xBz9CP0os3gjdz9Ho1BHX0cnQyc3A88AX6cARz8vIyNfA_2CbEdFALdA3L0!/. [Último acceso: 1 Junio 2012].

[41] Kolbi, «Planes y precios del internet móvil,» [En línea]. Available: http://www.grupoice.com/wps/portal/!ut/p/c5/ldBNDoIwEAXgs3iCDlModVn5LQgllBpkQ1gYQyLgwuj1JboG8c3y5ctMhjRknrF79tfu0U9jdyM1aViLoG1diUzQk7BBHjkPwA-QZ-7cn1lbJZaKPW6DChwKMnB8yhRDSJ2_dGQ0grSqogxdRXVBt2hYiIA1LdhXL_WIW3ZjlAs0c18abYHc85Cm8pB46Ta9fNuKNnRdm19fy-NpuJD. [Último acceso: 1 Junio 2012].

[42] Movistar, «Internet Móvil Prepago,» [En línea]. Available: http://movistar.cr/internet_movistar/planes-de-internet/prepago/. [Último acceso: 1 Junio 2012].

Apéndices

A.1. Glosario y abreviaturas

- Ancho de banda: cantidad de datos que se puede enviar a través de una conexión de red en un tiempo determinado,
- ARM Cortex – A8: Procesador del BeagleBoard xM
- Bash: lenguaje de programación para Linux
- BaudRate: número de unidades de señal por segundo.
- BeagleBoard: Tarjeta de desarrollo
- Datacard: Módem telefónico con conexión USB
- DNS: Domain Name Service
- Gateway: Módulo receptor de datos
- HSDPA: High-Speed Downlink Packet Access
- IEEE: Institute of Electrical and Electronics Engineers
- kbps: kilobits por segundo. Unidad de medición de la velocidad de transmisión de datos
- kB/s: kilobyte por segundo. Unidad de medición de la velocidad de transmisión de datos
- MAC: media access control, valor numérica característicos de cada dispositivo
- Nodo: un dispositivo modular Waspmote colocado en el puente
- NTP: Network Time Protocol
- PPP: Protocolo Punto a Punto
- Python: lenguaje de programación de alto nivel.
- RS-232: es una interfaz que designa una norma para el intercambio de una serie de datos binarios dos equipos
- SFTP: Secure File Transfer Protocol
- Scripts: archivo de ejecución de órdenes
- UART: Universal Asynchronous Receiver-Transmitter
- USB: Universal Serial Bus. Protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre ordenadores y dispositivos electrónicos
- Waspmote: dispositivo con arquitectura modular, la cual consiste en integrar únicamente los módulos que se requieren en cada dispositivo
- XBee: Marca electrónica de módulos de transmisiones inalámbricas
- ZigBee: Conjunto de protocolos de alto nivel de comunicación inalámbrica de bajo consumo, basada en el estándar IEEE 802.15.4

A.2. Script de Instalación en *Bash*

```
#!/bin/bash

apt-get update ##Actualiza la base de datos

apt-get upgrade ## Instala los paquetes mas nuevos

apt-get install ppp ##Instala la libreria para protocolo ppp en linux

apt-get install python-serial ##Libreria de python para lectura de puerto
 serial

apt-get install python-paramiko ##Libreria de python para SFTP

apt-get install python-gammu ## Libreria de python para usar el modulo
 datacard para envio y recepcion de mensaje

apt-get install gammu  ##Daemon de linux para envio y recepcion de mensaje

apt-get install comgt  ## Lectura de senal de la datacard

apt-get install sshpass  ##Pasa por parametro password al sftp , opcional

apt-get install minicom  ##Terminal , opcional


##Archivos de configuracion

cp gammurc /etc/gammurc    ##Gammu config file

##PPP SCRIPT

cp chatICE /etc/ppp/chatICE

cp chatICE /etc/ppp/chatMovistar

cp Movistar /etc/ppp/peers/Movistar

cp ICE /etc/ppp/peers/ICE

cp cronometro /etc/crontab     ##Scan every three minutes for incoming SMS

cp  75-tty-description.rules  /etc/udev/rules.d    ##  gateway  attached  to
 /dev/gateway   && datacard attached to /dev/datacard

cp InitSetDate.sh /etc/init.d/InitSetDate.sh   ## Set Date at boot


ln -s /usr/share/zoneinfo/America/Costa_Rica /etc/localtime.

update-rc.d InitSetDate.sh defaults 99


exit
```

Anexos

B.1. Especificaciones eléctricas de la tarjeta de desarrollo BeagleBoard [4]

Specification	Min	Typ	Max	Unit
Power				
Input Voltage USB		5	5.2	V
Current USB		350		mA
Input Voltage DC	4.8	5	5.2	V
Current DC		750		mA
Max Voltage without damage			12	V
Expansion Voltage (5V)	4.8	5	5.2	V
Curent (Dépends on source curent available)		750		A
Expansion Voltage (1.8V)	1.75	1.8	1.85	V
Current			30	mA
USB Host (Same as the DC supplied by the power plug or USB 5V)	4.8	5	5.2	V
Current (Depends on what the DC source can supply over what the board requires)		Varies		
Maximum current supplied by all four USB Host ports Total		1500		mA
USB OTG				
High Speed Mode			480	Mb/S
Full Speed Mode			12.5	Mb/S
Low Speed Mode			1.5	Mb/S
USB Host				
High Speed Mode			480	Mb/S
Full Speed Mode			12.5	Mb/S
Low Speed Mode			1.5	Mb/S
RS232				
Transmit				
High Level Output Voltage		5	5.4	V
Low Level output voltage		-5	-5.5	V
Output impedance		+/-35	+/-60	mA
Maximum data rate	250			Kbit/S
Receive				
High level Input Voltage	-2.7	-3.2		V
Lo Level Input Voltage			.4	
Input resistance	3	5	7	Kohms
JTAG				
Realview ICE Tool			30	MHz
XDS560			30	MHz
XDS510			30	MHz
Lauterbach(tm)			30	MHz
microSD				
Voltage Mode 1.8V	1.71	1.8	1.89	V
Voltage Mode 3.0V	2.7	3.0		V
Current			220	mA
Clock			48	MHz
DVI-D				
Pixel Clock Frequency	25		65	MHz

High level output voltage		3.3		V
Swing output voltage	400		600	mVp-p
Maximum resolution			1024 x 768	
S-Video				
Full scale output voltage (75ohm load)	.7	.88	1	V
Offset voltage		50		mV
Output Impedance	67.5	75	82.5	Ohms
Audio In				
Peak-to-peak single-ended input voltage (0 dBFs)			1.5	Vpp
Total harmonic distortion (sine wave @ 1.02 kHz @ -1 dBFs)		-80	-75	dB
Total harmonic distortion (sine wave @ 1.02 kHz) 2 0 Hz to 20 kHz, A-weighted audio, Gain = 0 dB		-85	-78	dB
Audio Out				
Load Impedance @100 pF	14	16		ohms
Maximum Output Power (At 0.53 Vrms differential output voltage and load impedance = 16 Ohms)		17.56		mW
Peak-to-Peak output voltage			1.5	Vpp
Total Harmonic Distortion @ 0 dBFs		-80	-75	dB
Idle channel noise (20Hz to 20KHz)		-90	-85	dB
Environmental				
Temperature range	0		+85	C

B.2. Hoja de datos del módulo Waspmote [6]

Waspmote

General data:

Microcontroller:	ATmega1281
Frequency:	8MHz
SRAM:	8KB
EEPROM:	4KB
FLASH:	128KB
SD Card:	2GB
Weight:	20gr
Dimensions:	73.5 x 51 x 13 mm
Temperature Range:	[-20ºC, +65ºC]
Clock:	RTC (32KHz)

Consumption:

ON:	9mA
Sleep:	62µA
Deep Sleep:	62µA
Hibernate:	0.7µA

Operation without recharging: 1 year *

* *Time obtained using the Hibernate mode as the energy saving mode*

Inputs/Outputs:

7 Analog (I), 8 Digital (I/O), 1 PWM,
2 UART, 1 I2C, 1 USB

Electrical data:

Battery voltage:	3.3 V - 4.2V
USB charging:	5 V - 100mA
Solar panel charging:	6 - 12 V - 280mA
Auxiliary battery voltage:	3V

Built-in sensors on the board:

Temperature (+/-): -40ºC, +85ºC. Accuracy: 0.25ºC
Accelerometer: ±2g (1024 LSb/g) / ±6g (340LSb/g)
40Hz/160Hz/640Hz/2560Hz

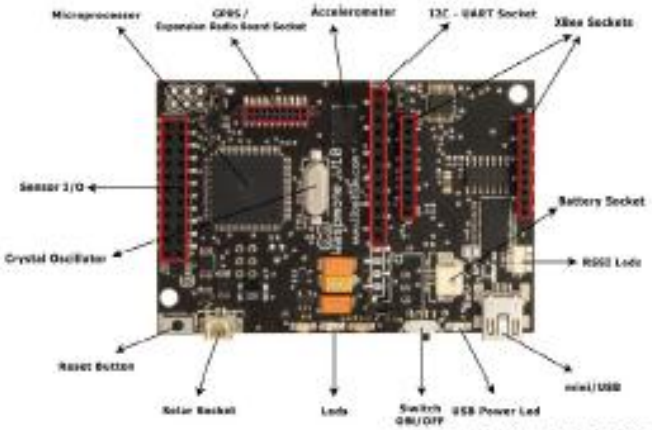


Figure 1: Waspmote Board Top

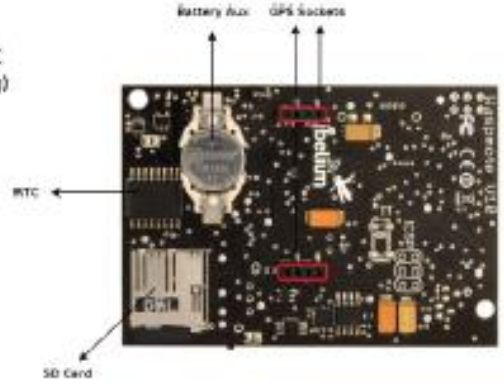


Figure 2: Waspmote Board Bottom

B.3. Hoja de datos del módulo 802.15.4/ZigBee [6]

802.15.4/ZigBee

Model	Protocol	Frequency	txPower	Sensitivity	Range *
XBee-802.15.4	802.15.4	2.4GHz	1mW	-92dB	500m
XBee-802.15.4-Pro	802.15.4	2.4GHz	100mW	-100dBm	7000m
XBee-ZB	ZigBee-Pro	2.4GHz	2mW	-96dBm	500m
XBee-ZB-Pro	ZigBee-Pro	2.4GHz	50mW	-102dBm	7000m
XBee-868	RF	868MHz	315mW	-112dBm	12km
XBee-900	RF	900MHz	50mW	-100dBm	10km
XBee-XSC	RF	900MHz	100mW	-106dBm	12km

Figure 3: XBee

*Line of sight and 5dBi dipole antenna

Antennas: 2.4GHz: 2dBi / 5dBi
868/900MHz: 0dBi / 4.5dBi
Connector: RPSMA
Encryption: AES 128b
Control Signal: RSSI
Standards: XBee-802.15.4 - 802.15.4 Compliant / XBee-ZB - ZigBee-Pro v2007 Compliant
Topologies: p2p, tree, mesh

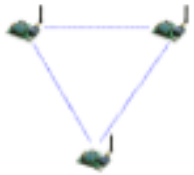


Figure 4: p2p

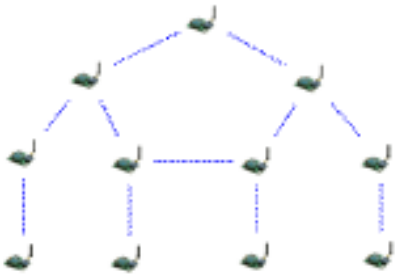


Figure 5: tree

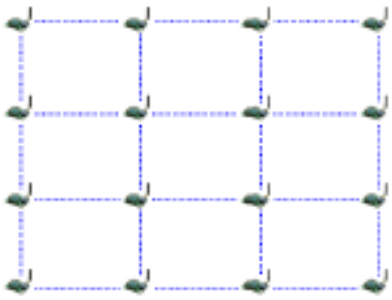


Figure 6: mesh

B.4. Hoja de datos del módulo Gateway [6]

Power supplies

- 1150mA/2300mA/6600mA Li-Ion rechargeable // 13000mAH **non - rechargeable**
- Solar Panel: rigid (7V – 500mA) and flexible (7.2V – 100mA)
- USB (220V-USB, car lighter USB)

USB-PC interface

Model: Waspmote Gateway *
Communication: 802.15.4/ZigBee - USB PC
Programmable buttons and leds
** Included in the developers Kit*

Compiler:

- IDE-Waspmote (open source)
- Language: C++
- Versions Windows, Linux and Mac-OS

Figure 26: Waspmote Gateway

Certifications

- CE (Europe)
- FCC (USA)
- IC (Canada)