eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Proyecto de Fin de Carrera

INGENIERÍA INFORMÁTICA

Aplicación para una red WSN basada en ContikiOS

Autor
Endika Ibarzabal Euba

informatika fakultatea
facultad de informática

Marzo de 2014

Índice general

Resumen del proyecto

Durante los últimos años hemos venido observando la tendencia a incorporar capacidad de procesamiento y comunicación a dispositivos que hasta entonces se utilizaban de modo independiente. La evolución de los móviles a *smartphones* es un claro ejemplo de dicha tendencia, aunque también cabe mencionar otros ejemplos, como es el caso de los denominados hogares inteligentes, en los que elementos del hogar se encuentran interconectados y pueden actuar de modo conjunto. Esta tendencia no se limita a sistemas independientes, sino que propone interconectar todos los elementos disponibles para conformar la denominada Internet de los Objetos/Cosas o *Internet of Things*, IoT.

Una de las mayores dificultades que se presenta en estos sistemas es que las características de estos nuevos dispositivos *inteligentes*, en general limitados en términos de cómputo, almacenamiento, autonomía o comunicación, queda a menudo lejos de los equipos informáticos tradicionales. Esta cuestión, junto con la ausencia de estándares para gestionar estos dispositivos, constituye un importante problema a abordar.

Considerando este marco, en este proyecto se ha desarrollado una aplicación orientada a este tipo de dispositivos. Más concretamente, la aplicación tiene como soporte una red de sensores inalámbricos, WSN, con el objetivo de realizar seguimiento de individuos.

Cabe destacar que el desarrollo de la aplicación se ha realizado utilizando Contiki OS, sistema operativo diseñado especialmente para dispositivos con características limitadas como los presentados anteriormente y firme candidato a convertirse en estándar.

Índice de figuras

Indice de tablas

1. CAPÍTULO

Introducción

Esta memoria contiene la documentación del Proyecto de Fin de Carrera realizado por Endika Ibarzabal Euba para optar al grado de Ingeniería Informática en la facultad de informática en San Sebastián de la UPV/EHU.

El proyecto se plantea dentro de una de las líneas de trabajo del Grupo de Sistemas Distribuidos del UPV/EHU (en la Facultad de Informática). En esta línea de trabajo se desarrolla la implementación de un sistema integral de seguimiento de rutas realizadas por individuos por medio de dispositivos autónomos de costo reducido (frente a otras opciones más costosas).

Más concretamente, el tipo de individuo al que se debe realizar el seguimiento es una especie de ave. Esta característica tiene una gran relevancia en el sistema, dado que los dispositivos que se deben incorporar a las aves para realizar el seguimiento deben cumplir unos requisitos mínimos muy exigentes en cuanto a resistencia, autonomía, capacidad de comunicación...

La opción principal considerada para implementar los dispositivos de seguimiento ha sido mediante una red de sensores inalámbricos. Esta aproximación requiere (1) adaptar el hardware de las motas que componen la red para cumplir los requisitos mínimos previamente mencionados y (2) programar las motas para ajustar su comportamiento al establecido por la aplicación (recogida de datos sobre posicionamiento, gestión de los distintos módulos...).

En un proyecto previo realizado en el mismo grupo de trabajo [10] se establecieron los primeros pasos del proyecto. Dicho trabajo se centró es aspectos relacionados con la viabilidad técnica del proyecto relativa a aspectos hardware, si bien también se planteó un diseño inicial de la aplicación que gestionaría dichos dispositivos.

Este proyecto tiene dos objetivos principales. En primer lugar el desarrollo de un prototipo que implemente las funcionalidades mínimas de la red de sensores (recolección de información sobre posicionamiento y gestión de los módulos para la optimización de la batería). El segundo objetivo principal consiste en la evaluación del sistema operativo Contiki como plataforma de desarrollo para las motas que forman la red.

Cabe destacar que, dada la limitación en horas del proyecto, no se considera inicialmente la prueba de la aplicación en hardware real, de tal modo la aplicación será validada tan sólo a nivel de simulación. Se contempla que tras este proyecto se realicen las pruebas correspondientes en hardware real y que, después de este análisis de factibilidad, se realice un análisis comparativo frente a otras opciones tales como TinyOS [7].

Organización del documento. La memoria del proyecto está estructurada del siguiente modo. En el capítulo *Documento de Objetivos del Proyecto* se presentan los objetivos generales del proyecto, junto con la descripción de las tareas correspondientes y su correspondiente planificación. Posteriormente se detalla la metodología a utilizar y el análisis de riesgos del proyecto. En el siguiente apartado, *Desarrollo Técnico*, se describen los pasos realizados durante el desarrollo del proyecto. Por último en el capítulo *Resultados, Conclusiones y Líneas Futuras* se evalúan los resultados obtenidos y se extraen las conclusiones del proyecto, incluyendo una breve descripción de las posibles líneas de trabajo futuras.

2. CAPÍTULO

Documento de Objetivos del Proyecto

2.1. Introducción

Tras la presentación en el capítulo anterior de los objetivos generales del proyecto, en este apartado se analizarán en más detalle y se las tareas necesarias para alcanzar dichos objetivos y se planificará su realización.

Cabe destacar que este proyecto forma parte de otro con un alcance mayor. Por lo tanto, es importante distinguir qué tareas se realizarán en este proyecto y cuales quedan como trabajo futuro.

Los objetivos de este proyecto se clasifican en dos tipos. Por una parte se consideran los prioritarios y por otra los de segundo orden.

Como objetivo prioritario se encuentra implementar una aplicación sobre Contiki que permita que las motas recojan datos sobre su posición a medida que se mueven y, periódicamente descarguen la información recolectada en un punto centralizado denominado estación base o *sink*.

En términos generales, definiremos como objetivos secundarios todas aquellas tareas que deberían ser ofrecidas por la versión final de la aplicación, pero que no resultan imprescindibles para el prototipo con el objetivo planteado en el punto anterior.

Organización del capítulo En el siguiente apartado, *Objetivos del proyecto* se describirán los objetivos del proyecto. A continuación se presentará la arquitectura del sistema (*Arquitectura del Sistema*). Tras lo cuál se establecerán las tareas necesarias y su correspondiente planificación *Planificación del Proyecto*. Por último, se hará un análisis de riesgos y se tratarán la disponibilidad de los recursos necesarios para el proyecto.

2.2. Objetivos del proyecto

Como se ha presentado anteriormente, el objetivo prioritario será crear un prototipo de una aplicación sobre Contiki que permita que las motas recojan datos sobre su posición a medida que se

mueven. Cuando las motas regresen a un punto centralizado, denominado estación base o *sink*, deberán descargar la información recolectada, de tal modo que (1) el espacio correspondiente pueda ser liberado para almacenar nueva información y (2) en la estación base queda acumulada la información de todas las motas para su posterior explotación (esta última parte queda fuera del alcance del proyecto).

Como ejemplo de aplicación se plantea un sistema de monitorización de rutas seguidas por varios ejemplares de gaviota. Cada uno de los ejemplares monitorizados lleva incorporada una mota que periodicamente registra la posición del ejemplar y la almacena. Con cierta regularidad, las aves regresan a su colonia, lugar ubicado en un punto que denominamos *nido* y en el que se encuentra una estación base o *sink* (en términos de WSN). Este último nodo permitirá recabar la información de las motas monitorizadas y enviarla a otro sistema para que pueda ser explotada.

Para alcanzar el objetivo planteado, se definen los siguientes objetivos de segundo nivel:

1. Diseño de la aplicación.
    a) Diseño de una aplicación para redes de sensores que recaba datos.
    b) Comunicación y utilización de módulos en función de la energía disponible, alcance ...
    c) Gestión eficiente de los datos almacenados
2. Formación en las aspectos concretos del área
    a) En el ámbito de redes de sensores: características asociadas a este tipo de redes
    b) En el Sistema Operativo Contiki
        1) Estudiar las distintas posibilidades para la simulación de aplicaciones
        2) Estudiar cómo implementar las diferentes características del sistema a simular: principalmente la movilidad de las motas, el módulo de posicionamiento incorporado y módulos de comunicación.

2.3. Arquitectura del Sistema

Según se ha descrito en el apartado de Objetivos del proyecto, el proyecto proponer implementar la solución mediante una red de sensores en la que se distinguen dos roles principales con un comportamiento distinto:

1. Motas en movimiento
2. Estación base: se trata de una mota situada en un punto estático y conectada a un equipo adicional que le permite tener mayor capacidad en términos de energía y capacidad de almacenamiento.

Dado que cada rol tiene asociado un comportamiento diferente, para cada uno de los dos roles se desarrollará el módulo correspondiente.

2.3.1. Sistema operativo y plataforma de desarrollo

El sistema operativo elegido para desarrollar el proyecto se denomina Contiki OS. Este sistema operativo basado en Linux, ha sido diseñado para utilizarla en redes de sensores inalámbricas. No siendo el único de este tipo como anteriormente se ha comentado.

Los desarrolladores de ContikiOS ofrecen diversas opciones para los desarrolladores, entre ellas, la más simple, y la utilizada en nuestro caso, se trata de su simulación por medio de una máquina virtual. En esta máquina virtual, podremos encontrar una versión simplificada del sistema operativo Ubuntu, en la cual, se encuentra instalado disponible ContikiOS junto con herramientas de desarrollo, siendo las más destacable el simulador denominado Cooja.

En el simulador Cooja podremos encontrar diversas herramientas y aplicaciones instaladas listas para su utilización, tales como, la herramienta PowerTrace de la cual se hablará posteriormente. Por otra parte, a las herramientas ya instaladas, se pueden añadir otras. En nuestro caso Mobility, de la cual hablaremos más adelante. A parte de estas herramientas, podremos encontrar en la carpeta `example` de Cooja, un conjunto de ejemplos para hacernos una idea de las utilidades y funcionamientos de ContikiOS. Cabe destacar que estos ejemplos son muy útiles para los principiantes. Sin ir más lejos se considera utilizar como referencia algunos de estos ejemplos como base para el desarrollo de este proyecto.

2.4. Planificación del Proyecto

A grandes rasgos se estima que la implementación supondrá el mayor esfuerzo del proyecto, con un consumo del 57 % del tiempo total. Por otra parte en la gestión, se prevé el 27 % de esfuerzo. Dentro de ésta se encuentran: la planificación, el seguimiento y control, las reuniones, la documentación y tiempo para la presentación. El 16 % de tiempo restante, se divide en la formación, diseño, pruebas y un tiempo reservado para posibles incidencias.

2.4.1. Objetivos prioritarios

Tal y como se presentó en el apartado Objetivos del proyecto, se define un conjunto de tareas mínimas que se han de llevar a cabo con prioridad.

- Simular el estado de la batería
- Gestión del módulo GNSS de posicionamiento (GPS)
  - Función para activar/desactivar módulo
  - Función consulta estado módulo (activado/desactiva/error)
  - Función obtener datos (posición + hora)
- Gestión del módulo de radio

    - Función para activar/desactivar módulo
    - Función consulta estado módulo (activado/desactiva/error)
- Gestión de estados: siguiendo el diseño propuesto en [10], se consideran los siguientes estados:
    - Estado 1: Dentro del alcance de la ES
    - Estado 2: Fuera del alcance de la ES
    - Estado 3: Saliendo del radio de la ES (transición de Estado 2 a Estado 1)
    - Estado 4: Entrando en el radio de la ES (transición de Estado 1 a Estado 2)

Restricciones y limitaciones

Las restricciones que sufrirá el proyecto son los siguientes:

- El presupuesto en horas del proyecto es de 375-400 horas. Por lo tanto, se deberá ajustar el alcance del proyecto a este presupuesto.

Ampliación del proyecto

Como se ha indicado antes, puede darse el caso de que se realicen unas tareas extra. El procedimiento para integrarlas en el proyecto es la siguiente:

1. Estudiar las ventajas y desventajas que supondrá la ampliación para confirmar su viabilidad.
2. Prever los cambios que sufrirá el proyecto al añadir la ampliación.
3. Una vez identificada la ampliación, decidir si se llevará a cabo. Para ello, una de las cosas a tener en cuenta será que el tiempo necesario será menor al 5 % del tiempo total del proyecto. Siempre y todo, la ultima palabra la tendrá el director del proyecto.
4. Llevar a cabo la ampliación.

Estos son las objetivos de segundo orden que se proponen:

- Gestión de estados
    - Estado 5 (Modo seguro): En este estado el Sink podría mandar actualizaciones de software a las motas.
- Incorporación de un nuevo módulo para gestionar la carga dinámica por medio de un panel solar. En este caso se podría considerar un cambio de comportamiento dependiendo de la carga de la batería para optimizar el rendimiento de esta última (en su tiempo de vida principalmente).

- Depurar aspectos relacionados con el consumo de los módulos de comunicaciones (GNSS/-radio). En este proyecto se manejarán cantidad absolutas. Sin embargo, se considera la posibilidad de que los consumos puedan variar. Como ejemplo, el GPS puede tener consumos diferentes en función de si se encuentra encendido y obteniendo coordenados, en standby, apagado o incluso al hacer el cambio entre los estados anteriores.

2.4.2. Estructura de Descomposición del Trabajo

La Estructura de Descomposición del Trabajo o EDT, agrupa y define el trabajo que se realizará en el proyecto. Las tareas que no se reflejan aquí no serán tomadas como parte del proyecto.

A continuación se presenta la lista de tareas necesarias para alcanzar los objetivos:

- F1-Formación: La formación necesaria para desarrollar el proyecto, se reparte en dos:
  - F1.1-ContikiOS: El sistema operativo en el que se basa la aplicaciones que se implementara.
  - F1.2-C: El lenguaje de programación que se utilizará para desarrollar el proyecto.
- O1-Obtener y preparar el entorno de desarrollo: El trabajo necesario para conseguir el entorno de desarrollo y simulación.
- D1-Diseño: Grupo de tareas necesarias para llevar a cabo el diseño de la aplicación, por ejemplo, prototipo y diseño de la interfaz gráfica.
- I1-Implementación: Partes que habrá que implementar para llevar a cabo el proyecto. Tales como, la gestión de los módulos GNSS y Radio, la simulación del movimiento de los nodos, del consumo de batería y la implementación de estados.
- G1-Gestión: Procesos tácticos del proyecto.
  - G1.1-Planificación: Definir los pasos a seguir para llevar a cabo el proyecto.
  - G1.2-Seguimiento y control: Seguimiento del proyecto.
  - G1.3-Reuniones: Reuniones realizados con el director del proyecto.
  - G1.4-Documentación: Documentación recogiendo todo lo trabajado en el proyecto.
  - G1.5-Presentación: Preparación de la presentación ante el jurado.
- C1-Incidencias: Tiempo extra guardado para incidencias que pueden suceder durante el proyecto. Si alguna de las tareas se alarga, se tomará tiempo de aquí.

La siguiente figura presenta el diagrama EDT.

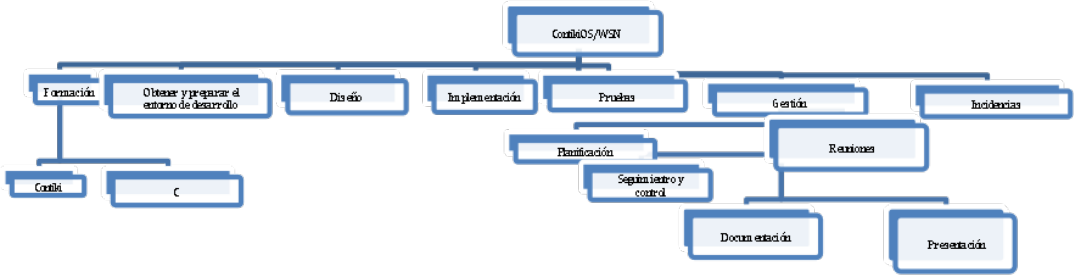


Figura 2.1. Estructura de Descomposión del Trabajo

2.4.3. Planificación

Metodología y Herramientas a utilizar

En esta parte, se analizará el ámbito de la planificación: explicación de las sub-tareas del proyecto, tiempo necesario para llevarlos a cabo, control de calidad, recursos y para finalizar los riesgos y soluciones previstas para tales.

Tareas a realizar

En la siguiente tabla, se presenta el tiempo estimado para realizar cada tarea.

Tabla 2.1. Estimación de las tareas del proyecto

Tarea	Horas	Porcentaje
Formación	30	8,0 %
-> Contiki	10	2,6 %
-> C	20	5,3 %
Obtener y preparar entorno de desarrollo	2	0,53 %
Diseño	5	1,3 %
Implementación	215	57,3 %
-> Simulación batería	45	12,0 %
-> Gestión del módulo GNSS	32	8,5 %
-> Gestión del módulo de radio	32	8,5 %
-> Gestión de estados	106	28,3 %
–> Estado 1	40	10,6 %
–> Estado 2	40	10,6 %
–> Estado 3	13	3,5 %
–> Estado 4	13	10,5 %
Pruebas	15	4,0 %
Gestión	102	27,2 %
-> Planificación	2	0,5 %
-> Seguimiento y control	2	0,5 %
-> Reuniones	8	2,1 %
-> Documentación	80	21,3 %
-> Presentación	10	2,6 %
Incidencias	6	1,6 %
TOTAL	**375**	**100,0 %**

La formación tiene asignado un esfuerzo considerable en el proyecto. Esto se debe a que antes de empezar este proyecto no se tiene ningún tipo de conocimiento sobre ContikiOS y redes de sensores inalámbricos. Aparte de eso, la aplicación se implementará en lenguaje C. Al no tener muchas horas de programación en este lenguaje, se prevé la necesidad de refrescar las bases de este lenguaje y estudiar aspectos específicos necesarios para este proyecto.

No se prevé mucho esfuerzo a la hora de obtener y preparar el entorno de desarrollo ni en el diseño. Este se debe a que, por una parte el entorno de desarrollo lo ofrece la comunidad de ContikiOS en forma de maquina virtual y que en principio, la aplicación tendrá un diseño simple.

Como se puede observar en la tabla la implementación tiene el mayor peso dentro del proyecto, esto se debe a que la finalidad de este trabajo es implementar una aplicación.

La otra tarea con más peso es la gestión, dentro de esta, la documentación es la que más importancia se le da. Esto se debe a que este trabajo se desarrolla como trabajo de fin de carrera, con el objetivo de presentarlo ante un jurado. El documento es el que se presenta y en el que se refleja todo el trabajo de esas 375 horas.

Cronograma

En este apartado, se presentarán las diferentes tareas en forma de diagrama de Gantt. Con este diagrama lo que se da a ver es el tiempo que se le dedicará a cada una de ellas y las dependencias que tienen.

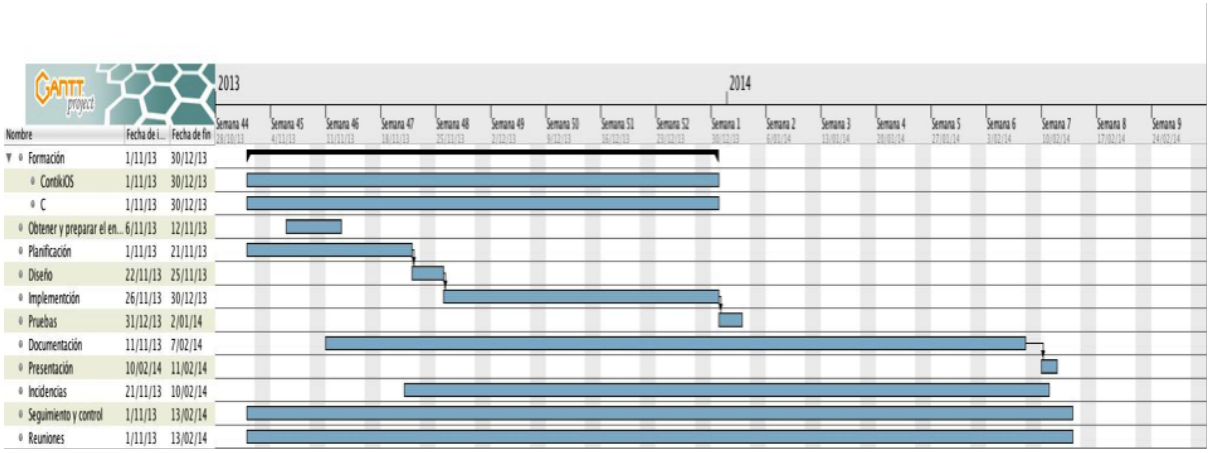


Figura 2.2. Diagrama GANTT

Como se puede ver en el diagrama, hay tareas que se realizarán durante todo el proyecto, esto es, de principio o fin:

- Seguimiento y control.
- Reuniones con el director.

Aparte de estas, las tareas que habrá que llevar a cabo al principio son, la formación y la planificación. Ya que no se tiene conocimientos sobre el tema que se va a desarrollar en el proyecto, se ve prioritario tener un mínimo conocimiento para poder realizar una planificación precisa. Una vez terminada la planificación y la obtención del entorno de programación, se empezará a diseñar. Hasta no tener un diseño no se comenzará la implementación.

Plan de calidad

En este apartado veremos que detalles habrá que tener en cuenta desde el punto de vista de la calidad. Se repartirá en dos partes principales, en el del producto y en el de los entregables.

Calidad del producto

Para asegurar la calidad del producto los elementos que se tendrán en cuenta serán los siguientes, los que se indican en color azul, indican la mínima calidad a conseguir, en color gris, las mejoras.

Tabla 2.2. Módulos principales de la aplicación

ID	Producto	Evaluación	Descripción
C1	Simulación batería	Capacidad y consumo de batería	▪ Datos simulados de consumo ▪ Datos simulados y reales de consumo
C2	Módulo GNSS	Posición y gestion	▪ Dar posición exacta ▪ Avisar de errores
C3	Módulo Radio	Posición y comunicaciones	▪ Comunicación sin errores ▪ Comunicación con menor consumo posible
C4	Gestión de estados	Estados implementados	▪ Cuatro estados principales ▪ Quinto estado

Calidad del proyecto y los entregables

En la siguiente tabla se indican las medidas a tomar sobre la calidad del proyecto y de los entregables.

Tabla 2.3. Indicativos de calidad de los entregables

ID	Producto	Evaluación	Descripción
C3	Entregables	▪ Formato ▪ Ortografía	▪ Portada, índice, numero de página, ▪ Sin faltas de ortografía graves.
C4	Control de las tereas	Plazos	▪ Terminar para la fecha indicada ▪ Terminar para dos días antes de la fecha indicada

2.5. Análisis de Riesgos

Antes de analizar los riesgos, se ha utilizado este diagrama para clasificar los distintos tipos de riesgo.

Tabla 2.4. Tipos de riesgo

Tipo de riesgo	Riesgo
Técnico	Tecnologías
Externo	Director de proyecto
Interno	▪ Trabajador ▪ Recursos
Dirección	▪ Estimación ▪ Planificación ▪ Comunicación

En la siguiente tabla, se resumen los riesgos que se describirán a continuación.

Tabla 2.5. Riesgos contemplados

ID	Riesgo	Impacto	Probabilidad	Influencia	Categoría
R1	Entorno de trabajo desconocido	Medio	60 %	Negativa	Técnico
R2	Estimación errónea	Alto	60 %	Negativa/ Positiva	Dirección
R3	Planificación errónea	Crítico	40 %	Negativa	Dirección
R4	Problemas de comunicación	Alto	40 %	Negativa	Externo
R5	Baja definida	Alto	60 %	Negativa	Interno
R6	Baja indefinidaa	Crítico	30 %	Negativa	Interno
R7	Fallar a una reunión	Bajo	30 %	Negativa	Externo /Interno
R8	Pérdida de información	Alto	40 %/20 %	Negativa	Interno
R9	Desconocimiento de las tecnologías	Medio	60 %	Negativa	Técnico

A continuación, se detallan todos los riesgos que se han previsto para este proyecto. En cada riesgo se han detallado las siguientes partes.

- Descripción: Breve resumen del riesgo.
- Impacto: Crítico, Alto, Medio, Bajo.
- Probabilidad: La probabilidad de que suceda.
- Plan de prevención: Medidas que se tomarán para que un riesgo no suceda.
- Plan de contingencia: Medidas que se tomarán una vez el riesgo suceda.
- Consecuencias: Las consecuencias de un riesgo.

- Influencia: Que tipo de influencia será, negativa o positiva.
- Categoría: En que categoría del gráfico antes descrito se encuentra.

2.5.1. R1. Entorno de trabajo desconocido

Descripción: A la hora de llevar a cabo una de las tareas, no tener los conocimientos suficientes. Tanto en la materia o herramientas necesarias.

Impacto: Medio.

Probabilidad: 60 %, por mucho que ya se haya trabajado anteriormente en otros proyectos, Contiki es una nueva tecnología.

Plan de prevención: De antemano se preverá el entorno en el que se trabajará y se les dará un tiempo de estudio adecuado.

Plan de contingencia: Dirigirse a algún entendido en esa materia para pedir ayuda.

Consecuencias: Necesitar más tiempo de lo debido en terminar una tarea. En consecuencia retrasar esa tarea y las que dependan de ella. En el peor de los casos, no poder llevar a cabo la tarea.

Influencia: Negativa.

Categoría: Riesgo técnico, tecnologías.

2.5.2. R2. Estimación errónea

Descripción: La estimación que se haya echo no sea exacta, esto es, que no se ciña a la realidad.

Impacto: Alto.

Probabilidad: 60 % Al no tener mucha experiencia en gestión de proyectos, es fácil desviarse en alguna de las tareas.

Plan de prevención: Tomar como ejemplo proyectos anteriores.

Plan de contingencia: Tener un tiempo invertido en incidencias.

Consecuencias: Tener que dejar de lado alguna tare por falta de tiempo.

Influencia: Negativa en el caso de que se haya estimado menos tiempo de lo necesario. Positiva, si el tiempo que se ha estimado es mayor que la que se ha necesitado.

Categoría: Riesgos en dirección, estimación.

2.5.3. R3. Planificación errónea

Descripción: La planificación realizada no se ciñe a la realidad.

Impacto: Crítico. En caso de que esto suceda, será necesario volver a hacer una planificación nueva.

Probabilidad: 40 %. Después de llevar a cabo unas cuantas planificaciones no se toma como un riesgo muy probable.

Plan de prevención: Una vez realizada la planificación. Pedir opinión a una persona con mas experiencia. En este caso al director del proyecto.

Plan de contingencia: Planificar de nuevo.

Consecuencias: Necesidad de planificar de nuevo el proyecto, esto conllevaría un gran esfuerzo.

Influencia: Negativa.

Categoría: Riesgo de dirección, planificación.

2.5.4. R4. Problemas de comunicación

Descripción: Problemas de comunicación con el director del proyecto.

Impacto: Alto.

Probabilidad: 40 %, por problemas tanto tecnológicas como humanas puede ser que suceda.

Plan de prevención: Asegurarse de mandar correctamente los mensajes barias veces.

Plan de contingencia: Si se repiten varias veces por problemas tecnológicos, buscar otra tecnología para la comunicación. Si el problema es humano, reunirse para buscar una solución.

Consecuencias: Mal entendidos entre el director de proyecto y el alumno.

Influencia: Negativo.

Categoría: Riesgo externo, director de proyecto. Riesgo interno, trabajador.

2.5.5. R5. Baja definida

Descripción: Por problemas externos, ya sean: de salud, personales, . . . El estudiante no puede seguir con el trabajo por un tiempo definido.

Impacto: Alto.

Probabilidad: 60 %.

Plan de prevención: Se prevé un tiempo para imprevistos. En el caso de que sea una baja de la que se tiene constancia con antelación (ej: Cita con el doctor), redistribuir las horas que se perderán.

Plan de contingencia: Informar al director del proyecto. Redistribuir el tiempo perdido.

Consecuencias: Aumentar la carga de trabajo para el resto de días. En el peor caso, retrasar las fechas de entrega.

Influencia: Negativa.

Categoría: Riesgo Interno, trabajador.

2.5.6. R6. Baja indefinida

Descripción: Por problemas externos, ya sean: de salud, personales, etc. El estudiante debe de abandonar el proyecto.

Impacto: Crítico.

Probabilidad: 30 %.

Plan de prevención:

Plan de contingencia: Avisar al director del proyecto.

Consecuencias: El proyecto se abandona.

Influencia: Negativa.

Categoría: Riesgo Interno, trabajador.

2.5.7. R7. Fallar a una reunión

Descripción: El estudiante o el director de proyecto no han podido acudir a reunión con cita previa.

Impacto: Bajo.

Probabilidad: 30 %

Plan de prevención: Hacer reuniones solo cuando sea necesario (si las reuniones son muy habituales es normal que alguna vez haya fallos). Convocatoria en fechas y horas que sean adecuadas para todos los integrantes.

Plan de contingencia: La persona que se haya presentado en la reunión, informará al otro de su falta, y se volverá a convocar otra reunión con los mismos puntos en la brevedad posible. Esto lo ara el que se haya presentado, ya que no se sabe cual es el motivo de ausencia del otro miembro.

Consecuencias: Si la reunión era necesaria para proseguir con el proyecto, este se retrasara.

Influencia: Negativa.

Categoría: Riesgo externo, director de proyecto. Riesgo interno, trabajador.

2.5.8. R8. Pérdida de información

Descripción: pérdida de información relacionada con el proyecto. Esta pérdida puede ser personal (PC, disco duro...) tanto en la nube (Dropbox).

Impacto: Alto.

Probabilidad: 40 % personal y 20 % nube la tecnología de Dropbox es fiable.

Plan de prevención: Aparte de la copia en el portátil personal. Se tendrá una copia diariamente actualizada en la nube (Dropbox) y una copia semanalmente actualizada en un disco duro externo.

Plan de contingencia: En el caso de la pérdida de información en el portátil/ disco duro, se hará una copia inmediata desde la nube. En el caso de que se pierda la información de la nube y la Dopbox siga funcionando, hace una nueva copia en Dopbox. Si la nube ha dejado de funcionar, se hará una copia en otra servidor de la nube (SkyDrive).

Consecuencias: En caso de que la pérdida sea al final del día, se perderá el trabajo de ese día. En caso contrario, no habrá consecuencias.

Influencia: Negativa.

Categoría: Riesgo Interno, recursos.

2.5.9. R9. Desconocimiento de las tecnologías a utilizar

Descripción: A la hora de realizar el proyecto, no tener conocimiento de alguna de las tecnologías.

Impacto: Medio.

Probabilidad: 60 %, por mucho que ya se han llevado a cabo este tipo de proyectos, Contiki es una nueva tecnología.

Plan de prevención: De ante mano se preverán las tecnologías necesarias y se les dará un tiempo de estudio adecuado.

Plan de contingencia: Dirigirse a algún entendido en esa materia para pedir ayuda.

Consecuencias: Necesitar más tiempo de lo debido en terminar una tarea. En consecuencia retrasar esa tarea y las que dependan de ella. En el peor de los casos, no poder llevar a cabo la tarea.

Influencia: Negativa.

Categoría: Riesgo técnico, tecnologías.

2.6. Recursos

Se distinguen dos tipos distintos de recursos. Los necesarios para desarrollar la aplicación y los necesarios para la documentación del proyecto.

2.6.1. Entorno de programación y simulación

Para la implementación y la simulación de la aplicaciones se usará el Sistema Operativo Contiki y su herramienta de simulación Cooja.

Desde el propio sitio web de Contiki OS se ofrece ya preparada y de modo gratuito una máquina virtual con el sistema operativo Contiki disponible y el simulador Cooja configurado[#].

La máquina virtual es de tipo VMWare, software que también se encuentra disponible de modo gratuito.

Como se ha señalado anteriormente, no se considera una implantación real de la aplicación, por lo que no será necesario hardware adicional, además del equipo del alumno.

2.6.2. Documentación de anteriores proyectos

De cara a la estructura de la memoria del proyecto se tomará como modelo de proyecto de fin de carrera proyectos desarrollados anteriormente por el autor durante la carrera, así como otros ya incluidos en la bibliografía.

3. CAPÍTULO

Desarrollo Técnico

3.1. Introducción

En este apartado se presentan los distintos módulos que componen la aplicación desarrollada. El desarrollo se centra principalmente en la gestión de los módulos de radio, GPS y batería.

Para el módulo de radio, detallaremos el protocolo de comunicación utilizada, ContikiMac, protocolo diseñado para redes de dispositivos de baja potencia.

En cuanto a la simulación del GPS, la principal herramienta utilizada ha sido el simulador de script de Cooja.

En el caso del consumo de batería, explicaremos qué es PowerTrace y como se utiliza, además de alguna mejora realizada para la visualización de datos.

Por último mostraremos los demás módulos utilizados y las pruebas realizadas para comprobar el correcto funcionamiento de nuestra aplicación. Además de el entorno utilizado para realizar dichas pruebas.

3.2. Análisis

El problema al que tenemos que dar solución es el de almacenar la posición de las distintas motas para más adelante transmitirlas a una base. Para ello utilizaremos motas que se comunican mediante radio con una estación base, llamado Sink. Los principales inconvenientes de estas motas son, el corto alcance del módulo de radio, la escasa capacidad de almacenamiento, junto con la escasa autonomía en términos de energía/batería.

La gestión de energía es a la que más importancia hemos dado, por lo que nuestro diseño se centrará en optimizar el consumo de la batería. Recordemos que, como se ha comentado anteriormente, estas motas no se podrán manipular tras ser colocadas, por lo que no es factible reemplazar la batería

periodicamente. Esto implica que es vital para la operatividad de la mota que el tiempo de vida de la batería sea el máximo posible.

En cuanto al problema con el almacenamiento, se tendrá que almacenar solo los datos necesarios y de la forma más eficiente posible.

Se tendrá que tener en cuenta, que siendo la estación base única y pudiendo haber una gran cantidad de nodos, la colisión en las comunicaciones será posible. Con lo cual, el protocolo de comunicación tendrá que controlar este tipo de errores.

Por último, las herramientas disponibles para la localización serán: el módulo GPS, que estará disponible en cualquier posición pero con un consumo mayor de batería, y el módulo de radio, que tendrá un menor consumo de batería, pero solo será utilizable dentro del alcance de radio del Sink.

3.2.1. Software del dispositivo de seguimiento

El software de seguimiento gestiona el uso de los distintos componentes que incorpora el dispositivo. Entre sus principales objetivos se encuentra minimizar el uso de los dispositivos con mayor consumo de energía.

Para el control de los componentes de las motas, se han diseñado distintos estados. En cada estado el nodo utilizará el módulo necesario para que el consumo sea lo menor posible. Así pues, cuando el dispositivo se encuentre cerca del nido, este utilizará el módulo de radio. Esta decisión se ha tomado en base a que el módulo de radio consume menor cantidad de batería que el módulo GPS. Siguiendo este criterio, el GPS solo será activado cuando el dispositivo se encuentre fuera del alcance de la radio del Sink, y el GPS sea el único modo de obtener la posición.

En esta sección, detallaremos cuales son los cuatro estados diferentes y la responsabilidad de cada una de ellas.

3.2.2. Estados de funcionamiento

A la hora de desarrollar la aplicación, se ha trabajo sobre la posibilidad de que el dispositivo cambie entre cuatro estados diferentes (más un quinto que no se considera en el proyecto), siendo dos de ellos de tránsito y los otros dos estados estables. En conjunto son capaces de cumplir las especificaciones detalladas anteriormente. En la siguiente tabla mostramos los 4 estados (más el *extra*) y una breve descripción de su cometido.

Tabla 3.1. Estados de las motas

Estado	Descripción de estado
Estado 1: Dentro del nido	▪ Indicación de posición periódica
Estado 2: Fuera del nido	▪ Toma de posición periódica
Estado 3: Sale del nido	▪ Cambiar el nodo de estado 1 a estado 2 ▪ Encender módulo GPS
Estado 4: Entra al nido	▪ Cambiar el nodo de estado 2 a estado 1 ▪ Apagar módulo de GPS ▪ Transmitir datos almacenados
Estado 5: Actualización software	No contemplado en el proyecto

Los dos primeros estados se encargan de las recogida de posición dependiendo de la posición en la que se encuentra el nodo. Por otra parte, los estados dos y tres, se encargan del tránsito entre los dos primeras estados. Aparte de eso, el estado cuatro, se encarga de mandar al Sink los datos recabados durante el periodo fuera del nido. Los dos primeros estados se realizarán de forma periódica, mientras que los estados tres y cuatro son de ciclo único.

Descripción de tránsitos:

1. Respuesta de Sink
2. No respuesta de Sink
3. No zona nido
4. Zona nido

3.2.3. Estado 1: Dentro del nido

En esta categoría se describirá el estado uno.

Descripción general del estado

Este estado ejecuta una función que comprueba periódicamente si el nodo sigue dentro del nido. Para ello, el nodo manda una mensaje al Sink con una petición de posición. Si recibe la respuesta

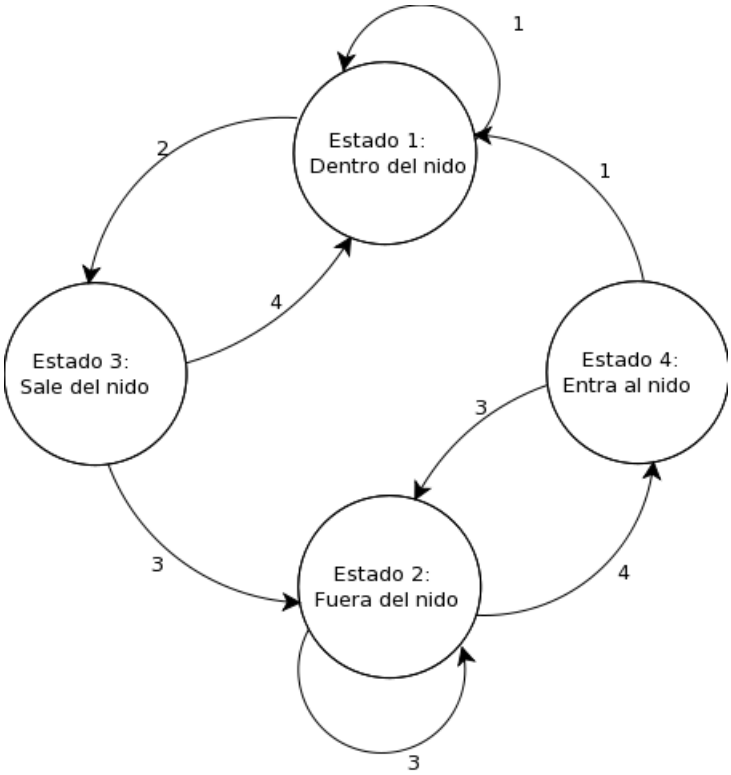


Figura 3.1. Estados y transiciones

del Sink el nodo sabrá que se encuentra dentro del nido. Para ello solo se utilizará el módulo de radio y no el de GPS.

A la hora de implementar la aplicación se han barajado dos formas de verificar la posición del nodo, si esta se encuentra dentro o fuera del nido, y así saber si hay necesidad de cambiar de estado o no. Para ello se podría utilizar tanto el módulo de radio como el módulo GPS. Como se ha comentado anteriormente, la opción del GPS supone un mayor consumo de batería. Por este motivo, se decide implementar la verificación de la posición del nodo mediante la opción de radio.

Directamente ligada a esta decisión, se tomó otra decisión que reduce el consumo de memoria en el nodo sin acarrear mayor gasto de energía. Sabiendo que el nodo establece comunicación con el Sink cuando se encuentra dentro del nido, se decidió que este guarde los datos de tiempo y localización de esa toma. Así el nodo solo tendrá que almacenar los datos cuando se encuentre fuera del nido.

Descripción de implementación

En este estado se configuran distintos parámetros, entre ellos, la frecuencia con la que el nodo se comunicará con el Sink, que a su vez será la que indique la frecuencia de toma de datos. Hay que tener en cuenta, que probablemente se activarán simultáneamente varias motas. Por este motivo es importante controlar y evitar en la medida de lo posible las colisiones en la comunicación con el Sink dado que las colisiones generan retransmisiones y, por tanto, consumo adicional de energía.

El nodo permanecerá dormido hasta que el temporizador active un evento para que se despierte.

Primero de todo, el nodo manda una mensaje de radio al Sink. Al mismo tiempo se activa un temporizador, el cual indicará un tiempo de respuesta para el nodo Sink.

Al finalizar este temporizador, se comprobará si el nodo ha recibido respuesta del Sink. En caso afirmativo, se interpreta que el nodo sigue dentro del nido y se continuará en el estado uno hasta volver a activarse el temporizador de recogida de datos. En caso contrario, se dará por hecho que el nodo ha salido del nido y, por lo tanto, del alcance del radio del Sink. Cambiando del estado estable uno al estado de tránsito tres.

En este estado, también se puede dar el caso de que el nodo tenga que transmitir los datos almacenados. Esto se explicará más adelante, ya que seria un caso especial si la transmisión fallase la primera vez.

Figura 3.2. Estado 1

3.2.4. Estado 2: Fuera del nido

En esta categoría se describirá el estado dos.

Descripción general del estado

En este estado, el nodo guarda periódicamente la posición en la que se encuentra, además de la hora en la que se ha recogido la información. En un entorno real se utilizará un módulo de GPS para obtener la posición, pero en este proyecto se realizará una simulación del software en Cooja sin utilizar ningún hardware externo.

No obstante, el diseño modular seguido en la aplicación permite que si en un futuro se plantea implantar el software en hardware real, no suponga ningún cambio global en la aplicación. Tan sólo habría que adaptar el módulo del GPS para que interactuase con el hardware GPS.

Para suplir la falta de un hardware que nos proporcione las coordenadas, en el simulador Cooja se ha utilizado, un plugin de Cooja denominado Mobility que permite indicar la trayectoria que deben seguir las motas en la simulación. Adicionalmente, también se ha tenido que programa un script asociado a Cooja que permita sustituir la funcionalidad del GPS al proporcionar a la mota las coordenadas en las que se encuentra en ese momento de la simulación. El plugin Mobility se presentará con más detalle en el apartado Mobilidad de las motas: Mobility.

Al estado uno se accederá desde el estado cuatro, y se saldrá de este estado mediante el estado tres.

Descripción de implementación

Al igual que en el estado uno, en este estado, un temporizador activará periódicamente la recogida de datos. En primer lugar, se comprobará si se encuentra o no dentro de la zona de radio. Para eso se enviará un mensaje de radio y al mismo tiempo se pondrá en marcha otro temporizador. Una vez terminado el tiempo del temporizador, en caso de que:

1. El nodo ha recibido respuesta: se confirma que se encuentra en la zona de radio, por lo que el nodo pasará al estado cuatro, en el cual se gestionan los módulos de radio y GPS para cambiar al estado uno.
2. Una vez pasado el tiempo del segundo temporizador, si el nodo no ha recibido ninguna respuesta, se dará por hecho que el nodo sigue fuera del nido. Con lo cual la única acción que se llevará a cabo será la de guardar la posición y horas en el nodo.

Figura 3.3. Estado 2

3.2.5. Estado 3: Sale del nido

En esta categoría se describirá el estado tres.

Descripción general de estado

Este estado de funcionamiento es de ciclo único y se ha creado como estado de tránsito entre el estado uno y el estado dos. Se asegura que el nodo se encuentra fuera del nido y que el anterior estado ha sido dentro del nido, o estado uno. A su vez, este estado implica que se va acumulando nueva información para enviar al Sink más adelante, ya que una vez salgamos de la zona de anidamiento, las tomas de posiciones se irán almacenando en la mota.

A este estado se accede desde el estado uno y mandará al nodo al estado dos.

Descripción de implementación

Como en los primeros estados, se envía un mensaje al Sink y se espera su respuesta. Una vez expirado el temporizador de espera, si no se ha recibido ninguna respuesta, se dará por hecho que estamos fuera del nido. Si el estado anterior era el estado uno, entonces es cuando activaremos este estado de tránsito La acción que se llevará a cabo será la de encender el módulo GPS.

A este estado se accede desde el estado dos y mandará al nodo al estado uno.

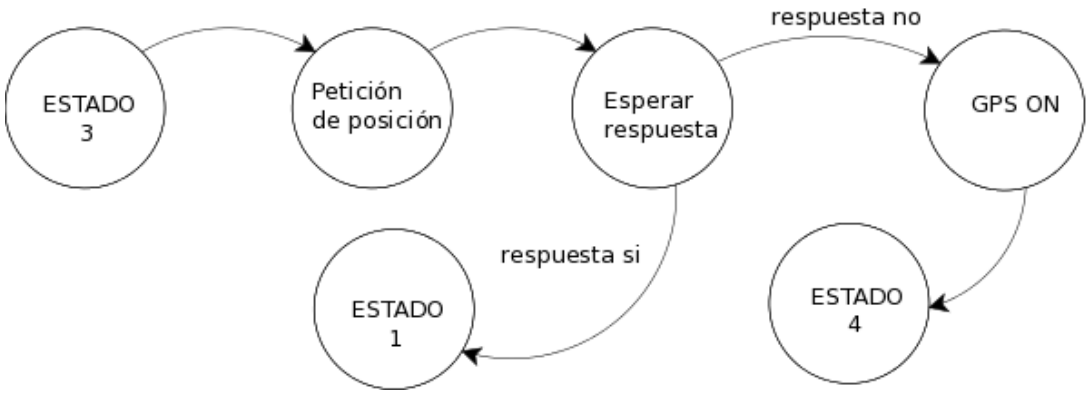


Figura 3.4. Estado 3

3.2.6. Estado 4: Entra al nido

En esta categoría se describirá el estado cuatro

Descripción general de estado

Este estado de funcionamiento es de ciclo único y se ha creado como estado de tránsito entre el estado dos y el estado uno. Al entrar en este estado se asegura que el nodo se encuentra dentro del nido y que el anterior estado haya sido fuera del nido o estado dos. A su vez, este estado será el responsable de mandar la información almacenada en el nodo, ya que este estado significa que nos trasladamos desde fuera del nido donde hemos recogido y almacenado información, a dentro del nido, donde tenemos acceso a comunicación por radio.

La decisión de transmitir los datos en el estado tres es debido a que es el único momento en el que se sabe que hay datos nuevos sin necesidad de verificarlo. Un dato a tener en cuenta es que, el primer intento de mandar los datos almacenados se da en el estado tres, pero se puede dar el caso de que los datos no lleguen correctamente. Para darle solución a ese inconveniente, el nodo no borrará los datos almacenados hasta recibir una verificación de que el Sink ha recibido los datos correctamente. Para ello, si en el primer intento no se lleva a cabo el objetivo, el nodo seguirá intentando mandar los datos una vez transitado al estado uno. Mientras el nodo no reciba el mensaje que verifica que los datos han sido recibidos correctamente, este no borrará los datos y hará un nuevo intento de mandar los datos cada vez que se despierte a recoger datos.

3.2.7. Optimización del almacenamiento de los datos de posición

Para almacenar la ruta seguida por una mota no es necesaria una precisión elevada. En este apartado se detalla uno de los protocolos de comunicación más utilizados en los receptores GPS comerciales. A continuación mostraremos el formato utilizado en nuestra aplicación, el cual se ha establecido simplificando el comercial previamente descrito. En este apartado también se discutirán las técnicas de simplificación utilizadas.

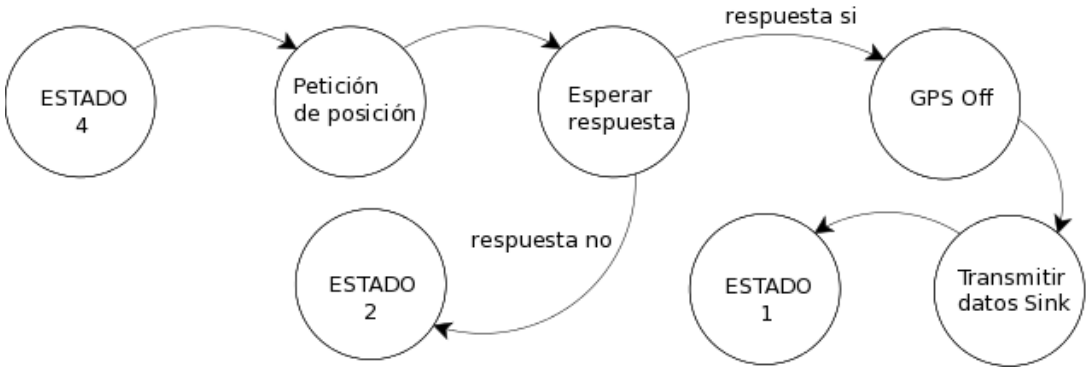


Figura 3.5. Estado 4

Este criterio pretende conseguir un ahorro de espacio en el dispositivo, y a su vez, un consumo menor de batería, ya que las tramas enviadas serán de menor tamaño y se hará menor consumo del módulo de radio.

Protocolo de comunicación NMEA

El protocolo NMEA (National Marine Electronics Association) es un estándar para el intercambio de información entre elementos electrónicos de navegación, entre ellos el GPS. Su utilización es recomendable dado el alto grado de compatibilidad que se puede conseguir en la gestión de la información.

El formato NMEA consiste en una línea de texto con caracteres ASCII. La línea comienza siempre con el carácter “$” y luego el nombre de una sentencia NMEA seguida de un número determinado de campos (los que contienen la información), los cuales, están separados por comas.

Las sentencias NMEA empiezan con las letras GP excepto las sentencias propias de los fabricantes que son adicionadas al repertorio general de la NMEA. Ejemplo:

```
$GPBOD,campo1,campo2,campo3,...,campo15
```

Las principales sentencias del estándar NMEA son:

Tabla 3.2. Sentencias NMEA

Sentencia	Descripción
GPBOD	Recorrido de origen destino
GPGGA	Datos de localización
GPGLL	Datos de latitud y longitud
GPGSA	Datos de los satélites en general
GPGSV	Datos de un satélite detallado
GPRMC	Datos mínimos de posicionamiento, tiempo y velocidad
GPRTE	Datos de una ruta (datos de entrada y salida)
GPWPL	Datos de un “waypoint” (datos de entrada-salida)
GPRMB	Datos mínimos de navegación
GPVTG	Datos del “track” y velocidad de movimiento

De estos mensajes sólo interesa guardar en el dispositivo la información relativa a la posición y hora. Estos datos se encuentran en el mensaje GPGGA (datos de localización).

A continuación se detallan los datos que se transmiten en este mensaje:

```
$GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh
$GPGGA,115510.20,3426.34,N,61932.26,W,1,08,1.25,8.2,M,3.4,M,,*4F
```

Tabla 3.3. Mensaje desglosado

#Campo	Ejemplo	Tamaño (byte)	Descripción
0	$GPGGA	6	Global Positioning System Fix Data
1	115510.20	9	Hora en formato: hhmmss.ss
2	3426.34	10	Latitud en formato: llll.ll
3	N	1	Orientación: N(norte) o S(sur)
4	61932.26	11	Longitud en formato: yyyyy.yy
5	W	1	Orientación: E(este) o W(oeste)
6	1	1	Indicación de calidad GPS: 0=nula/1=fija
7	08	2	Número de satélites visibles por el receptor
8	1.25	4	Dilución horizontal de posiciónn
9	8.2	7	Altitud de la antena en formato: x.x
10	M	1	Unidades de altitud: M (metros)
11	32.4	4	Superación geoidal: x.x
12	M	1	Unidad de separación: M (metros)
13	▪	▪	Tiempo desde la ultima actualización (segundos)
14	▪	▪	En modo difrenecial. ID de la estación
15	*4F	3	Checksum
16	▪	▪	Retorno de carro y salto de línea

Por lo tanto, necesitaríamos 71 bytes para en la memoria para guardar todos estos datos. Pero, como anteriormente hemos comentado, a nosotros solo nos interesarían parte de los datos de esa tabla. Concretamente los siguientes:

Tabla 3.4. Mensaje desglosado: campos

#Campo	Ejemplo	Tamaño (byte)	Descripción
1	115510.20	9	Hora en formato: hhmmss.ss
2	3426.34	10	Latitud en formato: llll.ll
3	N	1	Orientación: N(norte) o S(sur)
4	6932.26	11	Longitud en formato: yyyyy.yy
5	W	1	Orientación: E(este) o W(oeste)

Guardando estos datos en la memoria, tendría un tamaño en memoria de 32 bytes por cada toma realizada. Este formato es conveniente en el caso de querer mostrar los datos en pantalla, ya que se nos entregan como texto. Teniendo en cuanta que nosotros lo que queremos es guardar los datos en la memoria, este método no es eficiente. Por lo tanto realizaremos unos cambios que se explicarán a continuación.

Minimizando el volumen de datos

La estrategia que planteamos para reducir el volumen de datos es, por una parte, transformar los datos de caracteres ASCII a datos decimales y, por otra parte, acotar los valores de los datos que se almacenarán.

Convertir cadena de caracteres en dato decimales En forma ASCII los datos nos viene de siguiente manera:

- Hora (9 bytes): Horas/Minutos/Segundos/Décimas de segundos

En forma ASCII los datos nos viene de siguiente manera:

- Hora (9 bytes): Horas/Minutos/Segundos/Décimas de segundos
- Latitud(10bytes + 1byte): Grados/Minutos/Fracciones de minutos, Orientación
- Longitud(11bytes + 1 byte): Grados/Minutos/Fracciones de minutos , Orientación

Si transformamos estos caracteres de texto a números entero, de la manera en la que se muestra en el siguiente cuadro:

Campo		Valores posibles	Espacio (bytes)
HORA	Horas Minutos Segundos D.segundos	0-23 0-59 0-59 0-99	1 1 1 1
LATITUD	Grados Minutos Mminutos	-90-90 0-59 0-99999	1 1 2
LONGITUD	Grados Minutos Mminutos	-180-180 0-59 0-99999	2 1 2

Con este cambio de formato, pasaríamos de necesitar 32 bytes a tan solo necesitar 13 bytes. No obstante, como ya hemos comentado antes, podemos reducir aún más el tamaño de acotando además los datos.

Acotar los datos La cantidad de datos como la exactitud ofrecidas por los módulos GPS son mayores de las que nosotros requerimos. Es por eso que se busca reducir el espacio necesario lo máximo posible pero a su vez satisfacer las necesidad de la aplicación, con el fin de reducir notoriamente el volumen de los datos almacenados.

A continuación indicaremos de que manera hemos acotado los datos. Como ya hemos dicho minimizándolos al máximo pero a su vez respondiendo a nuestras necesidades.

Hora En cuanto a este campo, no vemos necesidad de guardar los segundos ni las décimas de segundo. En todo caso el error seria de menos de un minuto, algo que no nos parece significativo en nuestra aplicación. Por este motivo parece suficiente con utilizar 2 bytes, uno para la hora y otro para los minutos.

Latitud y Longitud Para estos dos parámetros seguiremos la misma estrategia de acotamiento. Como hemos indicada para el campo de hora, para estos dos tampoco nos parece necesario guardar los minutos. Con lo cual se guardarán los grados y minutos, siendo un coste de 2bytes para longitud y otros 2bytes para latitud.

Con lo cual la cantidad de bytes que necesitaremos para almacenar cada toma después de llevar a cabo este proceso será el siguiente:

Campo		Valores posibles	Espacio (bytes)
HORA	Horas Minutos	0-23 0-59	1 1
LATITUD	Grados Minutos	-90-90 0-59	1 1
LONGITUD	Grados Minutos	-180-180 0-59	2 1

Con esta técnica reducimos considerablemente la precisión, pero a su vez reducimos volumen de datos. Considerando que el sacrificio de precisión, esta justificada por el ahorro de espacio y batería.

Con lo cual, de 32bytes que almacenaríamos cada toma en el dispositivo, utilizando estas dos técnicas, solamente necesitaremos 6bytes de memoria. Esto nos supone un 81,25 % de ahorro de espacio.

Referencias consultadas en este apartado: [6]

3.3. WSN y Contiki

3.3.1. Wireless Sensor Networks (WSN)

Las redes de sensores inalámbricos o *Wireless Sensor Networks*, en adelante WSN, son redes formadas por un gran número de dispositivos o *motas* muy limitados en términos de potencia de cálculo, memoria, comunicación y energía y que conforman una red sin una infraestructura definida previamente. Habitualmente incorporan sensores que permiten recoger información del entorno (razón por la que habitualmente se encuentran distribuidas de modo espacial) y que, utilizando los recursos de la red, hacen llegar a unos nodos especiales, denominados *sink*, nodos que habitualmente hacen las veces de enlace con las redes convencionales (para así poder exportar la información).

Debido a las limitadas características hardware de las motas, la gestión de dichos dispositivos en todos las capas de funcionamiento (gestión de módulos, comunicación...) debe estar adaptada a estas restricciones. En este proyecto utilizaremos el sistema operativo Contiki.

3.3.2. Contiki OS

Contiki [2] es un sistema operativo diseñado para sistemas con hardware muy limitado, como es el caso de las WSN. Es de código abierto e implementa de modo nativo el protocolo IPv6 sobre IEEE802.15.4 ofreciendo una capa con 6LoWPAN. Además ofrece varias opciones para gestionar la capa MAC, entre las que destacamos ContikiMAC y soporte para una gran cantidad de plataformas hardware. A pesar de la gran cantidad de funciones que ofrece, destaca por su escaso consumo en términos de memoria y energía.

Las aplicaciones para Contiki pueden ser desarrolladas utilizando C estándar, lo cual supone un gran ventaja frente a otras opciones. Además, dispone de una librería denominada *protothread* que permite implementar multitarea de un modo relativamente sencillo y ofrece recursos adicionales como temporizadores.

Contiki ofrece también la posibilidad de simular las aplicaciones mediante un simulador denominado Cooja. Esta herramienta destaca por su flexibilidad, dado que puede simular código programado en distintos niveles, si bien en este proyecto sólo se ha considerado la simulación a nivel de código final (para facilitar su posterior implantación en hardware real).

Además de las herramientas que incorpora por defecto (ver apartado Cooja: herramientas utilizadas), Cooja también dispone en la actualidad de una gran variedad de plugins, algunos de los cuales se utilizan en este proyecto (ver apartados Simulación del consumo: PowerTrace y Mobilidad de las motas: Mobility).

3.4. Módulo de gestión de energía

Uno de los puntos importantes en las WSN es el consumo de batería. La mayoría de este tipo de redes están pensadas para implantarlas y no volver a ser modificadas. Esto es, no tener que cargar o cambiar las baterías. Por este motivo optimizar el consumo de batería es una de las prioridades en el diseño de sus aplicativo.

Dado que la apliación será probada y evaluada mediante simulación, no se podrá medir el consumo real de la aplicación. Por tanto, es necesario disponer en el simulador de un módulo o plugin que permita estimar el consumo de la aplicación en las distintas pruebas.

3.4.1. Simulación del consumo: PowerTrace

Como se ha señalado anteriormente, en este proyecto se utiliza el simulador Cooja de Contiki para simular la ejecución de la WSN diseñada. En este apartado se analiza una de las opciones que ofrece Contiki para simular cual seria el consumo de batería en las red. Para ello se ha utilizado el módulo PowerTrace.

Powertrace [9], [4] es un programa que puede ser incorporado como plugin a simuladores como Cooja y que permite estimar el consumo de energía generado por la ejecución de la aplicación simulada. Según [9], la estimación proporcionada por PowerTrace tiene un 94 % de precisión sobre los valores reales [9].

Utilización

El código de Powertrace se encuentra contenido en el fichero `powertrace.c`. Para utilizar su funcionalidad en la aplicación desarrollada, es suficiente con añadir el siguiente código.

```
#include "powertrace.h"
powertrace_start(CLOCK_SECOND * 2);
```

Nota: Dentro del paréntesis se indica cada cuando se desea que el sistema recabe información.

Información proporcionada

La función de PowerTrace, nos ofrece más información aparte del consumo de batería del sensor. Entre ellas: el tiempo del reloj interno, el identificador del sensor, cuantas veces se ha dado esta información... En nuestro caso la información que nos interesa se encuentra en los atributos:

Tabla 3.5. Atributos PowerTrace

Atributo	Descripción
All_cpu	el consumo de la cpu trabajando al 100 %
Last_cpu	el consumo de la cpu trabajando a media/baja carga
All_transmit	el consumo transmitiendo
All_listen	el consumo en escucha

En la siguiente imagen, se puede observar una captura de los output de la simulación. Cada línea representa un *print* de la aplicación. En las líneas con igual formato a la primera son las correspondientes a trazas de PowerTrace. Los atributos que se encuentran marcados, son los cuatro atributos anteriormente indicados.

```
2543    ID:2    260 P 2.0 0 2392 63146 0 370 0 370 2392 63146 0 370 0 370 (radio 0.56% / 0.56% tx 0.00% / 0.00% listen 0.56% / 0.56%)
2690    ID:1    260 P 1.0 0 2392 63146 0 370 0 370 2392 63146 0 370 0 370 (radio 0.56% / 0.56% tx 0.00% / 0.00% listen 0.56% / 0.56%)
4544    ID:2    516 P 2.0 1 5445 125627 0 764 0 764 3050 62481 0 394 0 394 (radio 0.58% / 0.60% tx 0.00% / 0.00% listen 0.58% / 0.60%)
4690    ID:1    516 P 1.0 1 5445 125627 0 764 0 764 3050 62481 0 394 0 394 (radio 0.58% / 0.60% tx 0.00% / 0.00% listen 0.58% / 0.60%)
5966    ID:1    broadcast message sent
6019    ID:2    broadcast message received from 1.0: 'Hello'
6545    ID:2    772 P 2.0 2 8670 187935 0 1251 0 1145 3223 62308 0 487 0 381 (radio 0.63% / 0.74% tx 0.00% / 0.00% listen 0.63% / 0.74%)
6671    ID:2    broadcast message sent
6693    ID:1    772 P 1.0 2 13026 183579 1586 1208 0 1133 7579 57952 1586 444 0 369 (radio 1.42% / 3.09% tx 0.80% / 2.42% listen 0.61% / 0.67%)
6791    ID:1    broadcast message received from 2.0: 'Hello'
8548    ID:2    1028 P 2.0 3 16277 245862 1577 1691 0 1513 7605 57927 1577 440 0 368 (radio 1.24% / 3.07% tx 0.60% / 2.40% listen 0.64% / 0.67%)
8693    ID:1    1028 P 1.0 3 16355 245775 1586 1713 0 1514 3326 62196 0 505 0 381 (radio 1.25% / 0.77% tx 0.60% / 0.00% listen 0.65% / 0.77%)
10547   ID:2    1284 P 2.0 4 19467 308206 1577 2085 0 1907 3187 62344 0 394 0 394 (radio 1.11% / 0.60% tx 0.48% / 0.00% listen 0.63% / 0.60%)
10693   ID:1    1284 P 1.0 4 19509 308155 1586 2107 0 1908 3152 62380 0 394 0 394 (radio 1.12% / 0.60% tx 0.48% / 0.00% listen 0.64% / 0.60%)
11770   ID:1    broadcast message sent
11894   ID:2    broadcast message received from 1.0: 'Hello'
12547   ID:2    1540 P 2.0 5 22775 370424 1577 2571 0 2289 3306 62218 0 486 0 382 (radio 1.05% / 0.74% tx 0.40% / 0.00% listen 0.65% / 0.74%)
12694   ID:1    1540 P 1.0 5 27228 365969 3160 2549 0 2277 7717 57814 1574 442 0 369 (radio 1.45% / 3.07% tx 0.80% / 2.40% listen 0.64% / 0.67%)
```

Figura 3.6. Salida creada desde el módulo PowerTrace

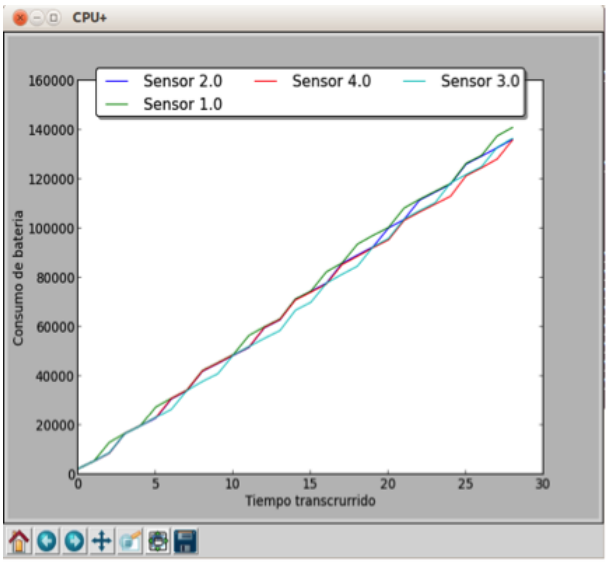
Como se puede apreciar en la imagen, no es fácil comparar los resultados de esa manera. Y más aun, si tenemos en cuenta que en este ejemplo solo interactúan dos sensores. En una red con mayor cantidad de sensores seria el procesado de las trazas se complica de modo exponencial. En el apartado Visualización se muestra cómo procesar dicha información para generar gráficos.

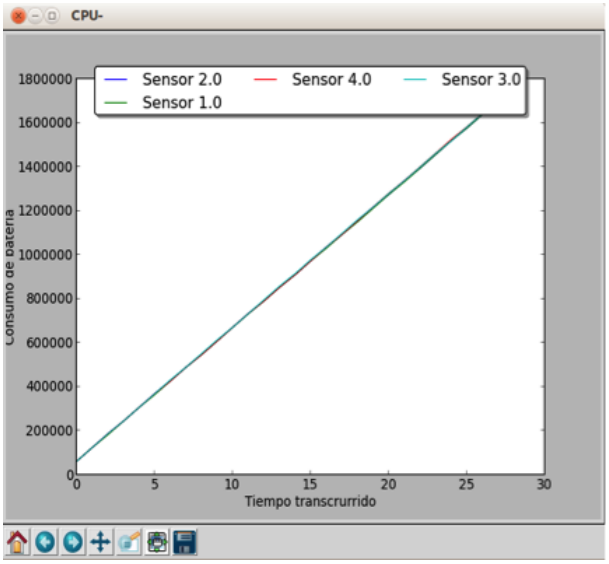
Visualización

El procesado de las trazas generadas se realizan mediante un pequeño script en Python (ver Apéndice *Anexo A: Consumo de batería: generación de gráficos*). Este script muestra gráficos con los datos que

sean relevantes para nosotros. Para ello necesitamos los datos que se muestran en pantalla en un archivo de texto, pero para eso, Cooja ya contempla una opción.

Una vez ejecutado el script, como resultado mostraría gráficos como los que se muestran a continuación.

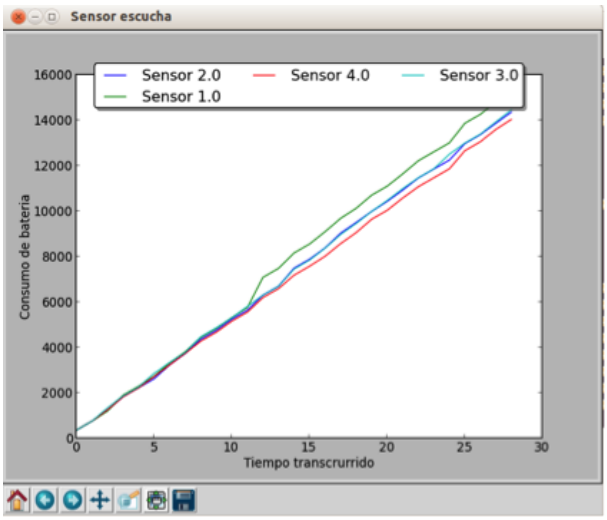


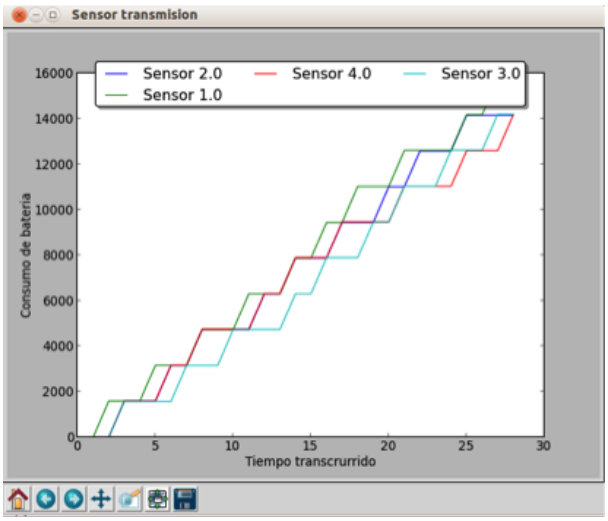


Con un total de cuatro gráficos, se muestran las cuatro diferentes variables que nos interesan. En las dos primeras el consumo de la CPU y en las dos últimas el del módulo de radio. Como se puede apreciar, con este método es más fácil comparar los datos extraídos, pudiendo diferenciar los nodos por colores y teniendo un solo tipo de dato en cada momento.

Consumo del GPS

El punto negativo de utilizar PowerTrace es que solo nos proporciona información acerca del CPU y la radio. Sin embargo, también estamos interesados en conocer el consumo del GPS. Por ese motivo hemos implementado dentro del módulo de GPS una simulación de consumo.


Sensor escucha
Sensor 2.0
Sensor 4.0
Sensor 3.0
Sensor 1.0
Consumo de bateria
Tiempo transcrurrido
0
2000
4000
6000
8000
10000
12000
14000
16000
0
5
10
15
20
25
30


Sensor transmision
Sensor 2.0
Sensor 4.0
Sensor 3.0
Sensor 1.0
Consumo de bateria
Tiempo transcrurrido
0
2000
4000
6000
8000
10000
12000
14000
16000
0
5
10
15
20
25
30

Se distinguen tres estados diferentes: apagado, reposo y encendido. Al igual que para PowerTrace, se indica cada cuanto queremos que se recabe información. La simulación ira sumando un consumo diferente dependiendo de cual de esos tres estados se encuentre. Cada cierto tiempo se mostrará por la salida *output* de Cooja, una línea con la siguiente información `G ID Consumo`.

3.5. Módulo GPS

En el sistema diseñado cada mota incorpora un módulo GPS. Sin embargo, este tipo de hardware no suele estar presente en las motas debido a su costo y a su consumo. Cooja no contempla por defecto la gestión de módulos GPS en su simulación y por este motivo en este proyecto ha sido necesario simular su funcionalidad.

3.5.1. Obtener las coordenadas

Dado que las motas pueden variar su posición durante la simulación, el módulo que simula el dispositivo GPS debe obtener dicha posición, obteniendo las correspondientes coordenadas X,Y en el mapa del simulador. En Cooja dicha información se puede obtener implementando una funcionalidad especial en la que la mota consulta al simulador del siguiente modo:

```
var x = mote.getInterfaces().getPosition().getXCoordinate();
var y = mote.getInterfaces().getPosition().getYCoordinate();
```

Con esas dos llamadas tendremos los dos valores que indican las coordenadas del nodo.

Posición en tiempo preciso

Para saber la posición de un nodo en una hora determinada, dependiendo del tipo de WSN se utilizarán métodos diferentes. Analizaremos como obtener la posición en dos tipos de redes.

Una de ellas, cuando los nodos tienen capacidad de comunicarse entre ellos sin interrupciones. Esto es, los nodos son estáticos, no se mueven de sitio, o los nodos se mueven distancias pequeñas y no se alejan hasta aislarse.

Por otra parte, puede haber otro tipo de redes en las que los nodos se dispersan perdiendo comunicación entre ellas, y en un momento dado vuelven a encontrarse para comunicarse con el Sink. Un ejemplo para este tipo de red es la de nuestro proyecto.

Comunicación constante

Si hay capacidad de comunicación constante, es suficiente con mandar una petición desde el nodo Sink al nodo cuya posición queremos saber. Este nodo nos devolverá una respuesta con sus coordenadas, X e Y.

Comunicación puntual

En este caso, no se podrá obtener la posición en tiempo real, pero si se podrá calcular donde se ha encontrado el nodo en una hora exacta. Para ello, deberemos esperar a que el nodo se comunique con el Sink y retransmita la información diaria.

Dentro de esta información se encontrara un registro de posiciones tomadas a lo largo del día. Como ya se ha dicho antes, estos dispositivos poseen una pequeña capacidad de almacenamiento. A la hora de tomar registro de las posiciones, habrá que tomar en cuenta esta limitación.

Una vez el nodo transmita los datos recolectados durante el tiempo que ha estado aislado, se llevará a cabo una simulación de trayectoria uniendo los puntos que el nodo ha recogido en ese periodo de tiempo. Un registro con una mayor cantidad de posiciones facilitaría crear una simulación más exacta.

Ejemplo

En la siguiente imagen, observamos la posición que ha tomado un mismo nodo en el transcurso de cuatro horas y media.

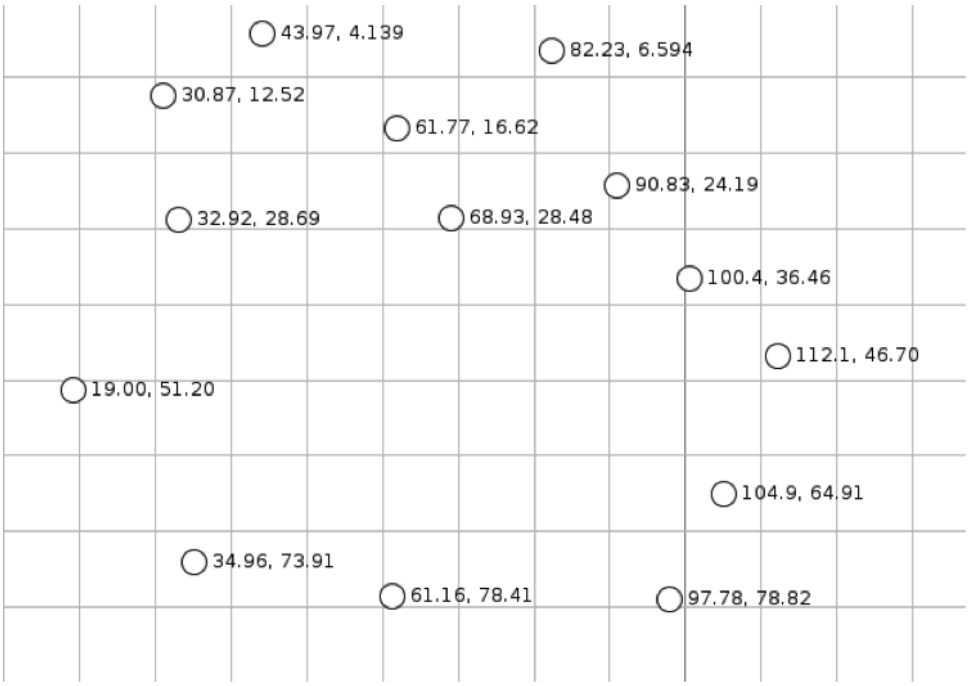


Figura 3.7. Posición de un nodo a lo largo del tiempo

Como se puede observar, el nodo ha tomado distintas posiciones a lo largo del periodo de tiempo (se le ha dado la orden de registrar su posición cada veinte minutos).

Una vez el nodo vuelva al punto de partida, comunicará con el nodo Sink, transmitiendo los siguiente datos

Tabla 3.6. Gestión de las posiciones

Hora	Coord X	Coord Y
10:00	34.96	73.91
10:20	61.16	78.41
10:40	97.78	78.82
10:40	97.78	78.82
10:40	97.78	78.82
11:00	104.9	64.91
11:20	112.10	46.70
11:40	100.40	36.46
12:00	90.83	24.19
12:20	68.93	28.48
12:40	61.77	16.62
13:00	82.23	6.54
13:20	43.97	4.13
13:40	30.87	12.52
14:00	32.92	12.52
14:20	19.00	51.2

Con esta información podremos crear una línea tiempo y deducir cual ha sido la posición del nodo en cualquier momento de ese intervalo.

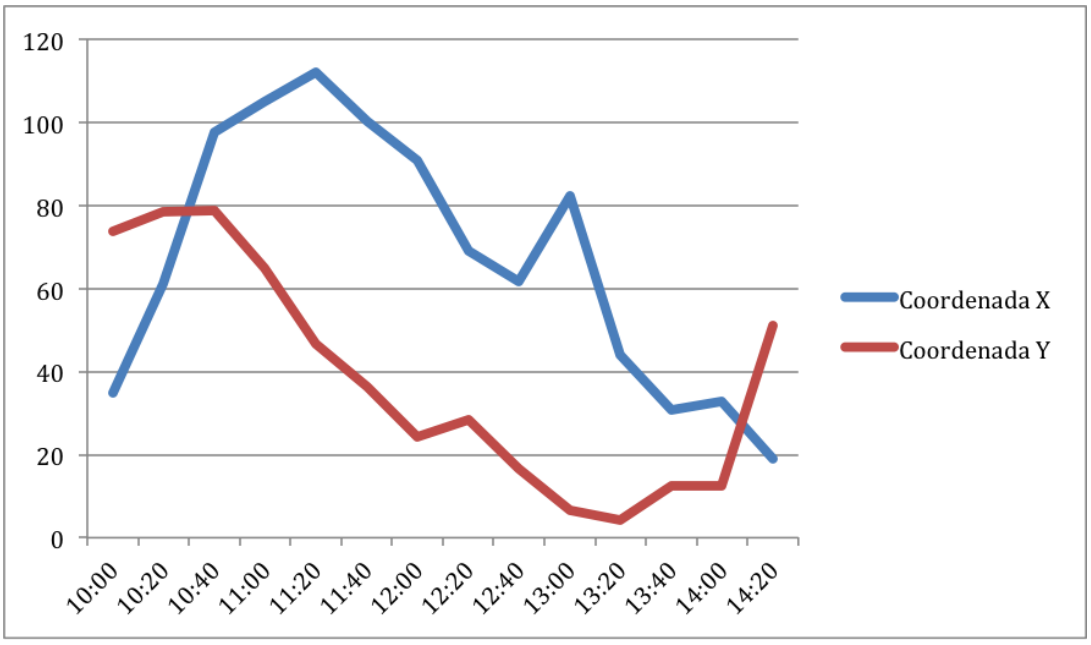


Figura 3.8. Gráfico de las posiciones

Cabe destacar que esta información no es 100 % precisa, cuanto más pequeño sea el intervalo entre las capturas de información más nos acercaremos a la realidad.

3.6. Módulo de radio

El módulo de radio es utilizado en todos los estados, y en el estado tres y uno incluso para mandar los datos almacenados al Sink. No obstante, no en todo momento se hace uso de la radio, con lo cual, se tiene que encender cuando sea necesario pero mantenerse en reposo durante el resto del tiempo. Para gestionar esto, Contiki incorpora una implementación en el módulo de radio denominada ContikiMac, la cual analizaremos a continuación.

ContikiMAC [8] es un protocolo de ciclos de trabajo de radio que utiliza despertares periódicos para escuchar las transmisiones de paquetes de los vecinos. Si se detecta una transmisión de paquetes durante un despertar, el receptor se mantiene despierto para recibir el paquete. Cuando el paquete se recibe correctamente, el receptor envía un mensaje para indicar que se ha recibido correctamente. Para transmitir un paquete, un remitente envía repetidamente su paquete hasta que recibe un mensaje de envío correcto.

Contiki ofrece otras opciones [5] tales como X-MAC y LPP, en este proyecto se ha utilizado ContikiMAC sin realizar un análisis comparativo ya que este aspecto queda fuera del ámbito del proyecto. Entre las líneas de trabajo futuro se encontraría encontrar la opción más adecuada en función del escenario de aplicación.

3.7. Mobilidad de las motas: Mobility

El simulador Cooja no ofrece la opción de mover las motas simuladas siguiendo posiciones predefinidas, siendo la única opción por defecto la de mover las motas interactivamente (mediante el sistema gráfico) durante la simulación. Sin embargo, existe un plugin para Cooja denominado Mobility que permite incorporar dicha opción.

Mobility [3] es un plugin implementado en Java por Federik Osterlind cuya funcionalidad es la de dotar movimiento a los nodos en las simulaciones realizadas en Cooja utilizando para unas rutas predefinidas. Más concretamente, se puede definir la posición en la que cada nodo se encontrará en un determinado momento por lo que también se puede simular el escenario de la aplicación, en la que existe un nodo estático, el sink, y regularmente las motas acuden a dicho punto.

Una de las características interesantes de Mobility es que trabaja con un formato compatible con el generador de posiciones BonnMotion [1], generador de posiciones/rutas desarrollado en la Universidad de Bonn.

3.7.1. Especificación

Para definir la ruta de movimientos de cada nodo es necesario modificar el archivo `positions.dat`. Un simple ejemplo de este archivo seria el siguiente:

```
0     0.0    1     0
1     0.0    2     0
0     0.2    5     1.5
1     0.2    5     2.5
0     0.5    2     0
1     0.5    3     0
```

Como podemos observar, cada línea se divide en diferentes columnas. Cada columna tiene un significado distinto, siendo importante respetar el orden de las columnas:

- Columna1: En la primera columna indicaremos el identificador del nodo, este identificador se le asignará a cada nodo cuando se añada en la simulación realizada en Cooja. Los identificadores comienzan desde 0 a partir del primer nodo que se añada.
- Columna2: En la segunda columna se identificará el tiempo en el que se realizará el movimiento. El tiempo se deberá anotar en segundos. Hay que tener en cuenta que el tiempo escrito en la línea X debe de ser mayor o igual a las escritas en las líneas anteriores a ella. De no ser así, la simulación mostrará un error al llegar a esa línea.
- Columna3: En la tercera columna indicaremos la coordenada X que tomará el nodo en la nueva posición. Esta coordenada se puede dar tanto en número entero como en decimales.
- Columna4: En la cuarta columna indicaremos la coordenada Y que tomará el nodo en la nueva posición. Esta coordenada se puede dar tanto en número entero como en decimales.

Aclaraciones

En el proyecto que llevamos a cabo, la simulación de movimiento tiene que cumplir los siguientes requisitos:

- El nodo Sink se encuentra en un punto fijo (no se mueve)
- Los nodos se mueven siguiendo patrones estimados y acotado y acotados. Esto quiere decir que una vez recorra todos los movimientos simulados tiene que comenzar de nuevo.

Para definir el punto en el que se situará el nodo Sink, es suficiente con añadir en la primera línea del archivo `positions.dat` lo siguiente:

```
0 0.0 0 0
```

Con esto lo que haremos será que el nodo se mantenga en las coordenadas X=0 y Y=0 de modo estático.

Para que los nodos se muevan ininterrumpidamente, tendremos que cerrar el ciclo de movimiento. Para ello, las primeras coordenadas X e Y que indicamos para un nodo tienen que ser las mismas que las ultimas. Así una vez llegue al final del archivo `positions.dat` comenzará de nuevo desde la primera posición.

3.8. Cooja: herramientas utilizadas

Las principales herramientas de análisis que brinda el simulador COOJA para realizar análisis del funcionamiento de una simulación:

- Mote output
- Radio messages
- Timeline
- Network
- Msp Code watcher

Con estas herramientas podremos observar la información de la simulación de distintas formas. Para controlar la simulación utilizaremos el panel de control el cual nos dará la opción de parar, pausar y proseguir con la simulación en distintas velocidades. La información mostrada en el panel de control concierne al tiempo transcurrido de simulación, el momento de la ultima parada y la velocidad relativa de la simulación.

3.8.1. Mote Output

Esta herramienta nos muestra los mensajes que los nodos imprimen durante su ejecución utilizando la instrucción `printf()`. El uso más común es la de comprobar el correcto funcionamiento de los nodos, pudiendo visualizar variables utilizadas para el funcionamiento de las comunicaciones entre los nodos.

La interfaz divide la información en tres partes distintas: `Time ms`, `Mote` y `Message`. Dándonos la información concerniente al tiempo en el que se ha impreso, el nodo que a llevado a cabo el comando, y que mensaje ha impreso. En la siguiente figura se puede observar el resultado:

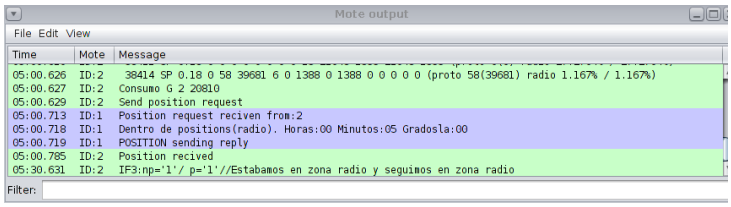


Figura 3.9. Herramientas de Cooja: Mote output

La cantidad de información mostrada en la ventana dependerá de dos factores. Por una parte la cantidad de nodos que tengamos en marcha en la simulación, y por otra parte, la cantidad de instrucciones `printf()` que contiene cada nodo. Puede darse el caso que la cantidad de datos sea tan grande, que resulte muy difícil su lectura, para ello la herramienta posee un filtro. Escribiendo lo que necesitemos buscar en el campo `Filter` se mostrará la información que contenga dicho texto. Por ejemplo si queremos ver la información sobre el nodo 2, es suficiente con ingresar el texto ID:2 en el campo `Filter`.

La opción `save to file`, exporta los datos a un `archivo.txt`. Este opción, nos es valida más adelante para extraer los datos sobre la batería y mostrarlos en gráficos mediante un script.

3.8.2. Radio messages

La información disponible en el *radio messages* esta vinculada a la interfaz de IEEE 802.15.4. En la siguiente imagen podremos observar el aspecto que posee.

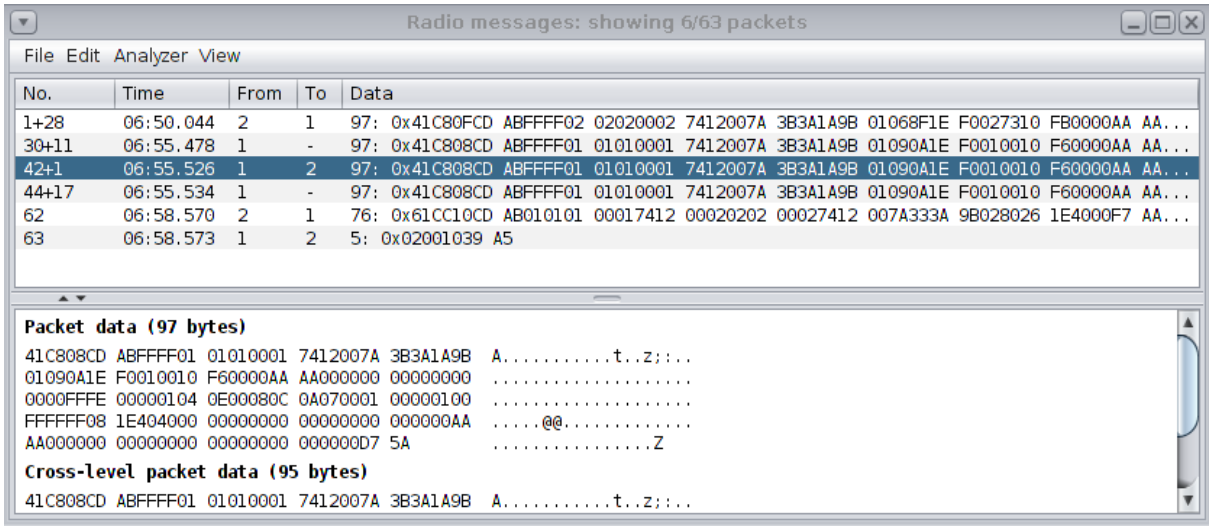


Figura 3.10. Herramientas de Cooja: Radio messages

En las columnas podemos ver la siguiente información: tiempo de simulación, ID del nodo transmisor, ID del nodo receptor y el dato que transmite.

3.8.3. Timeline

En esta ventana se presenta la actividad de radio y leds de todos los nodos a la vez.

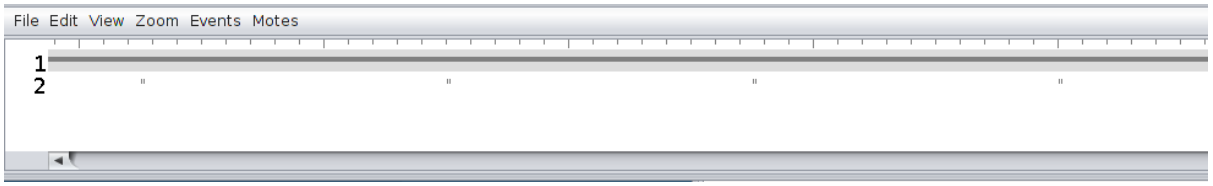


Figura 3.11. Herramientas de Cooja: Timeline. Salida conjunta de los nodos

La ventana se divide en líneas horizontales, cada línea horizontal representa un nodo, diferenciando por colores cada actividad.

3.8.4. Network

En esta ventana, podremos observar de modo gráfico la colocación de cada nodo. Pudiendo configurar diferentes parámetros de visualización tales como: coordenadas, alcance de radio, leds, tráfico de radio... Las opciones disponibles se muestran en la siguiente imagen

Figura 3.12. Herramientas de Cooja: Timeline. Salida diferenciada por nodo

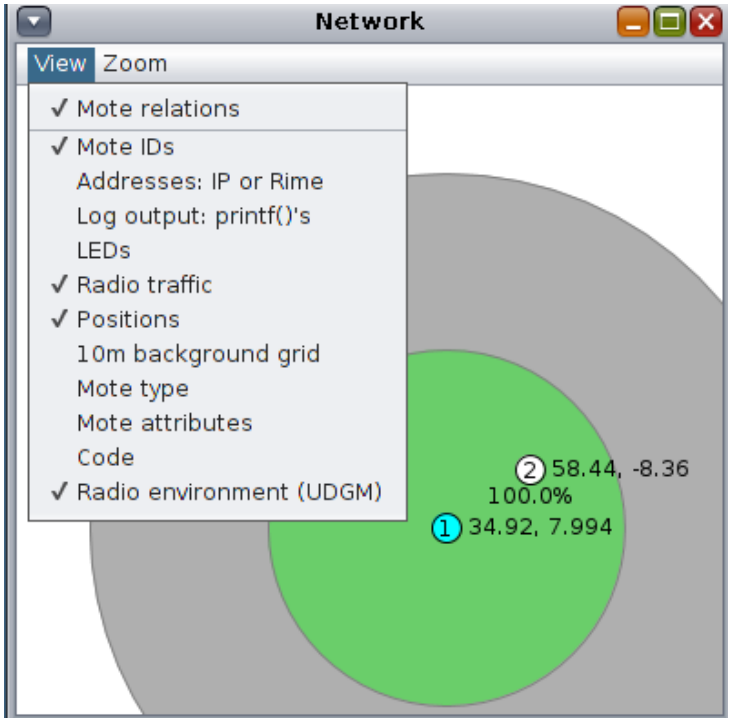


Figura 3.13. Herramientas de Cooja: Network. Características de la red

En nuestro caso las opciones más utilizadas han sido los que se encuentras seleccionados en la imagen: Mote id, Radio traffic, Positios, Radio enviroment.

- *Mote id*: para saber cual es el identificador de cada nodo.
- *Radio traffic*: Para ver las líneas de tráfico entre los distintos nodos.
- *Positions*: Muestra la coordenada del nodo en la simulación.
- *Radio enviroment*: Nos muestra el alcance de la señal de radio del nodo, en color verde el alcance útil del nodo 1.

3.8.5. Msp Code Watcher

Es la interfaz que nos permite visualizar el código que se esta ejecutando en cada nodo, tanto en C como en ensamblador, pudiendo agregar *watchpoints*. Esta herramienta nos ha sido de gran ayuda para utilizarla de modo *debugger*, pudiendo avanzar paso por paso en el código y analizar que sucede cada instante.

Es posible ejecutar un *msp code watcher* por cada nodo de la simulación, así analizando por separado la ejecución de cada nodo.

Para añadir un *watchpoint* es necesario presionar con el botón derecho en la parte del código en el que se quiera añadir. Este *watchpoint* poseerá un nombre y color para identificarlo, cuando la ejecución pase por ese punto ser parara, siempre y cuando se le haya seleccionado la casilla stop con anterioridad. Si ese no fuese el caso, la simulación proseguirá dejando una marca en el Timeline.

3.9. Pruebas y evaluación

Para comprobar el correcto funcionamiento de la aplicación y observar los resultados en diferentes escenarios se han llevado a cabo varias pruebas. Por otra parte, también se ha querido evaluar el consumo de batería en distintas situaciones.

El consumo de energía es un punto sensible en las redes de sensores inalámbricas. Por eso es necesario llevar un control del consumo de los nodos. Ya sea para cambiar el comportamiento o para optimizar comportamientos erróneos.

Como se presentaba en el apartado Simulación del consumo: PowerTrace, en nuestro proyecto para llevar a cabo esta tarea se ha utilizado la herramienta PowerTrace.

Se han llevado a cabo ciertas pruebas para observar cual seria el consumo de batería de los nodos y su comportamiento. Para ellos hemos planteado unos entornos hipotéticos y los hemos probado en el simulador Cooja.

Plantearemos tres situaciones distintos, las cuales serán de un consumo mínimo de batería a un consumo máximo. Cada situación será reflejada por un nodo en la simulación, esta se llevarán a cabo durante 12 horas.

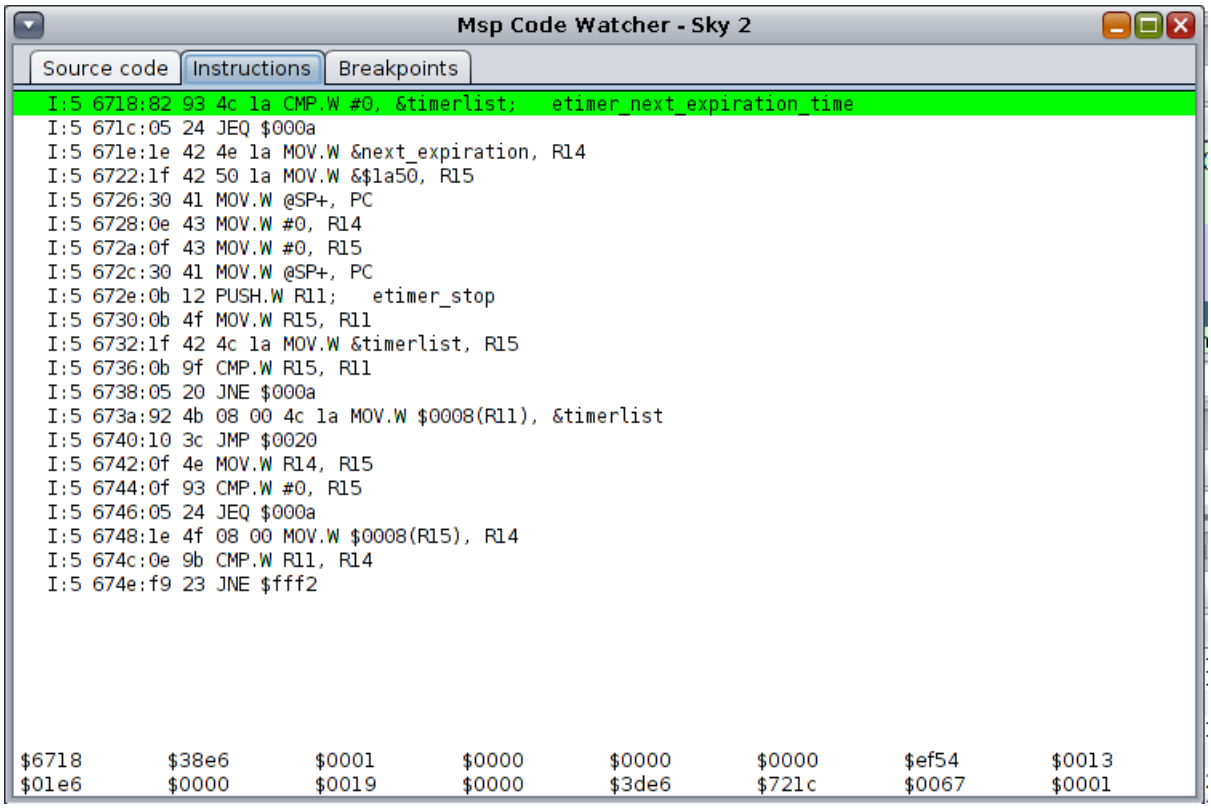


Figura 3.14. Herramientas de Cooja: msp code watcher

Cada nodo se repartirá de la siguiente manera en la simulación. Siendo el nodo 1 el que tomara el papel del Sink.

3.9.1. Nodo 2 Menor consumo

Como primera prueba, nos hemos planteado simular el menor consumo posible de batería. Barajando las dos opciones que tiene el nodo, estar dentro o fuera del nido, y teniendo en cuenta que el módulo de radio consume menos energía que el módulo GPS, se ha planteado pasar todo el tiempo en el primer estado.

En la siguiente imagen se puede observar cual ha sido la posición del nodo, número dos, y el nodo Sink, número uno.

3.9.2. Nodo 3 Consumo intermedio

Como segunda prueba, nos hemos planteado simular un consumo de batería intermedia, esto es, unas consumo con un uso cotidiano del nodo. Suponiendo que el dispositivo le sea implantado a un pájaro con vida diurna, el dispositivo pasará las horas de luz fuera del nido y la noche en el nido.

Siendo esto así y con las limitaciones planteadas a nuestra simulación, las 12 horas que durará, nuestro planteamiento será la de 6 horas de luz y 6 horas de noche. Con lo cual es dispositivo pasará las 6 primeras horas fuera del nido y las 6 restantes dentro de ella.

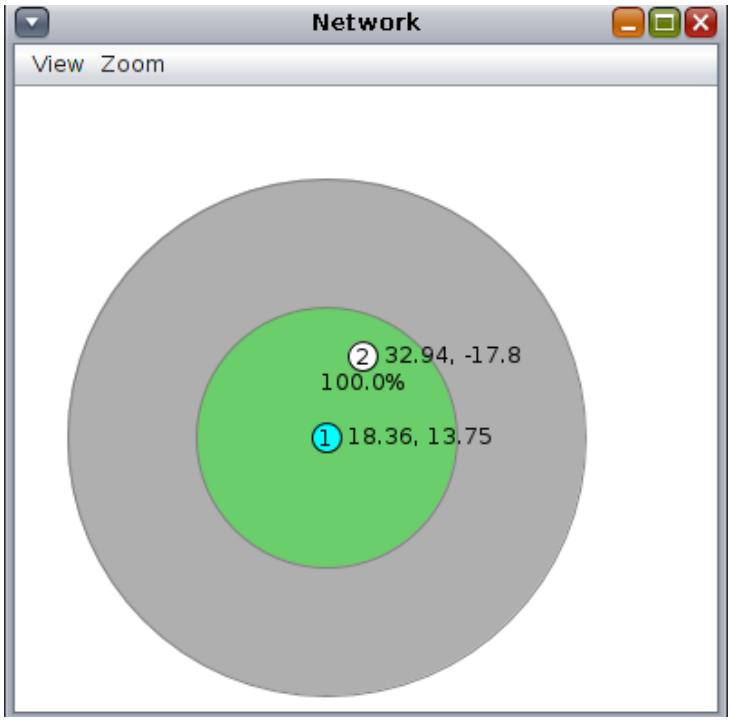


Figura 3.15. Prueba: menor consumo

En la siguientes imagen se puede observar cual ha sido la posición del nodo, número tres, y el nodo Sink, número uno.

Fuera del nido y dentro del nido respectivamente

3.9.3. Nodo 4 Mayor consumo

Como tercera y última prueba, nos hemos planteado simular el mayor consumo de energía posible. Barajando las dos opciones que tiene el nodo, estar dentro o fuera del nido, y teniendo en cuenta que el módulo GPS consume más energía que el módulo de radio, se ha planteado pasar todo el tiempo en el segundo estado, esto fuera del nido.

En la siguiente imagen se puede observar cual ha sido la posición del nodo, número cuatro, y el nodo Sink, número uno.

3.10. Conclusiones

Con las pruebas realizadas podemos llegar a las siguientes conclusiones. Por una parte, los planteamientos teóricos hechos han sido respaldados por los resultados. Siendo los resultados los esperados, teniendo un mayor consumo el nodo 12horas fuera del nido y menor consumo teniéndolo dentro del nido.

Para reflejarlo, hemos obtenido los siguientes gráficos a partir de los datos obtenidos en la simulación. Dichos gráficos muestran el consumo de cada nodo en los siguientes aspectos: CPU, radio y GPS.

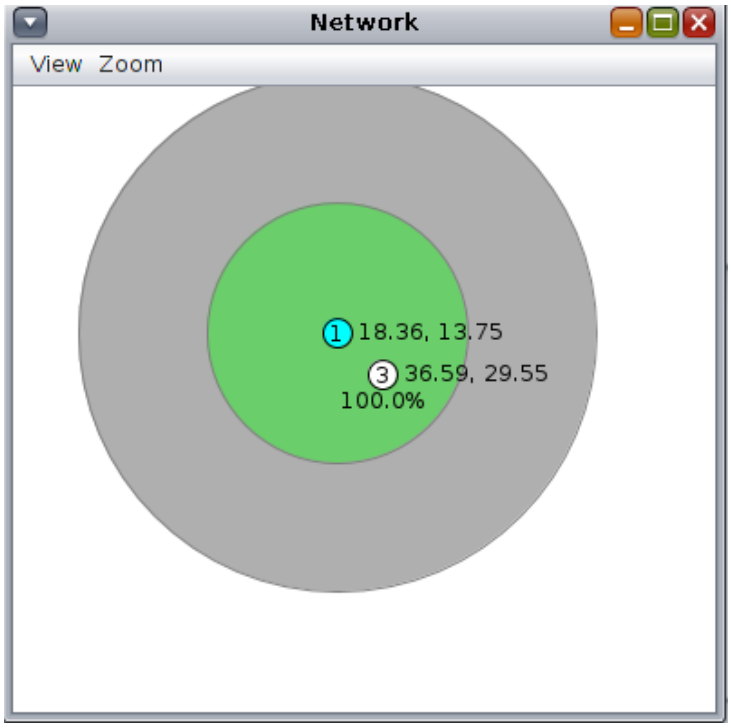


Figura 3.16. Prueba: consumo intermedio fuera del nido

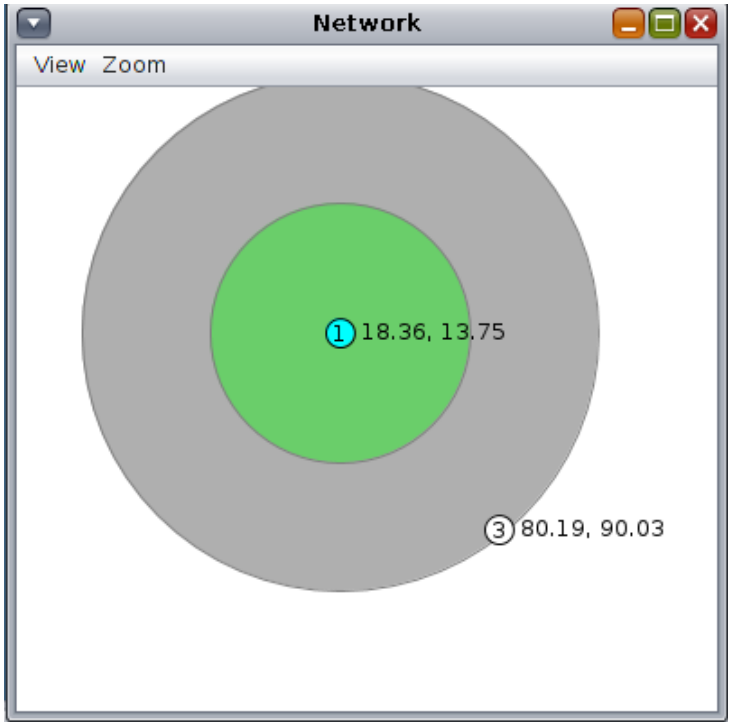


Figura 3.17. Prueba: consumo intermedio dentro del nido

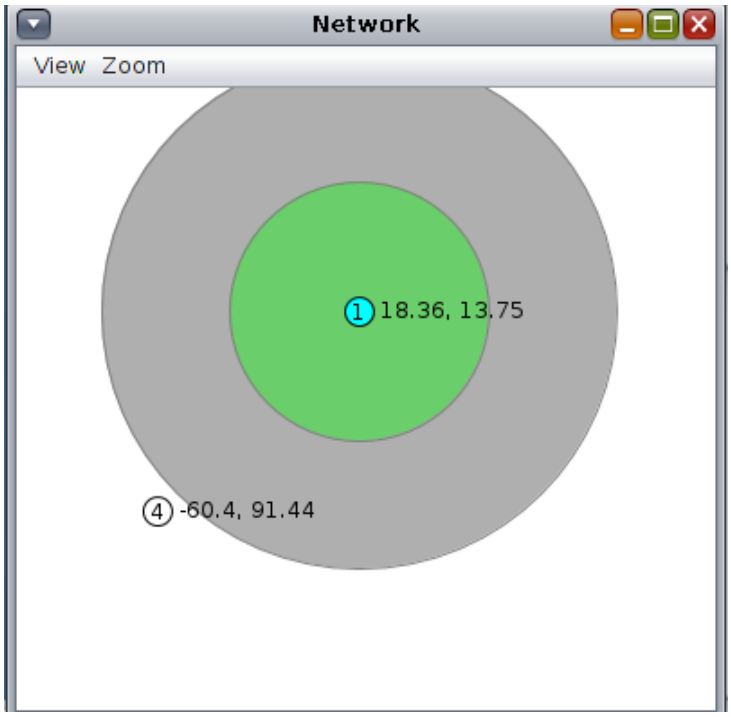


Figura 3.18. Prueba: mayor consumo

Consumo de CPU los tres nodos:

En el gráfico se observa que el consumo de los tres nodos ha sido el mismo. De aquí se puede concluir que el consumo de CPU es el mismo en todos los estados.

Consumo de radio de los tres nodos:

En el anterior gráfico se puede observar que el mayor consumo lo han tenido nos nodos 4 y 3. Estos dos nodos son los que se han movido fuera del nido, el nodo 4 durante las 12 horas y el nodo 3 durante 6 horas. El consumo de radio es mayor debido principalmente a dos factores. Primeramente, al usar la radio para decidir si el nodo se encuentra dentro del nido o no, cuando el nodo se encuentra fuera del nido, este realiza el intento de comunicarse repetidamente, aumentando su consumo. Por otra parte, los nodos que hay recolectando información fuera del nido utilizarán la radio para transmitir esa información al Sink.

En cuanto al consumo de GPS, en el gráfico no se refleja la línea verde, la cual representa al nodo 2. Puesto que este nodo se ha encontrado dentro del nido durante toda la simulación no utilizando dicho módulo. En cuando a los otros dos nodos, el consumo ha sido el doble para el nodo 4, ya que este a pasado el doble de tiempo utilizando el GPS.

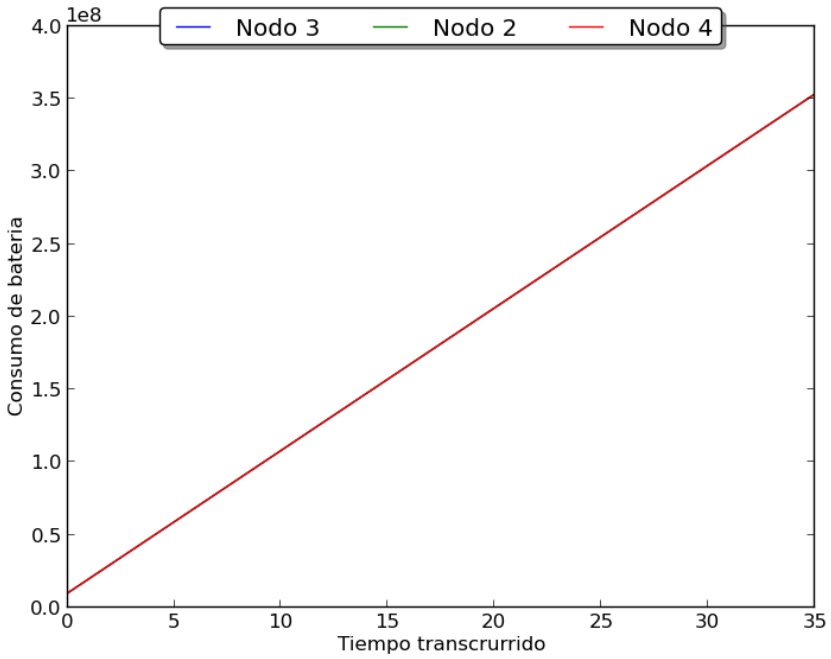


Figura 3.19. Consumo de CPU en la simulación

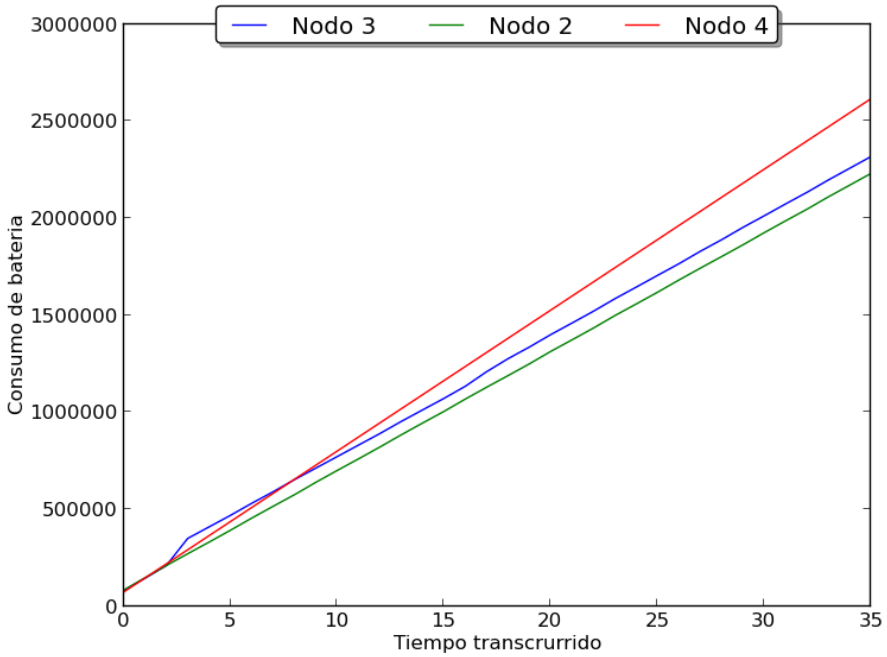


Figura 3.20. Consumo de radio en la simulación

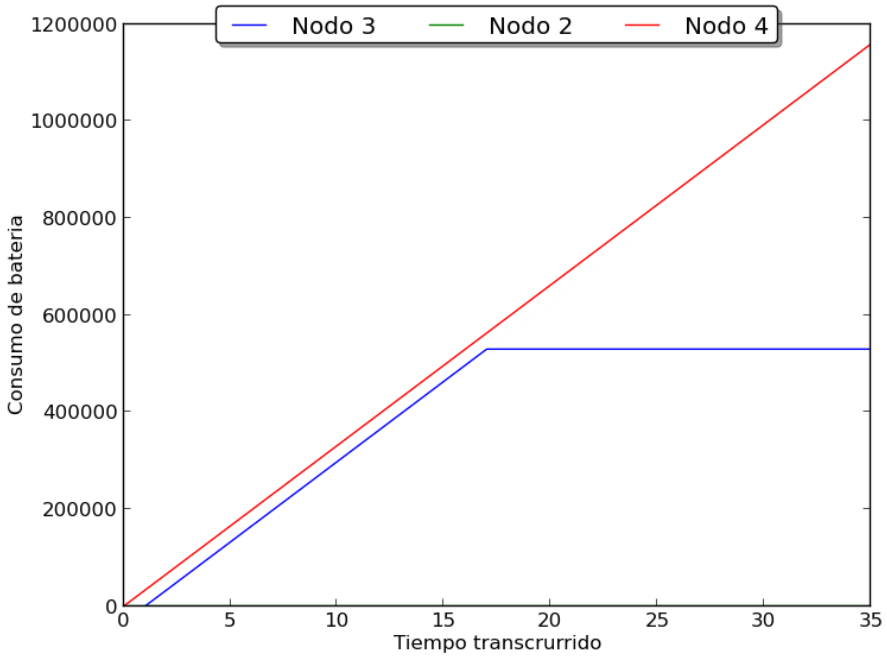


Figura 3.21. Consumo del módulo GPS de los tres nodos

4. CAPÍTULO

Resultados, Conclusiones y Líneas Futuras

4.1. Resultados

El resultado obtenido en este proyecto ha sido el deseado, ya que se han cumplido los objetivos planteados en un principio.

Se propuso implementar un software de seguimiento para individuos basado en el sistema operativo Contiki. Dentro de los objetivos prioritarios planteados se encontraban implementar cuatro diferentes estados (como continuación de un trabajo previo), un módulo de radio, un módulo GPS y un módulo para simular la batería. Por otra parte, también se planteó la opción de una ampliación del proyecto añadiendo un quinto estado. Para poder llevar a cabo este proyecto se debía cumplir unos requisitos.

El proyecto se realizó dentro de la estimación global de horas de trabajo, razón por la que no se pudo acometer las tareas que se habían planteado como posibilidad en caso de que al final de los objetivos principales todavía quedara un remanente de horas de trabajo.

En conclusión, el resultado de este proyecto ha sido un prototipo de la aplicación según las especificaciones y que ha sido probado mediante simulación.

4.2. Conclusiones

El desarrollo del proyecto se ha ajustado a la planificación inicial en cuanto a tiempo total invertido (375 horas) y metas propuestas. Sin embargo esto no quiere decir que todas las tareas se estimasen correctamente. Durante el proyecto se han ido compensando las carencia de tiempos de algunas tareas con el tiempo restante en otras. La principal razón para estar en que en la estimación de alguna de esas tareas ha sido la de falta de conocimiento suficiente del entorno. El proyecto se repartió en distintas etapas, y en las siguientes líneas mostraremos las conclusiones que hemos sacado de cada una de ellas.

Siendo la primera etapa la del estudio de Contiki y el lenguaje de programación C, la carga de trabajo estimada, 30 horas, se quedó muy lejos de la real, llegando a duplicarla en el caso del estudio de Contiki, en la que se estimaron 10 horas y se invirtieron 24. Esto fue debido a que se sobreestimó el conocimiento del lenguaje C y se infravaloró la dificultad de programar aplicaciones para en Contiki.

La segunda etapa seria la de diseño e implementación del software. La estimación realizada, con un total de 220 horas entre los dos, ha sido bastante precisa, con un desviación de 30 horas menos por debajo de lo planteado, cabe destacar que la implementación se simplificó más de lo esperado por el gran consumo de tiempo leyendo y entendiendo el código ya escrito. Esto ha sido una clara repercusión de la sobreestimación de conocimiento del lenguaje C.

En lo que a la documentación se refiere, el tiempo estimado ha sido suficiente para cumplir con la tarea. Pero, hay que tener en cuenta que parte del trabajo realizado se ha reutilizado de trabajos anteriores, facilitando notablemente a la hora de escribir la documentación.

En cuanto a la gestión, se han tenido que tomar decisiones cruciales a lo largo del proyecto. En un momento dado se tuvo que sobrepasar el numero de horas estimadas para la primera tarea de implementación que se realizó, el módulo de radio, para el que se invirtieron 38 horas, 5 más de las estimadas . Cabe destacar que esas horas fueron sobrepasadas conscientemente, con la creencia de que esas horas extra serian en beneficio del proyecto. Por otra parte, a la hora de implementar el módulo GPS y los estados, se tuvo que tomar la decisión de hacer las cosas más simples de lo que en un principio se esperaba. De no hacerlo, se corría el riesgo de dejar el proyecto a medias.

4.3. Líneas Futuras

En cuanto a líneas futuras, se ven dos claras ramas por las cuales se puede proseguir este proyecto. Por una parte implantar la aplicación en un entorno real y por otra profundizar en la implementación del código.

Como ya se ha comentado anteriormente, la aplicación desarrollada en este proyecto ha sido para ejecutarla en el simulador Cooja. Sería interesante probar la portabilidad de dicho código a una plataforma real. Para ello seria necesario cambiar el software para que interactuase con el hardware del nodo. En cuando al módulo GPS, habría adaptarlo al interfaz del módulo elegido, ya que está implementado para que trabaje con el simulador Coooja y no con un módulo real de hardware.

Por otra parte, debido a limitaciones de tiempo y conocimiento, no se ha profundizado lo deseado en la implementación. También seria interesante mejorar el código ya existente optimizándolo más, dándole así una mayor eficiencia. No solo eso, también seria posible añadir más funcionalidades a los nodos o incluso al Sink. Entre las funcionalidades que se podrían añadir se encontraría la implementación del quinto estado. El cual se utilizaría para actualiza el software de los nodos.

En cuando al funcionamiento global, cabria la posibilidad de modificar el comportamiento de los nodos dependiendo de la carga de la batería. Además de poder añadir un placa solar para la recarga de la batería y mayor durabilidad de la mota. Esto seria interesante para un proyecto basado más en la gestión de hardware real, aspecto que queda fuera del alcance de nuestro proyecto.

En cuanto al Sink, se podría trabajar en dar una salida de los datos almacenados mediante algún medio, tal como Internet. Por ejemplo, una vez tratados los datos, se podrían dejar disponibles mediante servicio web mostrando las rutas de los individuos de forma gráfica.

5. CAPÍTULO

Glosario

6LoWPAN IPv6 over Low power Wireless Personal Area Networks

Implementación de IPv6 diseñada para dispositivos de capacidades reducidas.

Estándar que permite utilizar IPv6 en redes basadas en el estándar IEEE 802.15.4. Hace posible que dispositivos como los nodos de una red inalámbrica puedan comunicarse directamente con otros dispositivos IP.

ASCII American Standard Code for Information Interchange

API Application Programming Interface

CoAP Constrained Application Protocol HTML HyperText Markup Language

DAG Directed Acyclic Graph

DAO Destination Advertisement Object

DIO DODAG Information Object

DIS DODAG Information Solicitation Message

DODAG Destination Oriented DAG

DIO DODAG Information Object

DIS DODAG Information Solicitation

DODAG Destination Oriented Directed Acyclic Graph ICMP Internet Control Message Protocol

ETX Excepted Transmission Count

GNSS Global Navigation Satelit System

HTTP HyperText Transport Protocol

IEEE Institute of Electrical and Electronics Engineers

IESG Internet Engineering Steering Group

IETF Internet Engineering Task Force

IoT Internet of Things

IP Internet Protocol

IPv6 Internet Protocol, version 6

ISP Internet Service Provider

JSON JavaScript Object Notation

MAC Media Access Control layer NFC Near field communication

NMEA National Marine Electronics Association

LLN Low Power and Lossy Network

MAC Medium Access Control

OF Objective Function

OSI Open Systems Interconnection PHY Physical layer

RAM Random-Access Memory

RDC Radio Duty Cycling

RFC Request for Comments

RPL Routing Protocol for LLNs (Low power and Lossy Networks).

Protocolo de encaminamiento propuesto por IETF para redes limitadas en comunicación (baja tasa de transmisión de datos).

http://tools.ietf.org/html/draft-ietf-roll-rpl-19

RFID Radio-frequency identification

ROM Read-Only Memory

TCP Transmission Control Protocol

UDP User Datagram Protocol

URI Uniform Resource Indentifier

XML Extensible Markup Language

WSN Wireless Sensor Network

Estándar IEEE 802.15.4 Estándar para redes inalámbricas de área personal (WPAN) con baja tasa de transmisión de datos. Está centrado en las capa física y de control de acceso al medio.

WPAN Wireleess Personal Area Network

A. ANEXO

Anexo A: Consumo de batería: generación de gráficos

El Script escrito en Python que se ha utilizado en la visualización del consumo de batería es el siguiente:

```
#!/usr/bin/env python

import csv
import matplotlib.pyplot as plt

from collections import defaultdict
cpuOverTime =  defaultdict(list)
sensorOverTime =  defaultdict(list)
gpsOverTime =  defaultdict(list)

with open('loglistener.txt', 'rb') as f:
    reader = csv.reader(f,delimiter=' ')
    for row in reader:
        if len(row)>2:
            if row[2] is 'G':
                gpsOverTime[row[3]].append(row[4])
            if row[2] is 'P':
                cpuOverTime[row[3]].append(row[5])
        sensorOverTime[row[3]].append(row[6])

plt.figure('CPU')
plt.figure('Radio')
plt.figure('GPS')

for i in cpuOverTime:
    plt.figure('CPU')
    plt.plot(cpuOverTime[i], label ='Nodo '+i)
    plt.ylabel('Consumo de bateria')
    plt.xlabel('Tiempo transcrurrido')
```

```
30      plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.05),ncol=3, fancybox=True, shadow=True)
31
32      plt.figure('Radio')
33      plt.plot(sensorOverTime[i], label ='Nodo '+i)
34      plt.ylabel('Consumo de bateria')
35      plt.xlabel('Tiempo transcrurrido')
36      plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.05),ncol=3, fancybox=True, shadow=True)
37
38      plt.figure('GPS')
39      plt.plot(gpsOverTime[i], label ='Nodo '+i)
40      plt.ylabel('Consumo de bateria')
41      plt.xlabel('Tiempo transcrurrido')
42      plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.05),ncol=3, fancybox=True, shadow=True)
43
44  plt.show()
```

Bibliografía

[1] -. Bonnmotion. http://sys.cs.uos.de/bonnmotion, 2014. Accessed: 2014-01-10.

[2] -. Contiki website. http://www.contiki-os.org/, 2014. Accessed: 2014-01-10.

[3] -. Mobility plugin. http://sourceforge.net/p/contikiprojects/code/HEAD/tree/sics.se/mobility, 2014. Accessed: 2014-01-10.

[4] -. Powertrace website. https://github.com/contiki-os/contiki/tree/master/tools/powertrace, 2014. Accessed: 2014-01-10.

[5] -. Radio duty cycling. https://github.com/contiki-os/contiki/wiki/Radio-duty-cycling, 2014. Accessed: 2014-01-10.

[6] -. Receptor gps garmin etrex. http://www2.elo.utfsm.cl/~mineducagv/docs/ListaDetalladadeModulos/tutoria 2014. Accessed: 2014-01-10.

[7] -. Tinyos website. http://www.tinyos.net/, 2014. Accessed: 2014-01-10.

[8] Adam Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, Swedish Institute of Computer Science, December 2011.

[9] Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes. PowerTrace: Network-level Power Profiling for Low-power Wireless Networks. Technical Report, Swedish Institute of Computer Science, 2011.

[10] Rubén García. Estudio de viabilidad técnica de un dispositivo de seguimiento GNSS. Master's thesis, University of the Basque Country UPV/EHU, 2013.

Índice