Implementació d'un protocol d'enrutament de baix consum per la plataforma de sensors TelosB

Gabarró Mora, Pol

Curs 2014-2015

Director: BORIS BELLATA

GRAU EN ENGINYERIA TELEMÀTICA

upf. Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

Declaració d'Autoria

I, Pol Gabarró, declare that this thesis titled, 'Implementació d'un protocol d'enrutament de baix consum per la plataforma de sensors Telosb' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Engineers like to solve problems. If there are no problems handily available, they will create their own problems.

Scott Adams

UNIVESITAT POMPEU FABRA

Abstract

Escola Superior Politécnica
Grau en Enginyeria Telemàtica

by Pol Gabarró
Tutor: Boris Bellalta

L'objectiu d'aquest treball és implementar un protocol d'enrutament que permeti una millora del consum de bateria per xarxes de sensors. Per justificar el desenvolupament d'aquest protocol, inicialment es realitzarà una anàlisi de l'estat de l'art de les plataformes de monotorització sensors que són d'ús més comú a dia d'avui. Posteriorment, s'estudiarà el sistema operatiu per a microcontroladors TinyOS i s'explicarà el funcionament teòric i pràctic del protocol. Per finaltizar s'analitzarà el funcionament del protocol de forma pràctica realitzant diverses proves.

El objetivo de este trabajo es implementar un protocolo de enrutamiento que permita una mejora del consumo de batería en redes de sensores. Para justificar el desarrollo de este protocolo se hará un analisis del estado del arte de las plataformas de monotorización de sensores que son de uso más común a dia de hoy. Después se pasará a analizar el sistema operativo para microcontroladores TinyOS y se explicará el funcionamiento teórico y práctico del protocolo. Para finalizar se analizará el funcionamiento del protocolo de forma práctica haciendo diversas pruebas.

The objective about this work is implement a routing protocol that allow to improve the battery consumption for the wireless sensors. For justify the development of this protocol, first of all it will analize the state of the art of the platforms of sensor monitoring that are more used nowadays. Then it will study the operative system for microcontrollers TinyOS and it will explain the theoretical and practical operation of the protocol. To end, the implentation of protocol and their performance is analyzed.

Agraïments

A Boris Bellalta, per l'ajuda i el seu suport durant el llarg període d'elaboració d'aquest treball

A Toni Adame, per la idea i creació de l'eficient algoritme. Les seva ajuda ha sigut essencial per la realització del treball.

A la Família, que sense el seu esforç i suport no seria on sóc ara, i amb la formació que finalment he arribat a assolir.

I finalment a l'Ariadna, per donar-me suport i ànims quan no sabia per on anar durant l'elaboració d'aquest treball.

Índex

Índex de figures

Índex de codis

Índex de taules

Capítol 1

Introducció

Les xarxes de sensors van néixer per solucionar problemes en la recollida de dades mitjançant sensors en iniciatives militars o industrials. Des de un principi han estan orientades a sentir i transmetre informació de forma econòmica i en llocs on la falta de infraestructures on fer aquesta transmissió amb els medis tradicionals podia comportar moltes dificultats i alts sobre-costos. Aquesta necessitat feia que l'única forma de transmetre informació fos mitjançant comunicacions inal·lambriques [1] No tots els sensors per raons energètiques i de distància poden establir connexions punt a punt a la base per transmetre informació. Per tant es necessari la creació de protocols especialitzats en aquests sensors que permetin la creació de xarxes de sensors on els paquets d'informació poguessin saltar de sensors a sensor fins arribar al punt de recollida de dades i d'aquesta manera cobrir llargues distàncies.

Amb la popularització d'aquestes tecnologies en iniciatives com l'Internet de les coses, han provocat una baixada de preu fent l'ús de les xarxes de sensors més popular. També ha comportat que comencin a aparèixer problemes amb difícils solucions.
El principal problema actualment es l'autonomia, al estar pensats els sensors per estar situats en punts de difícil accés on no hi ha cap punt de connexió a la xarxa de corrent, per tant molts dels sensors han de dependre d'una font energètica mòbil, com seria una pila (bateria) o bé plaques solars entre d'altres. Per solucionar-ho es pot "atacar"el problema per diferents camins. Un d'ells es millorar el consum del hardware dels sensors com per exemple amb la millora de la miniaturització [2] o bé creant protocols que permetin una millora del consum i d'aquesta manera augmentar l'autonomia.

Precisament l'objectiu del treball es enfocar la solució del problema amb la millora del consum, intentant trobar una solució amb l'implementació d'un protocol d'enrutament que permeti minimitzar el cost energètic de les comunicacions entre els sensors. Això

permetrà augmentar la duració global de les bateries de la xarxa de sensors i d'aquesta forma estalviar diners en el manteniment i la compra de noves bateries.

1.1 Motivació

La motivació principal per desenvolupar aquest projecte és la necessitat de solucionar un dels molts problemes que ocasionen que l'autonomia d'una xarxa de sensors sigui molt baixa, ja que actualment la duració de les bateries és molt més curta del que seria desitjable, el que, en xarxes grans provoca uns costos de manteniment elevats [3]. Per aconseguir aquest propòsit, desenvoluparem un protocol d'enrutament de baix consum que tindrà com a objectiu mantenir estable l'enrutament d'una xarxa de varis sensors, amb un consum energètic mínim. Amb aquest nou protocol s'espera estalviar energia de tal forma que hi hagi un impacte real en l'autonomia dels sensors.

1.2 Problema a resoldre

En aquest TFG s'haurà d'implementar un nou protocol d'enrutament que permeti aconseguir un augment de l'autonomia de les xarxes de sensors. Posteriorment, s'haurà de demostrar que aquest protocol dóna uns resultats que justifiquin la implementació de la millora.

1.3 Objectius concrets

Per poder implementar aquest protocol serà necessari dissenyar tots els processos que tindran les diferents fases del protocol, els tipus de paquets i analitzar i provar les tecnologies necessàries pel seu funcionament.
També s'haurà de comprovar mitjançant diferents proves si hi ha una millora en el rendiment i eficiència de la xarxa una vegada funcionant el protocol.
Per acabar, es farà una proposta del camí que ha de seguir el protocol en futures millores.

1.4 Resum de continguts

En el primer punt s'analitzarà l'estat de l'art per justificar si hi ha espai per un nou protocol d'enrutament (Capítol 2). Seguidament, analitzarem el sistema operatiu en el que funciona el protocol i quines funcionalitats ens poden resultar útils (Capítol 3).

Una vegada tractada la plataforma passarem a descriure el funcionament del protocol i com està implementat (Capítol 4). El funcionament pràctic del protocol s'analitzarà en el següent punt, s'intentaran resoldre els problemes detectats i es proposaran millores (Capítol 5).
Per finalitzar es conclou el treball fent un resum de tots els punts treballats i fent un anàlisis de quin seria el camí per on s'hauria de continuar investigant (Capítol 6) .

Capítol 2

Estat de l'art

En aquest punt es farà un esment de l'estat de l'art de les xarxes de sensors. Inicialment es parlarà de les plataformes més importants en l'actualitat, és a dir, on es connecten els sensors donat que per si mateixes no són un element actiu. Seguidament, es tractarà la comunicació entre les plataformes per acabar amb les opcions més populars de monitorització.

2.1 Plataformes per a xarxes de sensors

Com s'ha esmentat en l'introducció, per poder justificar la viabilitat del projecte, abans de res, és necessari exposar les diferents plataformes de microcontroladors que actualment hi ha en el mercat amb més rellevància, i a partir d'aquí, justificar si el projecte té viabilitat o no.

2.1.1 Arduino

Avui en día el més utilitzat dintre d'un àmbit no professional[4]. Arduino va néixer l'any 2005 a l'institut IVREA d'Italia, en aquell moment els estudiants treballaven amb el microcontrolador BASIC Stamp[5]. Aquest controlador tenia un cost de 100$ fet que feia que fos massa costós per l'institut, per aquest motiu el professor Massimo Banzi (un dels fundadors de Arduino) va decidir desenvolupar una placa microncontroladora amb un cost inferior a 30$[6]. Tot i així, al ser un projecte de codi lliure es poden trobar clònics xinesos per un cost inferior a 9$[7]. Fins i tot pots contruir-te el terminal tu mateix.[8]

En principi un Arduino és un microcontrolador Atmel de 8 bits amb diferents components que permeten la integració en altres circuits o dispositius. Normalment els models més senzills disposen de varis pins on fer la comunicació amb hardware extern tal com serien amb amb sensors o dispositius de comunicació inal·lambrica.

L'ús més comú de l'Arduino és dins del sector de la investigació i la pedagogia, donat que permet aprendre a programar microcontroladors per un cost molt baix. A més a més, gràcies al gran ús que té han aparegut moltes llibreries que permet fer prototipatge en projectes més complexos. L'Arduino com a plataforma està orientada a sensors i xarxes de sensors distribuits, per tant intenta buscar un baix consum energetic contrastant amb un baix poder de computació.

Hi ha diferents models, els més importants són els següents:

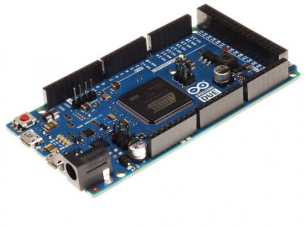
2.1.1.1 Arduino UNO

(A) Arduino UNO

Processador	ATmega328
Característiques CPU	8 Bits, 16MHz
Emmagatzematge	2KB RAM, 32KB Flash MEM
I/O	14 Digital Pins, 6 Analogics
Preu	30$

És l'Arduino més comú pels principiants, té un preu ajustat i és la plataforma on es pot trobar més informació. Com a avantatge, permèt la possibilitat de canviar el processador ATmega328.

2.1.1.2 Arduino Due

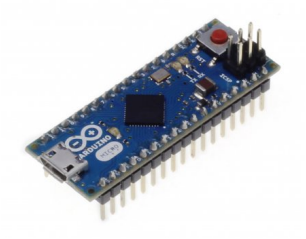


(A) Arduino Due

Processador	Atmel SAM3X8E ARM Cortex-M3
Característiques CPU	32 Bits, 84 MHz
Emmagatzematge	96KB SRAM, 512KB Flash MEM
I/O	54 Digital Pins, 14 Analogics
Preu	50$

És una nova versió de l'Arduino que com a característica principal té la inclusió d'un processador ARM de 32 bits. Això permet tenir una capacitat més gran de càlcul que l'Arduino UNO però també comporta un major consum de bateria.

2.1.1.3 Arduino Micro



(A) Arduino Micro

Processador	ATmega328
Característiques CPU	8 Bits, 16MHz
Emmagatzematge	2.5 KB RAM, 32KB Flash MEM
I/O	20 Digital Pins, 12 Analogics
Preu	27$

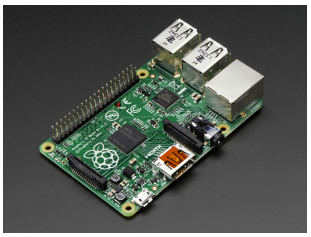
És un Arduino UNO que es caracteritza perquè la CPU està integrada dins de la placa, això permèt una major miniaturització i 0.5KB més de RAM[9].

2.1.2 Raspberry Pi

Tot i que en molts llocs és conegut com un microcontrolador, en realitat es tracta d'un ordinador de placa reduida o conegut com SBC. Inicialment, el disseny es basava en un microcontrolador Atmel ATmega644[10] tot i que finalment es va decidir per un chipset de la plataforma ARM. La plataforma va ser creada l'any 2006 dissenyada per professors i acadèmics amb l'intenció d'animar als més joves a aprendre sobre el món de l'informàtica. Igual que en el cas d'Arduino, Raspberry Pi va néixer com una plataforma oberta amb totes les especificacions i manuals per la producció del SBC disponibles de forma gratuïta. Qualsevol sigui empresa o usuari que tingui els suficients coneixements pot produir les seves pròpies plaques i vendre-les sense pagar cap tipuis de royaltie a Raspberry. A nivell de Hardware existeixen dos models amb millors característiques que l'altre. La plataforma consta de ports USB, un port Ethernet i diferents PINS de forma similar a Arduino. La plataforma permet instal·lar diferents sistemes operatius tant basats en UNIX com RASPBIAN o PIDORA, o en altres arquitectures com RISC OS.

2.1.3 Galileo

Davant de l'éxit de les plataformes de microcontroladors, Intel ha decidit treure un dispositiu que reculli el millor d'Arduino i el millor de Raspberry Pi, i per tant, obtenir una segmentació de mercat major que els productes en què es basa. Per aconseguir això,



(A) Raspberry Pi

Processador	ARM 1176JZF-S
Característiques CPU	32 Bits, 700MHz
Emmagatzematge	2.5 KB RAM, 32KB Flash MEM
I/O	8 GPIO
Preu	50$
Consum	700 mA

ha dissenyat una plataforma que funciona sota arquitectura x86 però que és totalment compatible amb la SDK d'Arduino. Al funcionar sobre x86 aquesta plataforma alhora també permet instal·lar sistemes operatius que consumeixin pocs recursos com sistemes operatius basats en UNIX. Encara que també permet correr el seu propi codi d'Intel basat en el projecte Yocto[11]. Per tant es podría resumir com un dispositu 3 en 1. En principi l'Intel Galileo està destinat al món de l'educació i investigació, però al ser un projecte molt nou les seves eines fan que tingui una corba de dificultat elevada per un estudiant i també es reporten varis problemes de fiabilitat.

(A) Intel Galileo

Processador	Intel Quark X1000
Característiques CPU	32 Bits, 400MHz
Emmagatzematge	256 MiB SDRAM, Targeta SD
I/O	10 Digital Pins
Preu	70$
Consum	700 mA

2.1.4 Telosb

Fins ara s'han descrit opcions més orientades al món de la investigació i l'educació. Dintre de l'àmbit més professional hi ha infinitats de solucions. Per tant, únicament en aquest TFG es centrarà en la plataforma on s'ha treballat.

El TelosB, tot i estar orientat en el món de la recerca, al fer ús d'un sistema operatiu com TinyOS que es fa servir dins del món professional, tots els programes que es fan són fàcilment adaptables a altres plataformes de sensors amb un ús més pràctic.
El TelosB es un mòdul de sensors de baix consum orientat a l'experimentació de xarxes de sensors sense fils. Aconsegueix una alta autonomia gràcies a tenir unes especificacions modestes i a la seva capacitat d'entrar en sleep i consumir molt poca potència durant una gran quantitat de temps. Per això no es un mòdul orientat a fer càlculs complexes, si no a recollir informació de sensors i enviarles a una base mitjançant una xarxa de sensors.

El dispositiu conté 3 leds i una connexió serial mitjançant USB. Com disposa de poca memòria on instal·lar les aplicacions només funciona amb sistemes operatius senzills com seria el TinyOS. Per les comunicacions té una antena que utilitza el protocol IEEE 802.15.4 que li permet transmetre a velocitats de fins a 250 kbps utilitzant la banda 2405 a 2480 Mhz [12].

(A) Telosb

Processador	TI MSP430
Antena	50 kbps, High Data Rate Radio
Emmagatzematge	10kB RAM
I/O	3 Leds

2.2 Xarxes de sensors i monotorització

Antigament quan es tenia un sensor, per exemple de temperatura i humitat, hi havia una feina diària molt importat que era anar a l'estació i apuntar en una llibreta les temperatures màximes i mínimes. Aquest procés es complicava cada cop més com major fos la quantitat de sensors que es tingués.

Per tant amb l'abaratiment dels costos de les tecnologies de la computació i la miniaturització de l'electronica es va veure com la solució per tots els problemes[13]. Els primers d'aquest sensors anaven connectats a ordinadors mitjançant cablejat, on cada X temps es prenien mostres i s'indexaven a una base de dades. Tot i així, aquesta tecnologia tenia la limitació física de què els sensors no eren autònoms i havien d'estar sempre connectats físicament a un ordinador central. Això juntament amb els elevats costos va provocar que aquesta tecnologia fos únicament viable dins de l'entorn professional.

Però a partir del segle XXI amb la popularització de l'Internet i la millora de les tecnologies de comunicació innal·làmbrica va començar a sorgir el concepte de "xarxes de sensors sense fils". On cada sensor estava connectat a un microcontrolador independent, que autònomament s'encarregava d'enviar les dades recollides tant a una base com a un altre sensor que estigués situat a distància formant d'aquesta forma xarxes de sensors intercomunicades entre sí. Juntament això i l'abaratiment d'aquestes tecnologies va fer que els sensors deixessin de ser únicament per ús professional a popularitzar-se per particulars.

2.2.1 Tecnologies de comunicació sense fils

Per establir xarxes de sensors existeixen un gran nombre de tecnologies de comunicació inal·làmbrica, ja que la connexió per medis físics a vegades és impracticable.

2.2.1.1 Xbee

És un mòdul de ràdio que utilitza el protocol estàndard 802.15.4-2003 orientat a les connexions punt-a-punt i en estrella. El principal avantatge del Xbee és que permet establir comunicacions amb un baix consum de corrent (1mW) i a distàncies de fins a 30m. També el protocol permet que es facin xarxes amb xbee's permetent un augment de la distància de comunicació. La placa Xbee en concret també té com a característica diferents PINS I/O on es poden connectar sensors senzills i fer actuar el xbee de forma autònoma sense necessitat d'estar connectat a cap placa microcontroladora.
El seu principal desavantatge és el poc ample de banda que hi ha en les transmissions donat que el protocol 802.15.4-2003 al ser WPAN (reds inal·làmbriques d'Àrea personal) no permet grans transferències en les comunicacions [14]. Un altre tema molt criticat és el alt cost, ja que aleshores la plataforma la fa inviable per propòsits comercials.

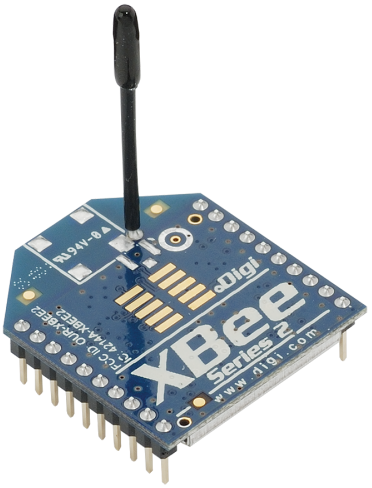


FIGURA 2.7: Xbee

2.2.1.2 Wifi

La tecnologia més comuna de comunicació inal·làmbrica utilitzada avui en dia. Es venen multitud de solucions ja preparades per plaques microcontroladores com Arduino Wifi Shield.

El protocol 802.11ah permetrà velocitats màximes teòriques de fins a 1 o 2 Mbits/s i amb distàncies que depenen de l'antena i la potència d'emissió, també s'ha millorat el consum respecte anteriors tecnologies wifi [15]. El seu avantatge més important en contrast amb altres tecnologies, és l'alta velocitat de transferència de dades. A més, al ser d'ús molt comú es pot trobar a preus molt econòmics.
Com a principal inconvenient, el consum de bateria continua sent elevat, pel que per dispositius amb necessitat de transmetre poques dades no és el sistema més adequat i fins el 2016 no estarà en el mercat.

2.2.1.3 Bluetooth

És una tecnologia comunament utilitzada en dispositius mòbils ja que permet transmetre dades a velocitats acceptables de transferència i en el cas de bluethoot ho permet a distàncies de fins a 1 metre amb un consum de 1mW[16]. Encara que també es venen solucions amb un consum més elevat però que arriben a distàncies de fins a 450m[17]
Per tant es converteix en una solució molt versàtil, donat que permet, com en les xbee's xarxes augmentar l'abast.
Per característiques tècniques com aquestes el converteixen en un dels dispositius de comunicació inal·làmbrica més utilitzats per les plaques microcontroladores

2.2.1.4 Infraroig

És la tecnologia més present a tots el comandaments a distància dels nostres televisors. De les esmentades en l'apartat 2.2.1 és la més senzilla, únicament és necessari un led emissor de llum infraroja i un fotoresistor del mateix tipus de freqüència. Els leds infrarojos es poden aconseguir a preus extremadament econòmics, igual que els fotoreceptor. El consum també és molt poc elevat, com a inconvenient principal està el fet que necessita tenir visió directe amb el fotoreceptor per fer la transmissió i que l'ampla de banda és molt baixa. La distància depèn de la potència del LED.

2.2.1.5 LTE

Permet fer ús de tecnologia mòbil per les comunicacions, ideal per dispositius que estiguin a qualsevol lloc amb cobertura mòbil, amb un ample de banda teòric de fins a 1Gbps/s [18]. Amb preus relativament econòmics la converteixen en la tecnologia més versàtil. Té el desavantatge principal de tenir un consum bastant elevat i l'obligació de pagar mensualitats a una companyia de telefònia ja que és necessari la compra d'una targeta SIM.

2.2.1.6 Resum

FIGURA 2.8: Consum/Ample de Banda

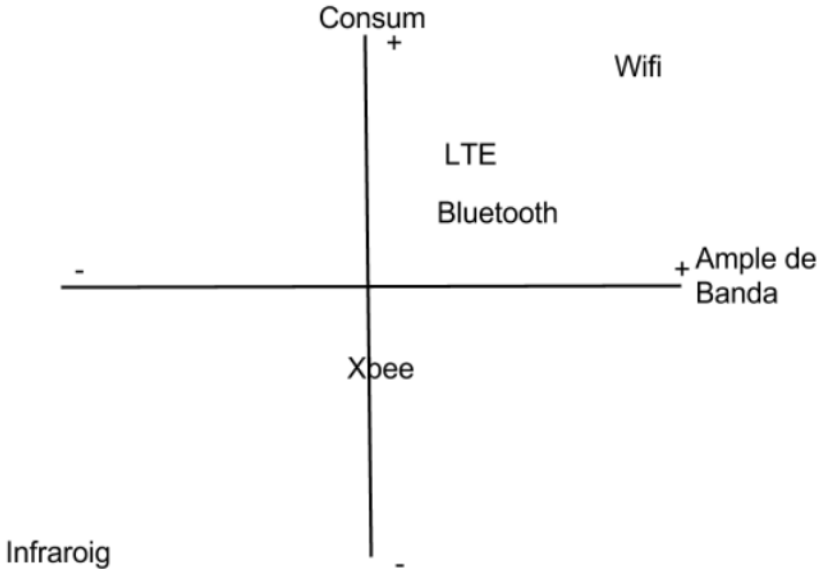
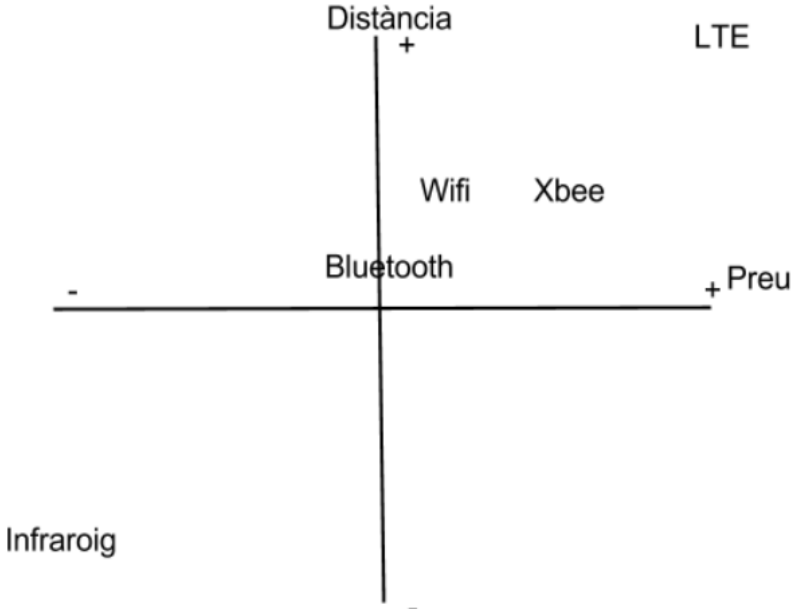


FIGURA 2.9: Preu / Distància



Com es pot observar, per decidir quina és la tecnologia més adequada per la comunicació inal·làmbrica d'un projecte serà necessari fer un balanç de la necessitat de distància de transmissió, el preu del dispositiu, la velocitat de transmissió i el consum energètic.

2.2.2 Tecnologies de monotorització

Fins ara s'ha tractat la placa amb el microcontrolador i la comunicació inal·làmbrica. Totes les dades preses pel sensor estan temporalment emmagatzemades dintre del chipset de RAM del microcontrolador. Però degut a la petita mida del xip de ROM no permet l'emmagatzematge de grans quantitats de dades ni tampoc una facilitat en visualitzar-les d'una forma correcte. Per tant ens veiem la necessitat de tenir en un servidor una plataforma on poder emmagatzemar les dades de forma eficient i estructurada. Per exemple, fent servir sistemes de base de dades i també permetre la visualització correcte d'elles utilitzant tecnologia com HTML5 i CSS.
Tot i així, hi ha vàries solucions professionals i lliures per motoritzar sensors com les següents:

2.2.2.1 Wirelless Sensor Tag

És una plataforma de pagament per monotorització de sensors, tot i que està orientat a sensors que venen juntament amb el producte, també proporcionen una completa API on et permet adaptar-ho per diferents microcontorladors. Com avantatges, permet la visualització dels sensors tant des d'un entorn web, com des de plataformes mòbils tant iOS com Android.[19]

2.2.2.2 Smart Citizen

És un projecte de software lliure densenvolupat per FabLab, amb l'idea de dissenyar una plataforma basada en arduinos on poder monotoritzar sensors de forma fàcil, barata i senzilla. Aquesta monotorització s'integraria en un futur en una espècie de xarxa social. Actualment es poden veure els dispositius de prova en un mapa global. Tot i que en un principi està orientat exclusivament pel seu dispositiu, també proporcionen una API per facilitar l'integració en qualsevol dispositiu.[20]

2.2.2.3 Open-Source Sensor Data Collection Framework (SDCF)

És un framework per monotorització, té com a principal avantatge que és open-source i altament configurable per un gran número de sensors. Un altre punt fort de ser codi obert, és la facilitat de fer modificacions en el codi per tenir una millor adaptació i personalització a la xarxa de monotorització que vulguis crear.[21]

2.3 Protocols d'enrutament per a xarxes de sensors sense fils

En aquest apartat s'analitzaran els diferents protocols d'enrutament per a xarxes de sensors sense fils (WSN). D'aquesta manera es justificarà la necessitat que hi ha de desenvolupar aquesta investigació [22].

2.3.1 Location-based Protocols

Els protocols basats en la localització (a partir d'ara LBP) els nodes tenen com adreça la seva posició en l'espai, d'aquesta forma es pot estimar la distància entre dos nodes i per tant es pot fer una estimació bastant precisa de l'energia necessària per fer la transmissió. Exemples de LBP tindriem MECN, SMECN, GAF, GEAR, Span, TBF, BVGF, GeRaF.

Per aconseguir estimar la posició en l'espai alguns protocols es fa ús del GPS com el GEAR, o de mètodes més complexos en el BVGF com el Voronoi diagram [23].

2.3.2 Data Centric Protocols

Els protocols de dades centralitzades (a partir d'ara DCP) tots els sensors envien els paquets de dades a la base independentment dels altres sensors de la xarxa. Tot i així, alguns dels protocols permeten que els sensors vagin agregant dades originades des de múltiples sensors i d'aquesta manera aconseguir un important estalvi d'energia. Exemples de DCP tindríem SPIN, Directed Diffusion, Rumor Routing, COUGAR, ACQUIRE, EAD, Information-Directed Routing, Gradient-Based Routing, Energy-aware Routing, Information-Directed Routing, Quorum-Based Information Dissemination, Home Agent Based Information Dissemination.

Per exemple, el protocol SPIN fa negociar els sensors entre ells per estalviar-se l'enviament de dades rendundants i d'aquesta manera estalviar energia. O per altra banda, EAD intenta establir un "spaning-tree"entre ells i aconseguir ser més eficients energeticament.

En aquest projecte el protocol que s'adapta estaría proper als Data Centric Protocols ja que l'enrutament funciona similar. Pero s'incorpora una fase de negociació entre nodes per establir la potència mínima per la seva comunicació.

2.3.3 Hierarchical Protocols

Els protocols jeràrquics (a partir d'ara HCP) intentan establir çlusters"o agrupacions de sensors, ja que l'enrutament de sensors divindint-los en agrupacions és una manera molt eficient energeticament d'enrutament [24]. Exemples de HCP tindríem LEACH, PEGASIS, HEED, TEEN, APTEEN.

El protocol HCP més popular és el LEACH, ja que està orientat a l'estalvi d'energètic i com a particularitat, fa ús de l'agregació igual que els Data Centric Protocols. Els altres protocols com HEED o PEGASIS són simplement extensions del LEACH on s'afegeixen noves característiques com densitat de nodes per millorar l'establiment de clusters.

2.3.4 Mobility-based Protocols

Són protocols orientats a sensors que poden estar en moviment (a partir d'ara MBP) ja que aquesta situació afegeix més grau de llibertat en el funcionament d'aquest i per tant, s'han de tenir en compte situacions que no passen en els anteriors protocols. Exemples de MBP tindriem SEAD, TTDD, Joint Mobility and Routing, Data MULES, Dynamic Proxy Tree-Base Data Dissemination.

Aquests protocols intenten solucionar problemes típics en sensors amb mobilitat com el energy sink-hole problem [25] que provoca un gran consum d'energia dels sensors més propers a la base, un dels que soluciona aquest problema sería el Joint Mobility and Routing protocol.

2.3.5 Multipath-based Protocols

Aquests protocols d'enrutament multicamí (a partir d'ara MpBP) sempre intenten connectar-se amb el veí més proper per enrutar-se i transmetre informació cap a la gateway. Aquest tipus de protocols es basen amb l'idea de que establir camins curts amb un cost energètic petit és més eficient que establir un sol camí llarg amb un cost elevat. Exemples de MpBP tindríem Sensor-Disjoint Multipath, Braided Multipath, N-to-1 Multipath Discovery.

Aquests procotols tenen com a característica principal que sempre intenten trobar els camins amb menys latència i per sensors que no tinguin veíns en comú. Com sería en el cas del Sensor-Disjoint Multipath.

2.3.6 Heterogeneity-based Protocols

En una xarxa de sensors heterogènia (a partir d'ara HBP) hi ha dos tipus de sensors, els que funcionen amb bateria i els que estan connectats a una font d'energia conntectada per cable. Per tant la forma de funcionar dels nodes amb bateria i dels nodes amb cable és completament diferent. Exemples de HBP tindriem IDSQ, CADR, CHR.

Una forma d'afrontar aquest problema, és com en el protocol CADR, on els sensors que funcionen amb bateria (L-sensors) intenten establir connexions a distàncies curtes amb un cost energètic baix i els sensors connectats per cable (H-sensors) intenten establir connexions de llarga distància donat que no és una preocupació el cost energètic.

2.3.7 QoS-based Protocols

Aquests protocols barregen funcions dels anteriors però amb l'objectiu de tenir en compte la qualitat del servei (QoS). Per tant, intenten obtenir un balanç entre eficiència energètica i una QoS. Exemples de QoS-based protocols tindríem SAR, SPEED, Energy-aware routing.

Un exemple d'aquests tipus de protocols tindríem el SAR que intenta tenir un sistema de prioritats de paquets per d'aquesta forma garantir la transmissió de determinats paquets que poden tenir certa rellevància.

Capítol 3

TinyOS

Fins ara s'han descrit totes les característiques necessàries per tenir un sistema sensor-microcontrolador-monotorització.
En sistemes com Arduino, l'usuari final simplement haurà d'escriure un codi en el compilador proporcionat per l'empresa i instal·lar els binaris dins de la ROM del microcontrolador i el programa funcionarà sense problemes. Això succeïx perquè amb sistemes d'aprenentatge senzill per l'usuari ve un sistema operatiu simple pre-instal·lat dins de la placa microcontroladora que s'encarrega entre altres coses de gestionar threads i feines de escriptura i lectura (I/O).
Tot i així en plaques més professionals com el TelosB que es la placa que s'utilitza en aquest treball ve sense cap mena de sistema operatiu instal·lat.
Això es deu que al ser plaques microcontroladores més professionals s'intenta que el desenvolupador apliqui un sistema operatiu adaptat a les necessitats del producte final, i d'aquesta forma fer-ho més eficient depenent la tasca que estigui destinat. En el cas del TelosB utilitzaré el TinyOS ja que es un sistema operatiu amb un gran quantitat de suport i documentació per aquesta placa i amb un molt baix consum de bateria ja que l'objectiu del projecte es minimitzar el consum. [26]

3.1 Què és? Característiques generals

TinyOS es un sistema operatiu dissenyat per xarxes de sensors sense fils. L'objectiu principal de TinyOS es fer una abstracció d'alt nivell del software per d'aquesta forma permetre la multiplataforma.
Programar a alt nivell té com a avantatges que redueix els temps de desenvolupament del codi al reduir la quantitat de línies necessàries per fer un mateix propòsit d'un programa i també facilita la escalabilitat d'un codi. Tots aquests avantatges el fan un

sistema operatiu atractiu a nivell empresarial ja que resulta en una reducció dels costos i temps de desenvolupament. També una de les particularitats de TinyOS es que està desenvolupat per fer un ús molt eficient de memòria RAM i ROM[27]. D'aquesta forma deixar al programari la màxima memòria possible per el seu funcionament.
Un altre característica es que TinyOS fa servir la llicència BSD[28], aquesta permet la lliure disponibilitat del codi per a qualsevol tipus de propòsit comercial ja que permet legalment no alliberar el codi font com també imposar en cas de fork la teva pròpia llicencia[29].

Un punt molt important del sistema operatiu es que de sèrie suporta un conjunt de mòduls inal·lambrics, per tant únicament fent servir la llibreria estàndard que ve ja instal·lada seria suficient per fer comunicacions inal·lambriques, sense la necessitat d'adaptar ni compilar cap driver concret. Això permet que un codi per TelosB es pugui utilitzar sense fer cap mena de canvi en un MicaZ o qualsevol microcontrolador que tingui una versió de TinyOS.

Un dels desavantatges principals es el llenguatge de programació, ja que tot i ser d'alt nivell, es un llenguatge poc usual i fa que la corba d'aprenentatge dels usuaris novells sigui elevada. Per tant el fa un sistema orientat a usuaris amb un nivell avançat de programació. Aquest llenguatge es descriurà amb major profunditat en el apartat 3.3 .

Com TinyOS està dissenyat com un sistema operatiu amb poc consum d'energía, permet entre d'altres coses entrar en mode sleep, que seria un mode de baix consum d'energia per despertar-se puntualment quan fos necessari fer algun càlcul concret.
Per tant si es necessari fer un ús continuo computacional pot donar problemes tant de consum energètic, com de malfuncionament del codi. La documentació recomana intentar fer els càlculs per blocs i espaiats temporalment.

3.2 Arquitectura

TinyOS es un sistema operatiu orientat a sistemes incrustats, d'aquesta forma té una arquitectura molt particular. Totes les plaques microcontroladores, per qüestions de mida, consum i preu pateixen de tenir poca quantitat de RAM i emmagatzematge. Característica que fa completament incompatible amb un sistema operatiu d'ús general com seria UNIX. Tot i així tampoc ha de tenir totes les característiques que té un sistema operatiu tradicional ja que les necessitats d'ús d'un dispositiu incrustat no té tants requeriments com un ordinador de sobretaula. També es necessari com s'ha descrit en el punt 3.1 que el sistema operatiu estigui adaptat a tenir un baix consum de energia.

L'arquitectura d'un sistema operatiu estàndard seguiria les següents característiques:

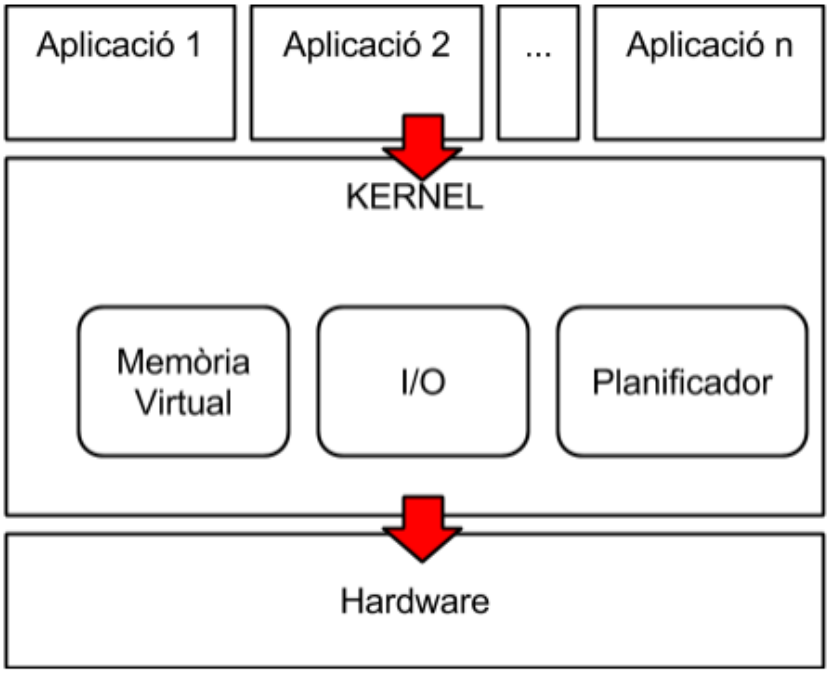


FIGURA 3.1: Arquitectura d'un sistema operatiu Típic

On tindríem 3 capes, primer les aplicacions que per executar-les accedirien en el kernel, que s'encarregaria de fer totes les feines de escritura i lectura (I/O), també s'encarregaria de repartir el temps de computació entre totes les aplicacions que funcionen alhora en el sistema operatiu (si es multi-procés) i memòria virtual. Aleshores el Kernel faría aquesta comunicació amb el hardware que es on es processarien totes les dades[30]. En el cas de un sistema operatiu per dispositius de baix consum i potència moltes d'aquestes característiques son inncesesaries com per exemple l'ús de memòria virtual, ja que al tenir una quantitat molt limitada d'enmagatzematge no hi ha espai per ficar també la RAM. O l'ús d'un planificador, ja que treballar en vàries aplicacions a la vegada fa que el programador perdí més el control del hardware i per tant sigui més difícil garantitzar un baix consum energètic. Que un microcontrolador que pugui fer multitasca perd el sentit de ser d'ún microcontrolador ja que com s'ha definit en el punt 2, un microcontrolador té com a objectiu fer tasques petites i especialitzades. Per tant no té sentit tot el cost energètic i en espai en la memòria ROM de tenir diferents aplicacions corrent a la vegada. Per necessitats com aquestes hi ha l'ús de threads on es el programador qui té el control i es fa un ús més eficient.

Aquestes característiques no necessaries en un sistema operatiu de baix consum van fer que es desenvolupes la següent arquitectura (figura 3.2) . S'ha decidit prescindir d'un kernel, el programador té accés directe a nivell de hardware, però tot i així el llenguatge de programació necessari es a alt nivell. Això es degut a fer ús d'un llenguatge de programació orientat a components com seria el nesC. Aquest punt s'explica amb més

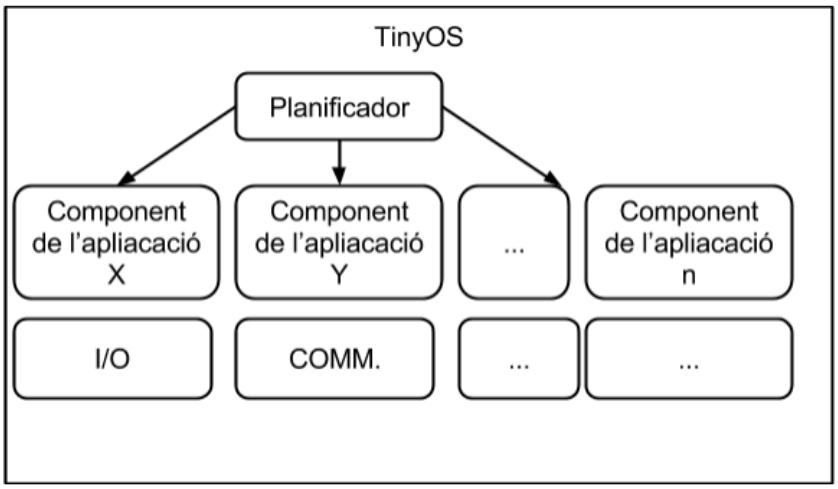


FIGURA 3.2: Arquitectura TinyOS

profunditat en el punt 3.3. Una altra característica es que TinyOS es un sistema operatiu que només permet un procés. Per tant d'aquesta forma s'aconsegueix un major control del hardware i un menor consum energètic. També s'ha eliminat la memòria virtual degut al limitat espai de la memòria ROM, fent que el programador treballi directament sobre els punters de memòria. Degut a l'exclusió de la multitasca TinyOS permet treballar en "tasks" de forma molt avançada per d'aquesta forma suplir la falta de RAM.

3.3 Com es programa

Com s'ha explicat breument en el punt 3.1 el TinyOS amb l'objectiu de buscar un gran rendiment i eficiència s'utilitza un llenguatge de programació orientat a components basat en C que es diu NesC.

3.3.1 Llenguatge orientat a components

Tota computació basada en arquitectura de von Newman funciona a llenguatge màquina, aquest llenguatge es molt lent d'escriure'l i es poc escalable. Per això existeixen abstraccions del llenguatge màquina que a partir d'una sintaxis més senzilla per el programador, permet facilitar el desenvolupament d'aplicacions.

Aquesta llenguatge com més abstracte es faci respecte el llenguatge màquina es diu que va pujant de nivells (levels). Per exemple un dels llenguatges més comunament utilitzats en el món de la programació i dels que més anys porta en actiu es el C. Aquest llenguatge es a baix nivell; per tant poc abstracte i més proper al llenguatge màquina. També es un llenguatge orientat a funcions, ja que el codi es fragmenta en porcions

individuals que tenen entrades i una sortida. Amb el temps van néixer nous paradigmes de la programació com l'orientada a objectes que permetia una abstracció encara més elevada afegint facilitat de programar i escalabilitat en el codi. Llenguatges com el C++ o Java fan ús d'aquest paradigma. Una d'aquests paradigmes de la programació es el llenguatge orientat a components, que es una abstracció semblant al paradigma orientat a objectes[31].
La programació orientada a components es similar a la programació orientada a objectes (a partir d'ara POO), està basada en la noció de "component" i està dissenyada amb els propòsits de solventar els defectes en la reutilització d'objectes que pateix la POO i la falta d'empaquetament i mercadotècnia.

Un component seria un empaquetament de funcions que actuen de forma individual respecte altres parts del codi, i es comunica asincronament. La comunicació es fa mitjançant l'ús de interfícies. Estan dissenyades de tal forma que per el programador sigui transparent tot el codi intern i s'adapti a altres components perquè pugui encaixar. Un altre fet important dels components es que han de ser substituïbles, un component que compleixi X requisits ha de poder ser substituïble per un altre component que també compleixi X. I tot el programa continuï funcionant sense problemes.

3.4 NesC

NesC es el llenguatge orientat a components que utilitza el TinyOS. Es va escollir aquest llenguatge ja que al ser un llenguatge orientat a components era apropiat per el funcionament de l'arquitectura del TinyOS. NesC es un llenguatge que utilitza la mateixa sintaxi que el C amb algunes particularitats en l'estructura del codi.

Per exemple l'estructura d'un component senzill per el control de un LED seria la següent:

```
1  configuration Led {
2    provides {
3      interface LedControl;
4    }
5     uses {
6      interface Gpio;
7    }
8  }
9  implementation {
10
11   command void LedControl.turnOn() {
12     call Gpio.set();
13   }
```

```
14
15    command void LedControl.turnOff() {
16      call Gpio.clear();
17    }
18  }
```

Índex de codis 3.1: nesC Component example

On es pot observar 2 parts; Configuration i Implementation:

- Configuration: A configuration s'especifiquen els components que es faran servir i els components que aportarem. Les entrades i sortides s'especifiquen amb "interface" que seria el canal de comunicació bidireccional entre components.

  En el cas del nostre codi en "provides"especifiquem quina es la interface que aporta el nostre programa i en üsesl·l'interfície que fa servir per comunicar-se. Com en aquest codi estem fent un component de control de LED fem servir la interface del Gpio que ens permet tenir control amb el PIN on està connectat el LED i aportem una interface anomenada LedControl per poder accedir des de fora.

- Implementation: Aquí s'implementen totes les funcions necessàries per el programa, en el cas del control del led, tenim dos funcions una per apagar i l'altre per encendre.

  Com s'observa aquest llenguatge té una estructura molt definida que permet que el codi sigui compatible amb altres components i alhora obliga a fer per el programador ús de bones pràctiques de programació. Cosa que fa que el codi sigui escalable.

  En el cas que es vulgui programar una aplicació simplement s'ha de cridar Main.C-Boot perquè el TinyOS executi sempre aquest component al arrancar.

Capítol 4

Protocol i implementació

Com s'ha resumit en l'intruducció l'objectiu d'aquest TFG es implementar un protocol d'enrutament dissenyat per Toni Adame Vázquez per a Xarxes de Sensors sense Fils. Aquest protocol busca com a principal objectiu reduir el consum energètic de la ràdio ja que es el component que més energía gasten normalment en els microcontroladors i que per tant més s'ha de vigilar el consum[32].
La ràdio té tres estats, quan es un emissor, quan es un receptor i quan està en IDLE. El protocol està enfocat en treballar d'optimitzar l'estat d'emissió. Per aconseguir aquest pròposit es buscarà transmetre els paquets en la mínima energia possible.

Per aconseguir aquest pròposit s'han buscat una serie d'objectius a complir:

1. L'enllaç entre dos terminals sempre serà amb el mínim d'energia possible:
   El TelosB pot establir conexions a distàncies de fins a 112m quan no hi ha cap obstacle [33], per tant es un cost inútil establir una connexió amb l'energía per defecte ja que no sempre el terminal estarà a la màxima distància funcional. Per tant un dels objectius del protocol es establir una negociació que permeti establir la connexió amb la mínima energía operacional.

2. S'haurà d'establir una política d'enrutament
   Ja que no sempre serà possible establir una connexió directe amb el gateway, s'haurà de crear un enrutament simple que permeti els paquets saltar de mote a mote per arribar al gateway.

4.1 Funcionament

El funcionament del protocol a nivell lògic es el següent:

Una vegada encès el dispositiu, es conecta al gateway i estableix un intent d'enllaç. En cas de no poder-se connectar directament al gateway intentarà buscar un enllaç alternatiu on establir la connexió i fer de pont cap el gateway. Si tot i així no ha estat possible establir connexió mitjançant l'enllaç alternatiu esperarà un delay per tornar intentar tot el procediment de nou. Una vegada iniciat amb èxit l'intent d'enllaç, el sensor negociarà amb el mote que està connectat la potència mínima possible per aconseguir una transmissió amb èxit. Intentarà enviar un paquet de prova, i si la connexió ha estat establida amb èxit esperarà un timeout per tornar a renegociar la potència mínima de nou (Delay A). Això es fa per què en cas que el dispositiu s'hagi desplaçat es pugui renegociar la nova potència necessària.

Per tant podríem definir el nostre protocol com un Data Centric Protocols ja que els nodes sempre intentarán establir connexió amb el gateway i en cas que no sigui possible, s'enrutarien amb altres nodes de la xarxa.

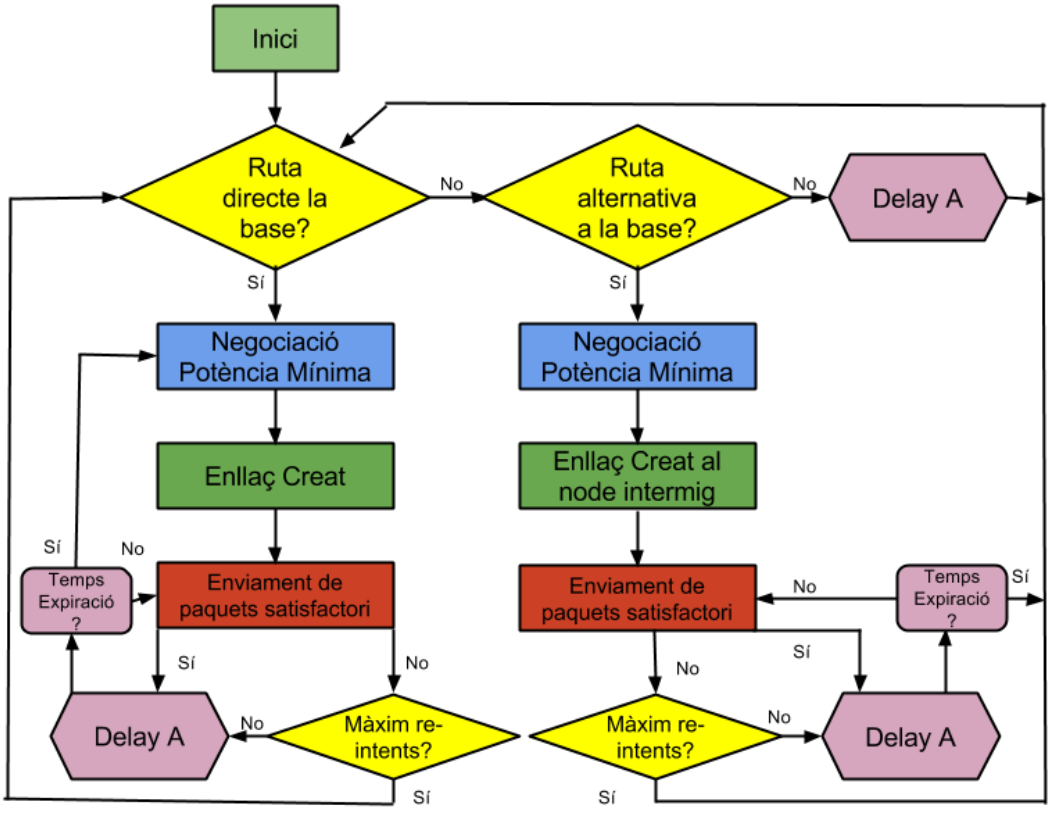


FIGURA 4.1: Diagrama seqüencial

Per tant podem dir que el protocol consta de 3 fases:

- Fase d'establiment de connexió.
- Fase de negociació de potència mínima.
- Fase de manteniment

4.2 Estructura Paquet

Els paquets que farà servir el nostre protocol d'enrutament tindran una mida màxima de 64 bits en les capçaleres, ja que com més petits siguin els paquets que es transmeten menys possibilitat d'error hi haurà durant la transmissió i per tant menys energia costarà. També tenen un pes reduït ja que la simplificació de l'estructura permet un menor consum energètic al ser necessari un menor temps de transmissió per paquet.
L'estructura dels paquets es la següent:

Bit offset	**0-16 Bits**	**16-32 Bits**
0	Origen	Destí
32	id	Tipus
64	Opcional1	Opcional2

Taula 4.1: Estructura del paquet

1. El primer camp es l'origen, un nombre enter de 16 bits que indica el mote origen d'on ve la transmissió.
2. El segon camp es el destí, també de 16 bits on indica quin es el mote destí on va la transmissió, en cas de ser un broadcast la transmissió la rebrien tots els motes que estiguessin en cobertura.
3. Id es un camp on cada paquet té un número únic per garantir que no hi hauràn duplicitats durant la transmissió.
4. Tipus es un camp de 16 bits que serveix per indicar quin tipus de paquet es, tenim els següents:
   (a) SYN: Es un paquet per la fase de Establiment d'enllaç, es el paquet que s'envia en busca de la Gateway.
   (b) SYN-ACK: Es el paquet de resposta per establir l'enllaç.

(c) PWRN: Es el paquet que s'encarrega de negociar la potència de la senyal del node, aquest paquet conté un paràmetre Opcional amb el valor de la potència emesa de la senyal.

(d) PWRN-ACK: El paquet de resposta per establir la potència de la senyal, aquest paquet conté un paramtre opcional amb el valor de la potència que ha de utilitzar per l'emisió.

(e) PWRN-OK: Paquet de resposta que indica que la potència de la senyal es correcte i es pot passar a la seguent fase.

(f) DATA: Es el paquet que es farà servir per transmetre informació per analitzar el funcionament del protocol.

(g) DATA-ACK: Es el paquet de resposta del paquet DATA.

5. Opcional1 es un camp opcional que es fet servir per algun dels tipus de paquets descrits anteriorment.

6. Opcional2 es un camp opcional que es fet servir per algun dels tipus de paquets descrits anteriorment.

En el següent diagrama podem veure el procés a nivell de transmissió de paquets de forma simplificada, com hi ha la primera fase on s'estableix connexió, la segona fase on hi ha la negociació de potència mínima, i la tercera fase on hi ha el manteniment de l'enllaç.

A la figura 4.2 observem que per la transmissió del protocol s'utilitzen els diferents tipus de paquets. Una vegada explicat a grans trets el funcionament del protocol, passaré a profunditzar les accions que succeïxen durant cada fase.

4.3 Fases

Una vegada encès el mote, el que primer farà es detectar si es comporta com un base o si es un dels molts sensors. Una vegada fet això encen la ràdio i passa a la fase 1.

4.3.1 Fase d'establiment de connexió

L'objectiu d'aquesta fase es establir una connexió directe amb la base, en cas de no arribar directament s'intentarà establir la connexió amb algun dels motes veïns.

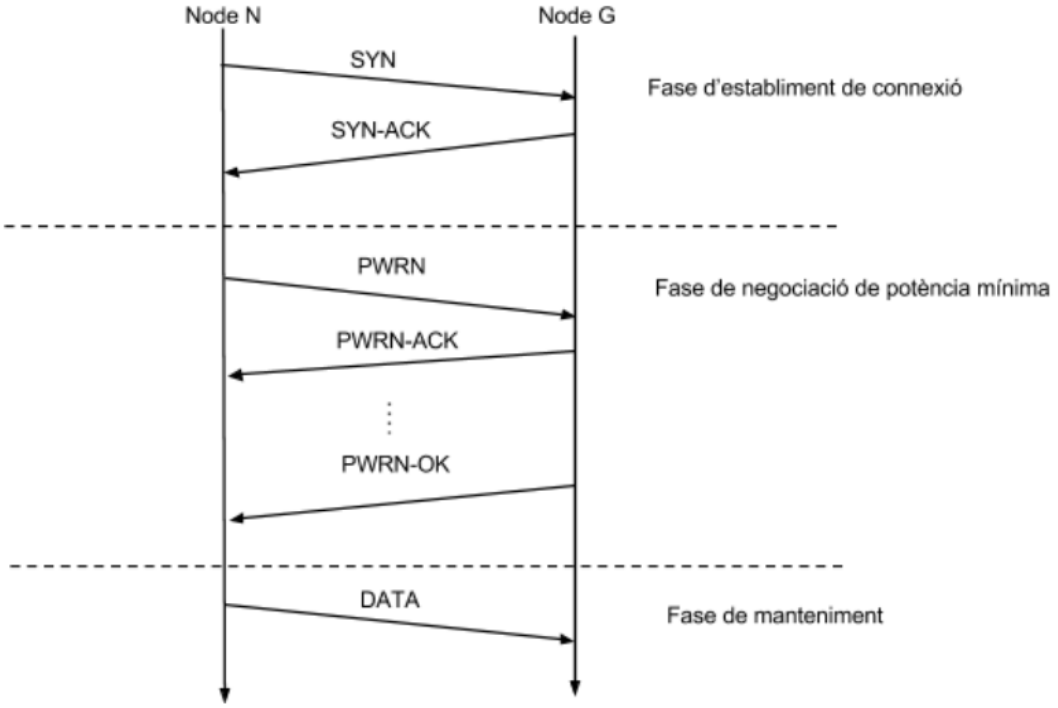


FIGURA 4.2: Diagrama transmisions nivell paquet

4.3.1.1 Funcionament

Una vegada la ràdio del mote ha arrancat envia un paquet en busca d'un gateway on connectar-se.
Aquest paquet es de tipus SYN, i té un pes de 64 Bits. L'adreça de destí sempre serà la del Gateway que per defecte es 0. Taula [4.2].

Bit offset	**0-16 Bits**	**16-32 Bits**
0	Mote-n	Gateway
32	id	SYN

TAULA 4.2: Estructura del paquet SYN

Si aquest paquet arriba al gateway es respòs amb un paquet tipus SYN-ACK que torna al mote inicial. Taula [4.3].

Bit offset	**0-16 Bits**	**16-32 Bits**
0	Gateway	Mote-n
32	id	SYN-ACK

TAULA 4.3: Estructura del paquet SYN-ACK

Una vegada aquest SYN-ACK arriba al mote inicial es passa a la fase de negociació d'energía.

En cas que hi hagi hagut algun error, i no hagi sigut possible establir la connexió amb el gateway. Com per exemple que el paquet SYN o SYN-ACK no hagin arribat el mote enviarà un paquet SYN en mode broadcast a tots els dispositius. Taula [4.4].

Bit offset	**0-16 Bits**	**16-32 Bits**
0	Mote-n	Broadcast
32	id	SYN

TAULA 4.4: Estructura del paquet SYN en Broadcast

Els dispositius que els hi hagi arribat el broadcast, respondran amb un paquet tipus SYN-ACK on el mote origen agafarà el paquet i es connectarà al mote que tingui millor senyal.

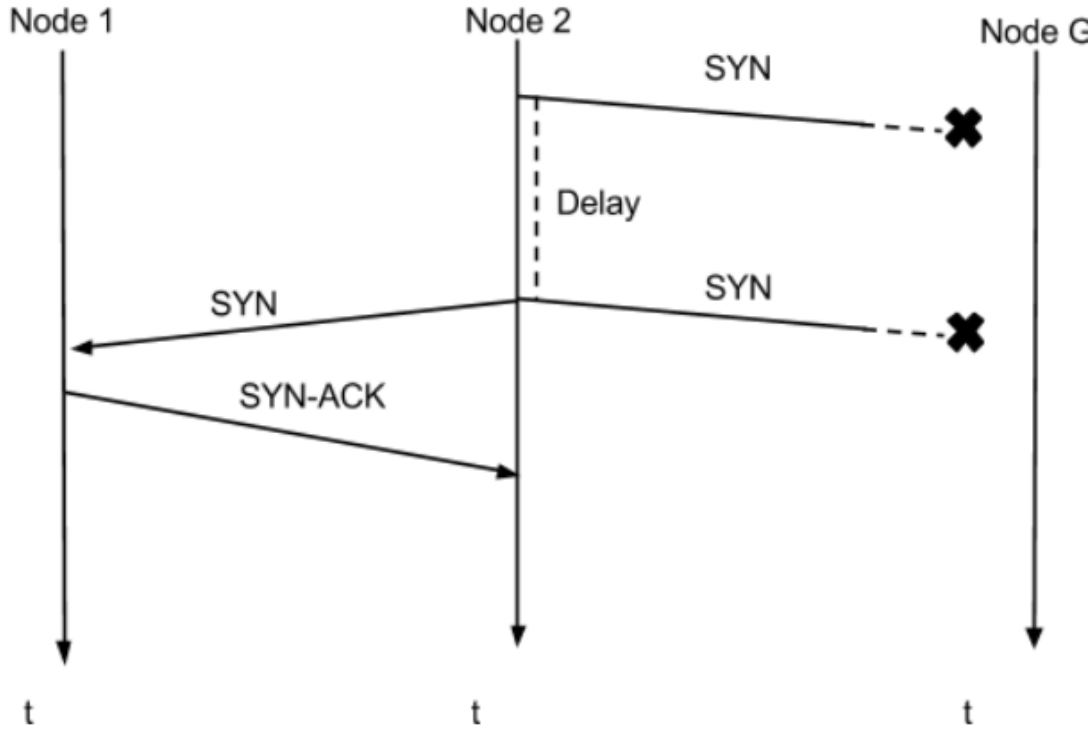


FIGURA 4.3: Diagrama establiment de la connexió a nivell de paquet

Una vegada connectat tant sigui al gateway com en el d'un node secundari es procedirà a la següent fase.

4.3.1.2 Implementació

En aquesta secció es pasarà a explicar com es l'implementació de la part teorica feta en el codi nesC.
El format dels paquets es un struct de tipus nxstruct, ja que ens permet encapsular diferents variables per un missatge de TinyOS.

```
1  typedef nx_struct {
2
3    nx_uint16_t origin;
4    nx_uint16_t destiny;
5    nx_uint16_t id;
6    nx_uint16_t typeP;
7    nx_uint16_t opcional1;
8    nx_uint16_t opcional2;
9
10 } MSGPROTOCOL;
```

Índex de codis 4.1: nesC Estructura paquet

Implementació de la connexió simple bidireccional

Per començar el protocol d'enrutament primer de tot farem una comunicació bidirecional punt a punt amb 2 TelosB.

Figura 4.4: Exemple gràfic connexió bidireccional de dos TelosB

Per fer aquest primer experiment partirem del codi Blink, on simplement es un timer:

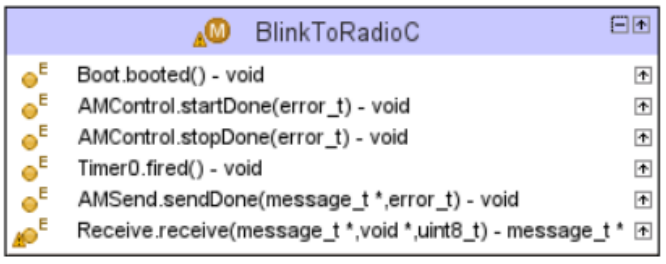


Figura 4.5: Gràfic Components

La comunicació es per broadcast, per tant no s'ha implementat cap protocol d'enrutament. Per el funcionament simplement s'envia un missatge a adreça BROADCAST, que

arriba a tots els nodes. En el moment que es rep un paquet es fan parpadejar els leds. Aquest codi ens permetrà implementar el primer pas del protocol:

Implementació dels Timers

Crearem un Timer que s'executarà si passat el delay no ha sigut respòs el primer paquet SYN al GATEWAY.

```
1  event void NoSyncResponseTimeout.fired() {
2    if(NoSyncResponseTimeoutBool) {
3
4      startErrorSyn();
5    }
6
7    if(NoSyncBResponseTimeoutBool){
8      call AMControl.stop();
9    }
10 }
```

Índex de codis 4.2: nesC Timer delay SYN

Crearem un altre Timer perquè en cas de cap SYN-ACK hagi estat respós, el mote es mantigui amb mínim consum d'energía fins que ho torni a intentar de nou.

```
1  event void WakeUpTime.fired() {
2      call AMControl.start();
3  }
```

Índex de codis 4.3: nesC Timer wake up

Implementació del enviament de paquets

I per enviar els paquets ho fem de la següent manera:

```
1 MSGPROTOCOL * p_pkt = (MSGPROTOCOL * )(call Packet.getPayload(&pkt,
2         sizeof(MSGPROTOCOL)));
3
4     call CC2420Packet.setPower(&pkt, DEFAULT_POWER);
5
6     memcpy(call AMSend.getPayload(&pkt, sizeof(p_pkt)), &p_pkt, sizeof
    p_pkt);
7     id = id + 1;
8     p_pkt->origin = TOS_NODE_ID;
9     p_pkt->destiny = GATEWAY;
10    p_pkt->typeP = SYNC_;
11    p_pkt->id = id;
12
13    if( ! busy) {
14      if(call AMSend.send(AM_BROADCAST_ADDR, &pkt, sizeof(MSGPROTOCOL)) ==
    SUCCESS) {
```

Índex de codis 4.4: nesC Enviar paquets

Primer es crea el paquet en memòria, després li indiquem a la ràdio la poténcia de senyal que ha de tenir el paquet. Omplim els camps del paquet abans del enviament, i comprovem que la ràdio no està ocupada abans de fer el "send".

4.3.2 Fase de negociació de potència mínima

L'objectiu d'aquesta fase es negociar una potència mínima d'emisió entre dos nodes connectats, per d'aquesta forma estalviar energía.

4.3.2.1 Funcionament

Una vegada establerta la connexió s'enviarà un paquet tipus PWRN al node que s'està connectat. Aquest paquet té uns pes de 96 bits. El paquet tipus PWRN té fa servir el camp Opcional1 per transmetre la potència en que s'ha emès la comunicació.

Bit offset	**0-16 Bits**	**16-32 Bits**
0	Mote-n	Mote-enllaç
32	id	PWRN
64	Potència Emesa	-

Taula 4.5: Estructura del paquet PWRN

Una vegada rebut aquest paquet per el mote destí, analitza la potència que s'ha rebut de la senyal, si està per sobre dels marges considerats estables (-72[dBm] a -80[dBm]) s'envia un paquet de resposta indicant que pot baixar la potència emesa. Si observa que la potència està per sota del marge estable s'envia un paquet indicant que ha de d'augmentar la potència. El camp opcional1 es el que conté aquestes dades.

Bit offset	**0-16 Bits**	**16-32 Bits**
0	Mote-enllaç	Mote-n
32	id	PWRN-ACK
64	Potència proposada	-

Taula 4.6: Estructura del paquet PWRN-ACK

Una vegada la potència estigui dintre dels marges estables el mote negociador respondrà amb un paquet tipus PWRN-OK indicant que s'ha assolit la potència estable i ja es pot passar a la següent fase.

Bit offset	**0-16 Bits**	**16-32 Bits**
0	Mote-enllaç	Mote-n
32	id	PWRN-OK

TAULA 4.7: Estructura del paquet PWRN-OK

A la figura 4.6 es pot observar un exemple del intercanvi de paquets típics en la fase de negociació:

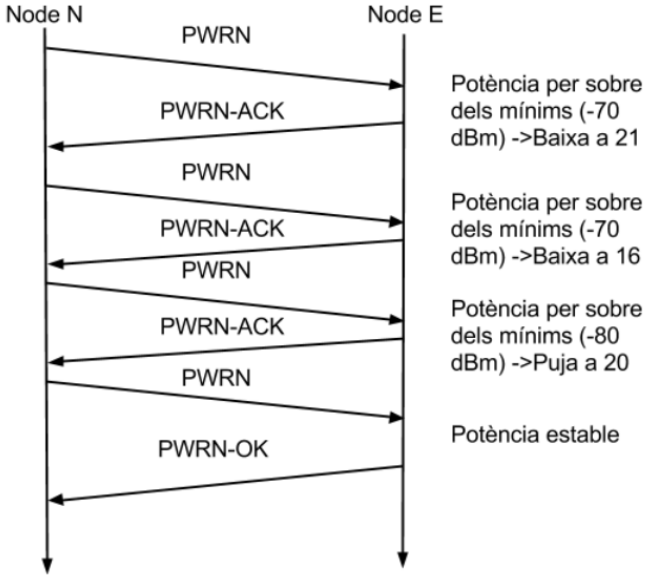


FIGURA 4.6: Diagrama de negociació de potència

4.3.2.2 Implementació

Primer de tot per aconseguir implementar aquesta fase serà necessari utilitzar vàries eines que no ens proporcionen les llibreries estàndards del TinyOS. Aquestes eines són saber la potència amb que es rep una senyal y controlar la potència d'emissió de qualsevol paquet.

Lectura de la potència de la senyal en un TelosB

Per implementar el protocol, necessitem saber en tot moment quina es la potència de les senyals dels nodes que estan connectats. Per poder fer això serà necessari implementar la llibreria "printf-serial" que permet printar informació per consola, i una llibreria que ens permeti treballar amb el xip de ràdio del TelosN a baix nivell. Ja que amb les llibreries de xarxa d'alt nivell no es pot obtenir informació de la potència de la senyal. El chip de ràdio del TelosB es el CC2420P [34], per tant el component necessari per treballar amb

l'energía a baix nivell es el CC2420ActiveMessageC.

També ha sigut necessari fer la conversió en unitats de la potència de la senyal mitjançant aquesta fórmula [35]:

P = RSSI_VAL + RSSI_OFFSET [dBm]
RSSI_OFFSET = -45

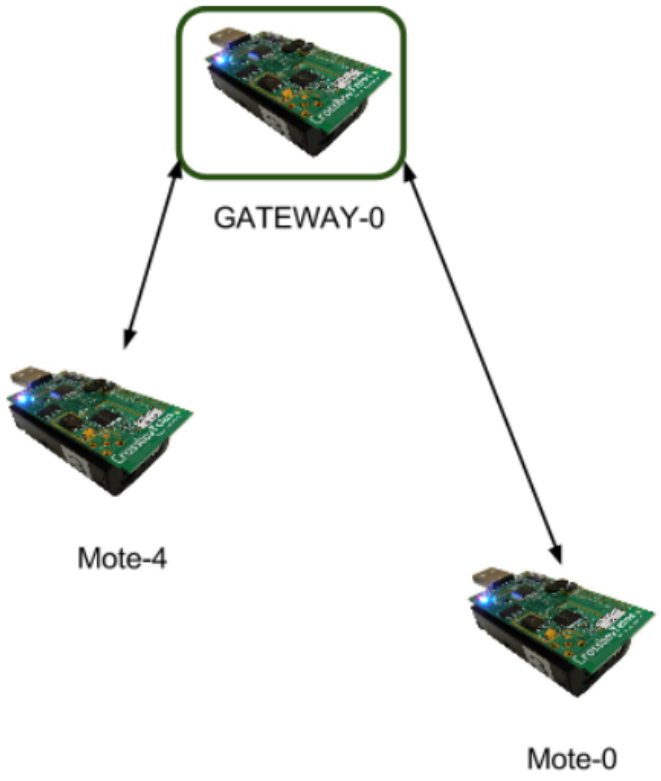
On RSSI_VAL es el valor de la potència de senyal rebuda i el RSSI_OFFSET es un valor obtingut empíricament.

Així s'ha implementat l'obtenció de la potència de la senyal en el codi:

```
1  uint16_t getRssi(message_t * msg) {
2    return(uint16_t) call CC2420Packet.getRssi(msg);
3  }
```

Índex de codis 4.5: nesC getRssi Implementació

Per comprovar que el funcionament es correcte s'han fet una provas. Es connecten 2 motes a diferent distància a un gateway i emeten a igual potència s'ha obtingut el següent resultat:



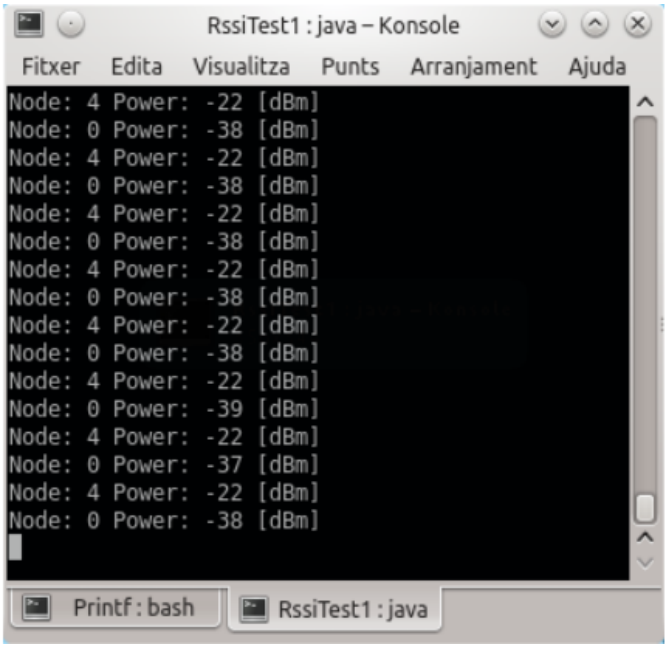


FIGURA 4.7: Comprovació que la potència es detectada correctament

S'observen 2 nodes un amb més potència que l'altre, un resultat que compleix l'expectat.

Control potència senyal d'un TelosB.

El TelosB ofereix una gran quantitat de llibreries per el control de l'energia, ja que es un sistema operatiu enfocat en trobar una alta autonomia. A nivell de TinyOS està la LowPowerListening que permet especificar el interval de temps que la ràdio està encesa. A major temps més probabilitats de que es rebi correctament un paquet. Després a nivell de chip tenim la funció trucada[36]

```
1 call CC2420Packet.setPower(message_t *p_msg, uint8_t power);
```

ÍNDEX DE CODIS 4.6: nesC Control de potència mínima

Que permet especificar la potència d'emissió que tindrà el pròxim paquet. Entre els valors 0 i 33. Els valors d'emissió de potència no tenen unitat ja que simplement signifiquen una divisió entre el màxim de potència que pot emetre el xip (33) i el mínim (0). Per testejar el correcte funcionament a una igual distancia (1 [m] aproximadament), s'han provat els valors de potència de 33, 11 i 3.

Es pot observar com la potència d'emissió del paquet rebut es sempre 0, això es degut que el paràmetre power es part del metadata del paquet, per tant no es transmès quan el paquet es enviat, per tant quan el paquet es rebut aquest valor es el per defecte (0) ja que només es utilitzat per paquets de sortida [37].

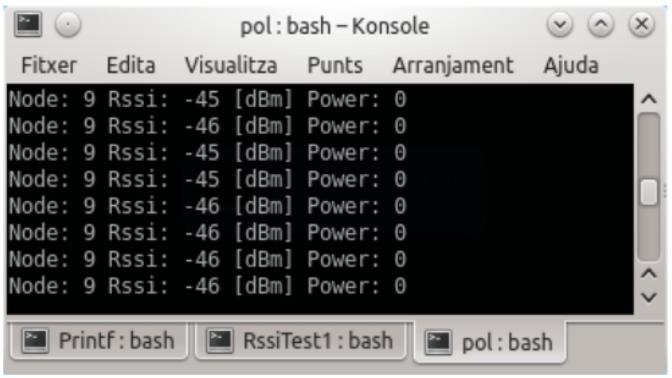


FIGURA 4.8: Potència rebuda emetent a 33

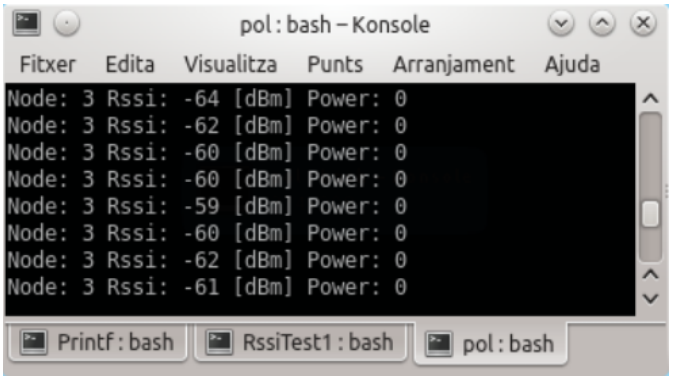


FIGURA 4.9: Potència rebuda emetent a 11

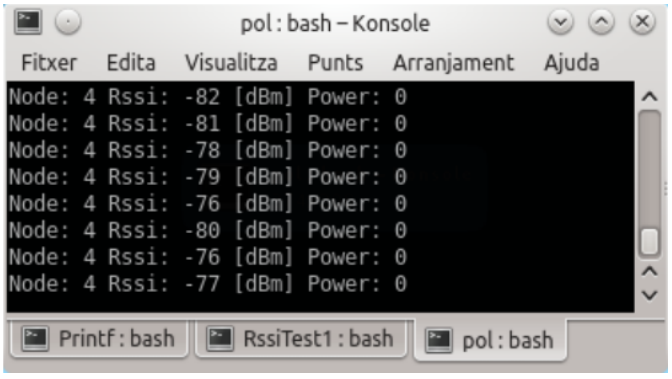


FIGURA 4.10: Potència rebuda emetent a 3

Implementació del enviament de paquets

Per enviar els paquets PWRN ho fem igual que el codi 4.4 pero amb la difèrncia que afegim com a opcional1 la potència en que s'ha emés la senyal.

Implementació algoritme negociació

Per negociar la potència com s'ha explicat en el punt 4.3.1.1 serà una aproximació en forma lineal. O sigui la potència anirà disminuint o augmentant linealment fins assolir la potència estable dintre d'uns marges.

Declarem els marges:

```
#define MAXDBM -72
#define MINDBM -80
#define MINENERGY 5
#define MAXENERGY 31
\label{codi:marges}
```

ÍNDEX DE CODIS 4.7: nesC Marges de potència de recepció i emissió estables

S'ha establert que la potència rebuda que es considera estable es entre -72 [dBm] i -80 [dBm]. I la potència d'emissió acceptable es entre 31 i 5. Això es així ja que 5 considerem que es una potència mínima que permet fer connexions estables

La negociació d'aproximació lineal s'ha implementat de la següent manera:

```
if(p_pkt->typeP == PWRN && p_pkt->destiny == TOS_NODE_ID) {
          u_int16_t rssi = getRssi(msg) - 45;
          u_int16_t powert = p_pkt->opcional1;
          if((rssi > MINDBM && rssi <= MAXDBM) || (powert <= MINENERGY)) {
            if(TOS_NODE_ID == GATEWAY) {
              printf("Negotiation PWR Node %d PWR %d RSSI %d [dBm] PUT POWER: OK\n",
                  p_pkt->origin, p_pkt->opcional1, rssi);
              printfflush();
            }
            sendPWRNOK(p_pkt->origin);
          }
          else
            if(rssi < MINDBM) {
            powert += 2;

            if(powert > MAXENERGY) {

              if(TOS_NODE_ID == GATEWAY) {
                printf("Negotiation PWR Node %d PWR %d RSSI %d [dBm] PUT POWER: OK\n",
                    p_pkt->origin, p_pkt->opcional1, rssi);
                printfflush();
              }
              sendPWRNOK(p_pkt->origin);

            }
            else {
              if(TOS_NODE_ID == GATEWAY) {
                printf("Negotiation PWR Node %d PWR %d RSSI %d [dBm] PUT POWER: %d\n",
                    p_pkt->origin, p_pkt->opcional1, rssi, powert);
                printfflush();
              }
              PWRNACK(p_pkt->origin, powert);
```

```
33              }
34            }
35            else
36              if(rssi > MAXDBM) {
37              powert -= 2;
38              if(TOS_NODE_ID == GATEWAY) {
39                printf("Negotiation PWR Node %d PWR %d RSSI %d [dBm] PUT
    POWER: %d\n",
40                    p_pkt->origin, p_pkt->opcional1, rssi, powert);
41                printfflush();
42              }
43              PWRNACK(p_pkt->origin, powert);
44            }
45          }
```

Índex de codis 4.8: nesC Implemetació de la negociació lineal

On anem disminuint o augmentant linealment la potència de dos en dos.

4.3.3 Fase de manteniment connexió

Aquesta fase es la més senzilla, té tres propòsits, comprovar que s'ha establert la connexió, enviar els missatges debug per saber dades sobre les connexions que s'han establert. I la creació de timers que comprovin cada cert temps si encara es manté la connexió.

4.3.3.1 Teoria

Per debuguejar i comprovar connexions farem servir paquets tipus DATA.

Bit offset	**0-16 Bits**	**16-32 Bits**
0	Mote-n	Mote-enllaç
32	id	DATA
64	RoutingDATA	Connection Check

Taula 4.8: Estructura del paquet SYN en Broadcast

On els camps RoutingDATA guardan una llistat amb tots els motes per on s'ha enrutat el paquet, i ConnectionCheck es un booleà que indica que el paquet DATA serveix per comprovar la connexió. També es necessari indicar que els paquets DATA tenen com a propietat principal que s'enruten entre motes fins arribar a la base. En canvi els altres paquets només treballen a nivell d'enllaç.

4.3.3.2 Implementació

Només haurem d'implementar un mètode que ens permeti enregistar els motes que ha passat un paquet:

```
1 p_pkt->opcional1 = p_pktR->opcional1 * 10 + TOS_NODE_ID;
```

Índex de codis 4.9: nesC Implemetació registre enrutament

I un timer per cada X temps detectar si la connexió encara es manté, en cas que no tornariem de nou a la fase 1 4.3.1.1.

Capítol 5

Validació, problemes i millores

En aquest capítol una vegada ja tenim programat el protocol el provarem intensament per comprovar si de veritat compleix totes les funcions implementades i s'analitzaran els resultats. Després es treballaran tots els problemes i errors que s'han detectat durant les probes per acabar proposant i implementant millores pel protocol.

5.1 Validació del funcionament

En aquest apartat intentarem fer diferents proves per comprovar el correcte funcionament de les tres fases dels TelosB. Primer de tot comprovarem que l'enrutament funcioni correctament portant la situació al límit. I mirant diferents tipus de situacions si el resultat pràctic es comporta a l'estimat en el teòric.
Després analitzarem la negociació d'energía, el temps que es tarda en establir una connexió estable, el número de paquets enviats entre d'altres proves.
En el tercer apartat mirarem l'estabilitat de la connexió una vegada fet la negociació per comprovar si la connexió realment es estable o no.

5.1.1 Proves d'enrutament

Una vegada tot programat, s'ha posat a la pràctica si es feia correctament l'establiment de la connexió. Un dels problemes per comprovar l'enrutament es que els TelosB poden transmetre a grans distàncies, per tant per aconseguir fer enrutaments amb facilitat s'ha tingut que baixar la potència de la senyal al mínim. D'aquesta forma la cobertura es redueix a distàncies de menys de 1.5[m] obligant als terminals a fer servir l'enrutament per arribar a la base. Per culpa d'aquestes modificacions temporals les potències de recepció rebudes poden donar valors fora dels marges de seguretat; -72 [dBm] i -80 .

[dBm]
Primer s'ha col·locat els terminals en forma d'estrella.

5.1.1.1 En estrella

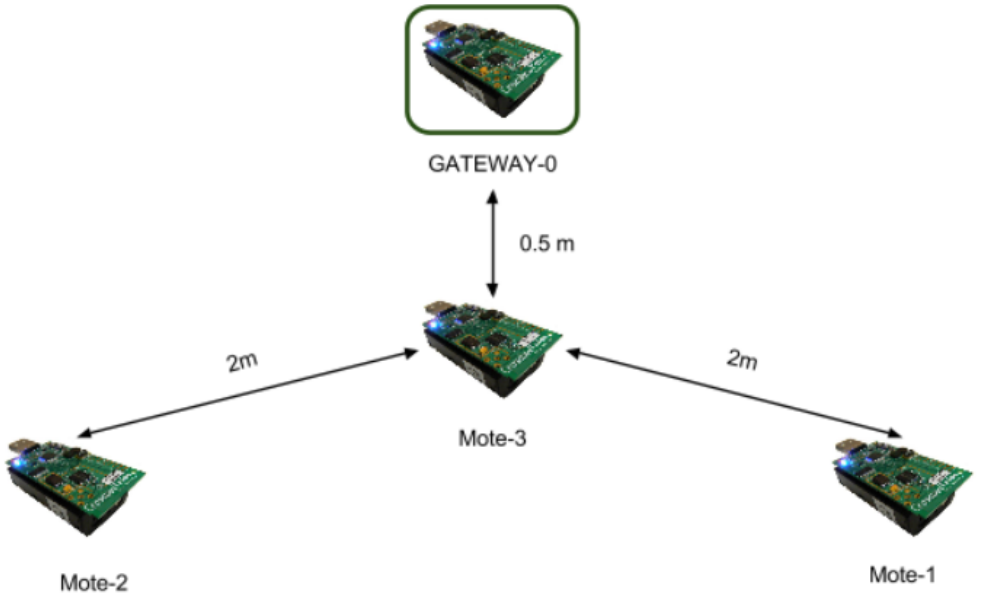


FIGURA 5.1: Col·locació dels terminals en forma d'estrella

Posant-los en aquesta posició obligo als terminals a treballar de tal forma que únicament el mote-3 tingui connexió directe amb el GATEWAY y el mote-2 i el mote-1 no. Per tant els resultats esperats son que els paquets del mote-2 i mote-1 passin sempre pel mote-3.
Comprovem:

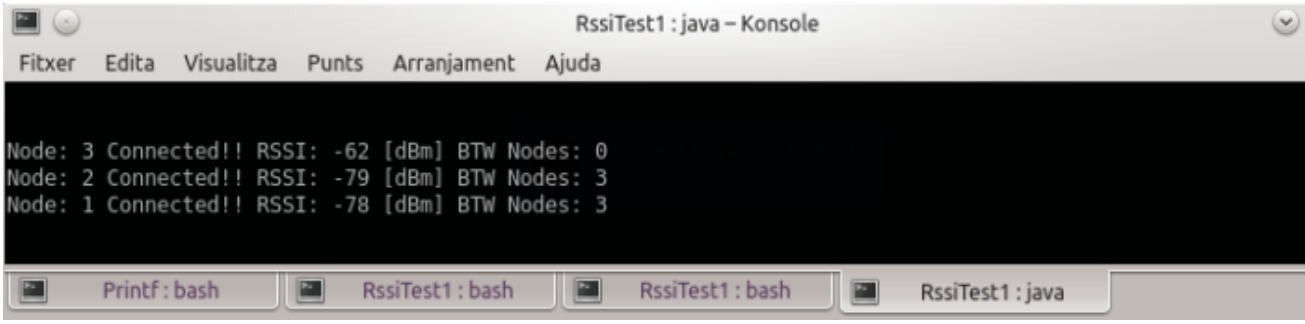


FIGURA 5.2: Comprovació dels terminals en forma d'estrella

Podem observar com el Mote-3 s'ha connectat directament al Gateway-0 tal com mostra "BTW Nodes: 0". També podem observar com el Mote-2 i el Mote-1 s'han connectat

al Mote-3 i d'aquí han saltat cap el Gateway-0.
Amb aquest petit experiment es demostra que l'enrutament almenys funciona amb un salt. El pròxim pas seria comprovar si funciona amb múltiples salts.

5.1.1.2 En bus

S'han col·locat els terminals de tal forma que únicament cada un d'ells pugui veure a 2 veïns. Figura 5.3

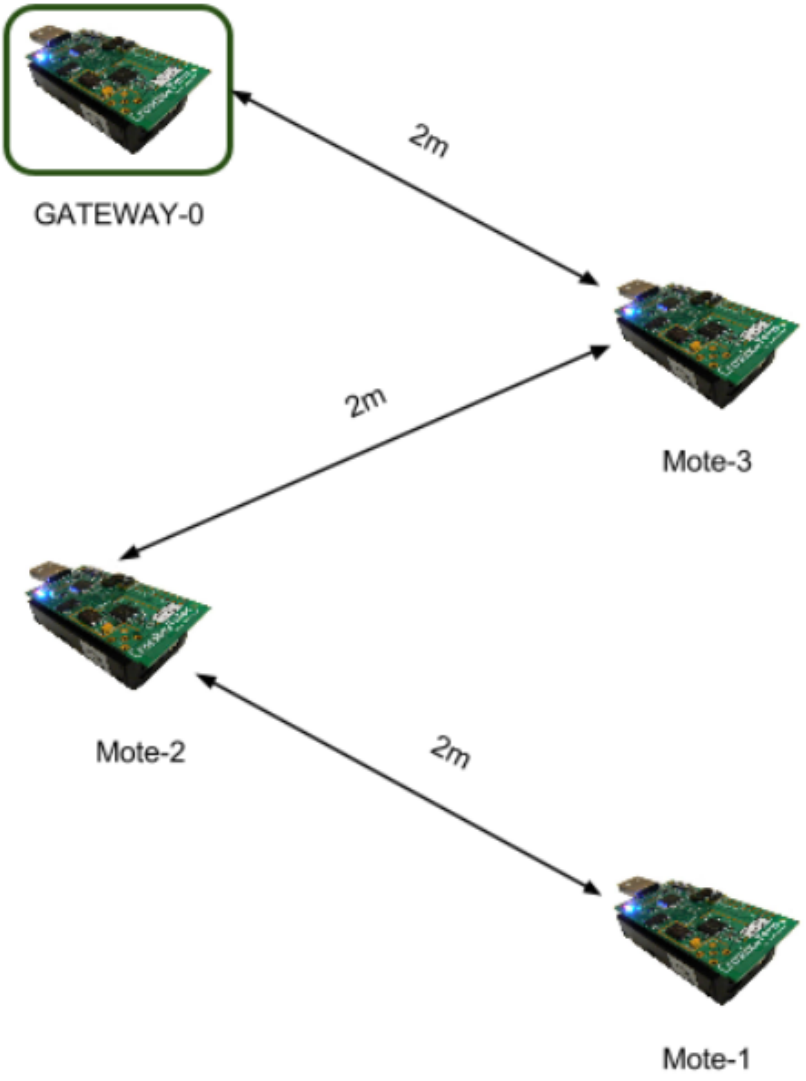


FIGURA 5.3: Col·locació dels terminals en forma de bus

Si tot funciona com està previst s'haurien d'observar fins a 2 enrutaments fets pel terminal 1:

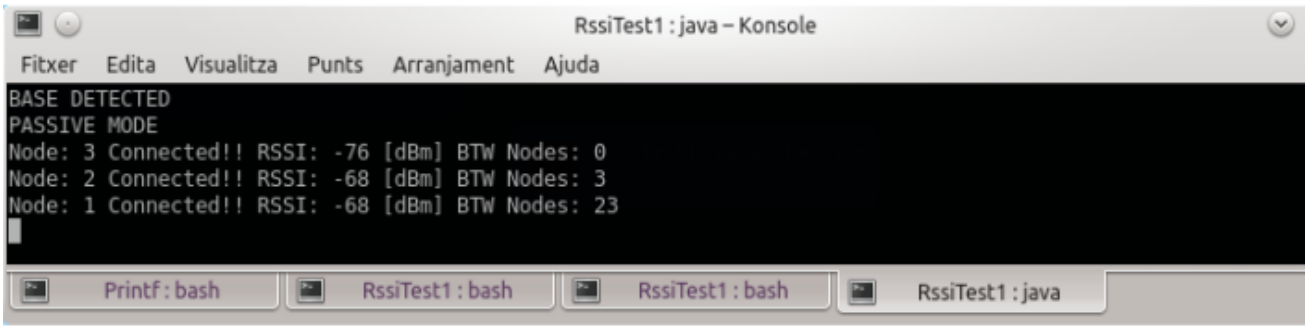


FIGURA 5.4: Comprovació dels terminals en forma de bus

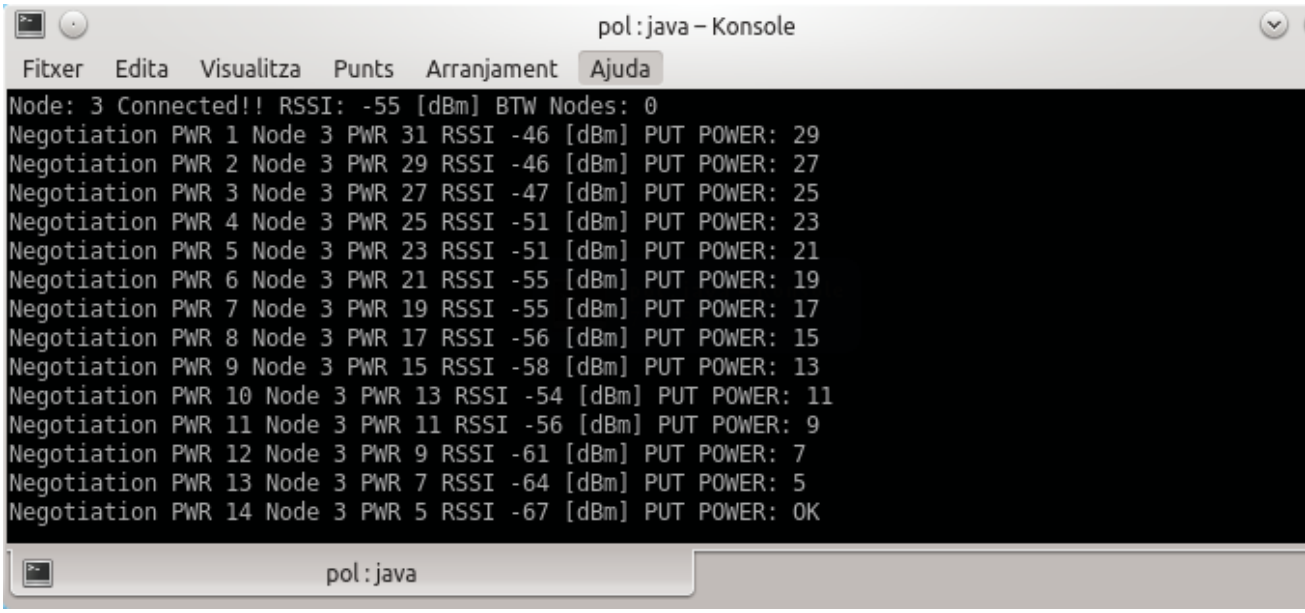
Quan ho hem posat a la pràctica, s'observa com el Mote-3 està connectat al Gateway, el Mote-2 està Connectat al Mote-3, i el Mote-1 es connecta al Gateway passant pel Mote-2 i el Mote-3.

5.1.2 Proves de negociació d'energia

Primer de tot s'ha fet una prova simple per comprovar si la negociació d'energia es compleix. Per fer això ha sigut necessari implementar una funció de log per veure les dades de la negociació. Després s'ha provat si es possible de fer la negociació amb 2 o més terminals a la vegada.

5.1.2.1 Negociació d'un terminal a poca distància

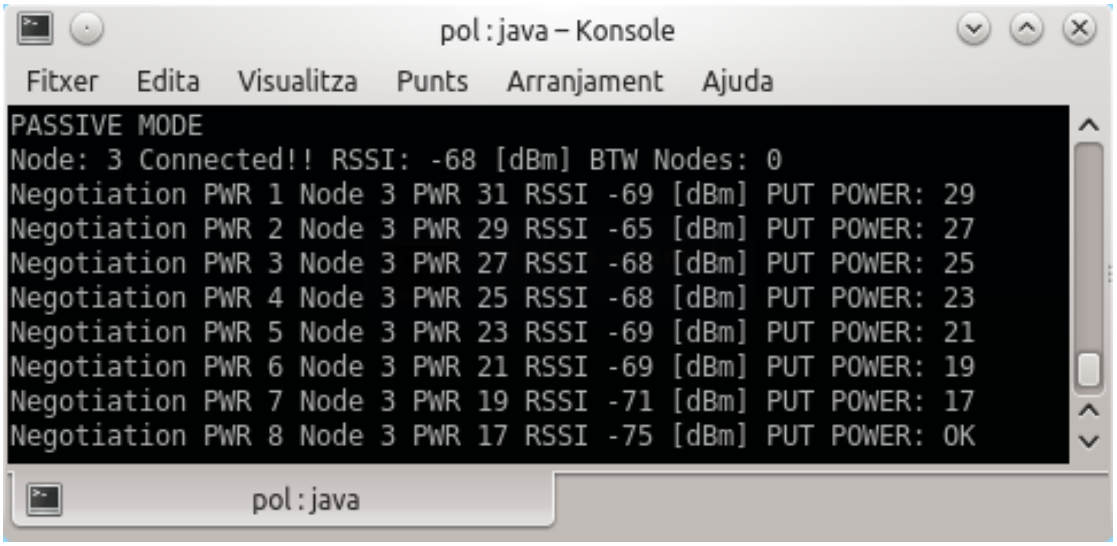
Establim una base amb un terminal a una distància aproximada de 1[m]. Observem els resultats:



S'observa com es necessita enviar fins a 14 paquets per aconseguir establir una connexió estable. Això es degut a que el actual algoritme que es fa servir va disminuint la potència d'emissió gradualment i per tant en distàncies curtes es necessiten més paquets que en llargues. Aquest problema es tracta més a fons en el punt 5.3.1.

5.1.2.2 Negociació d'un terminal a mitja distància

Establim una base amb un terminal a una distància aproximada de 10[m]. Observem els resultats:



S'observa com que a major distància es reben els paquets amb una menor potència la disminució de potència es menor. Per tant podem assolir la potència estable amb un menor número de paquets.

5.1.2.3 Negociació amb varis terminals alhora

Ara posem 2 terminals a difernets distàncies (1 i 10[m]).

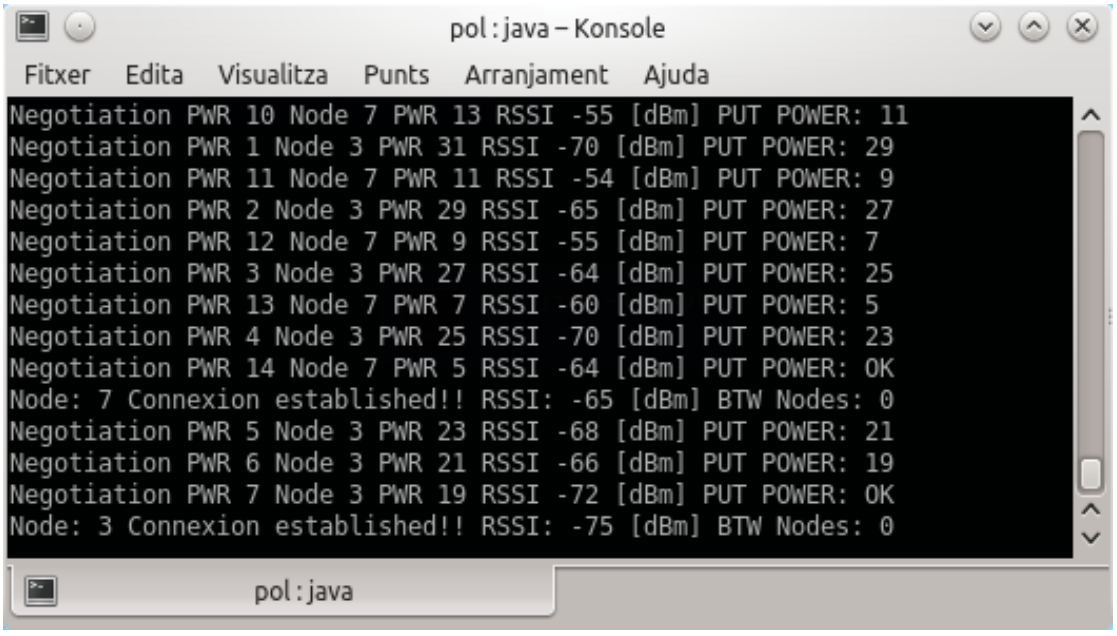


FIGURA 5.5: Negociació d'energía a 1 i 10 [m]

S'observa com el terminal que està més proper (7) negocia amb menys paquets que el terminal que està més llunyà (3).

5.1.3 Establiment de la senyal

En aquest apartat simplement mirarem durant quan període de temps la connexió es manté estable. Això ho podem saber perquè cada 30s es fa un intent de transmissió d'un paquet DATA amb un DATA-ACK de resposta. En cas no rebre cap es reinicia el protocol des de el principi.

5.1.3.1 Enviament del primer paquet DATA

Tal com es veu a la figura 5.5 una vegada establerta la negociació s'envia un paquet DATA per comprovar si la senyal es manté. Una vegada rebut el DATA per la base,

s'envia un paquet ACK indicant al TelosB que la connexió es estable. Aleshores el TelosB encén un LED BlAU.

5.1.3.2 Manteniment de la connexió durant un període de temps

S'ha programat que cada 30 segons s'envii el paquet data. Com podem observar en la figura 5.6, una vegada establerta la connexió s'ha anat enviant paquets DATA cada per comprovar que la connexió es mantingui estable.

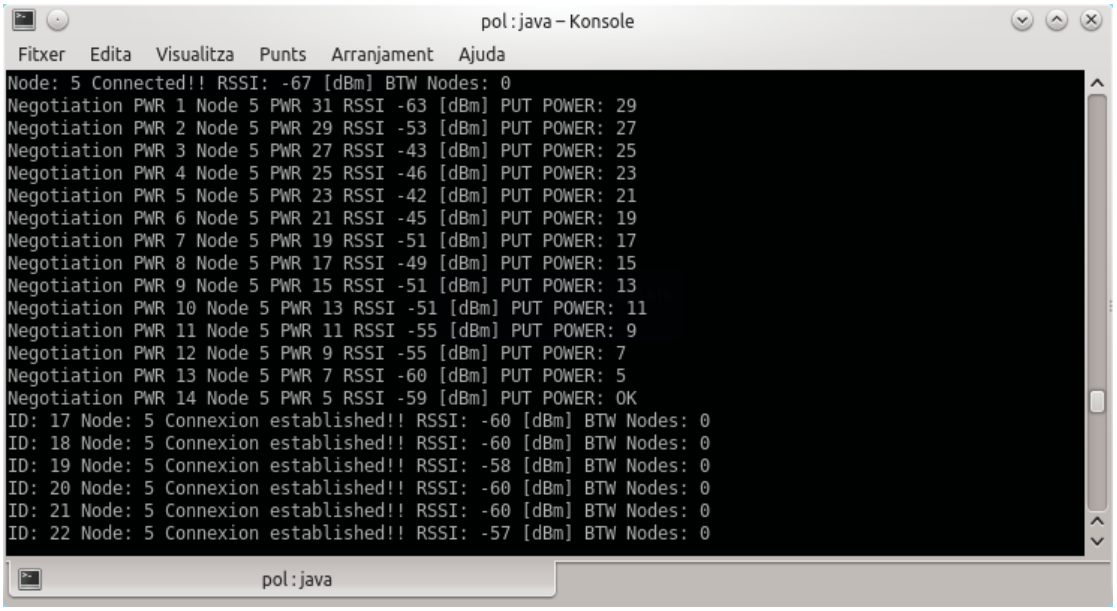


FIGURA 5.6: Manteniment de la connexió

I aquí la mateixa situació però forçant a que els nodes s'enrutin per arribar a la base:

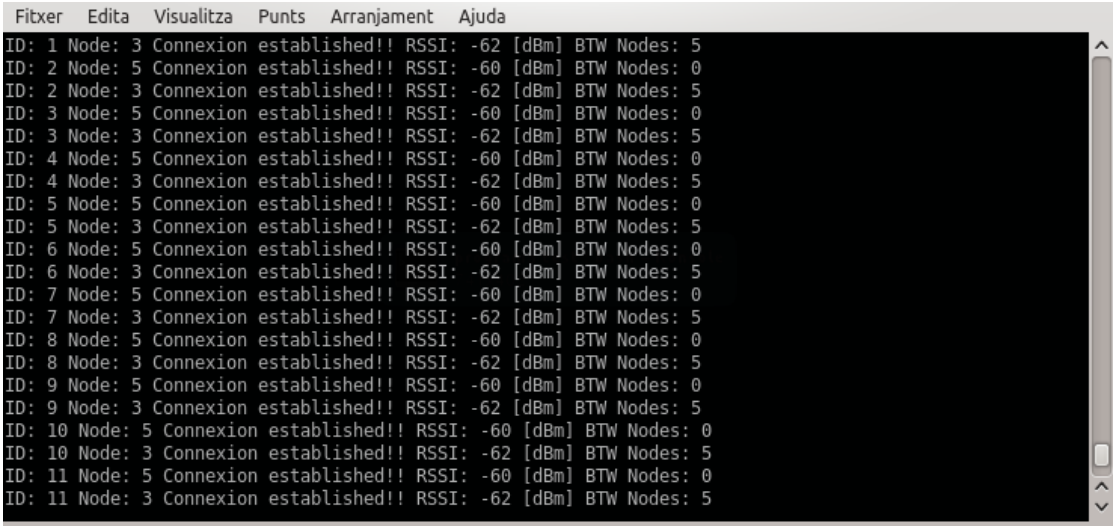


FIGURA 5.7: Manteniment de la connexió

5.1.3.3 Reinicii de la connexió

Per fer aquesta prova simplement el que farem es establir una connexió del TelosB separats per una distància de 0.5[m] per després separar-los a una distància de uns 10[m] això provocarà que en el pròxim paquet DATA detecti un tall de la connexió i reinicii el protocol.
Podem observar a la figura 5.8 com si es talla la connexió el TelosB reinicia el protocol.

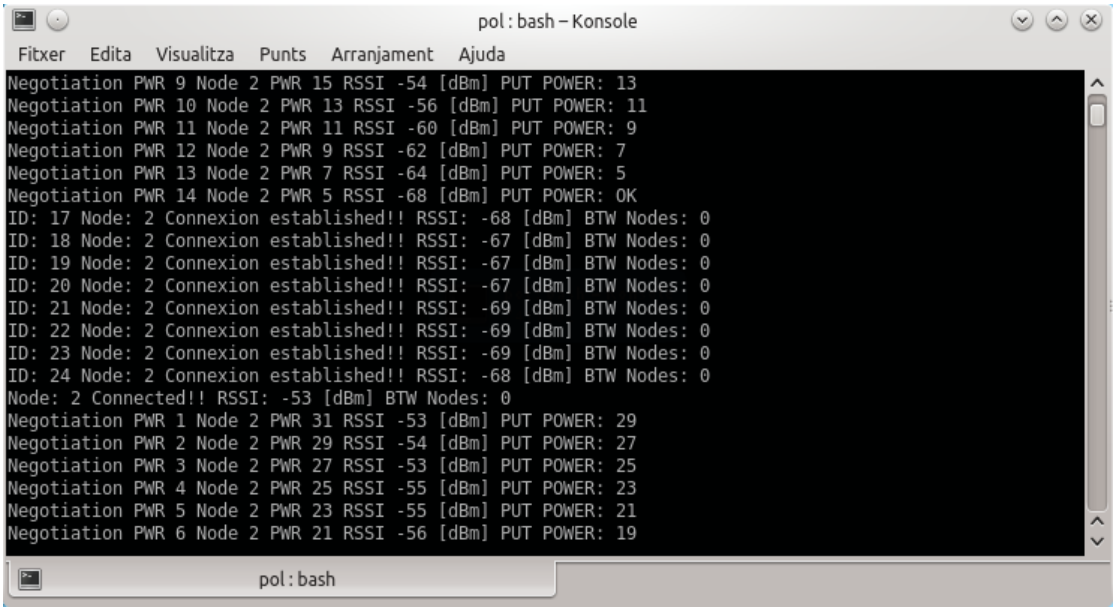


FIGURA 5.8: Reinicii del protocol

5.2 Problemes i solucions

En aquest apartat analitzarem els problemes que s'han trobat i proposarem solucions pràctiques o teòriques per fer front al problema.

5.2.1 Infinite Loop Problem

L'error del bucle infinit(Infinite Loop Problem) succeix quan dos motes, s'enllancen amb si mateixos sense arribar a la base, passa quan hi han situacions com la de la figura 5.9:

Això provocava que tant el Mote-2 com el Mote-3 es connectessin entre ells sense haver cap sortida cap a la gateway. Problema que feia que quan el mote-3 transmet un paquet cap a la base el mote-2 el reenvies al mote-3 i així infinitament. Provocant un bucle infinit, augmentant considerablement el consum energètic.
La solució va ser tant simple com fer una comprovació abans d'enviar un SYNC-ACK si està connectat a un altre mote.

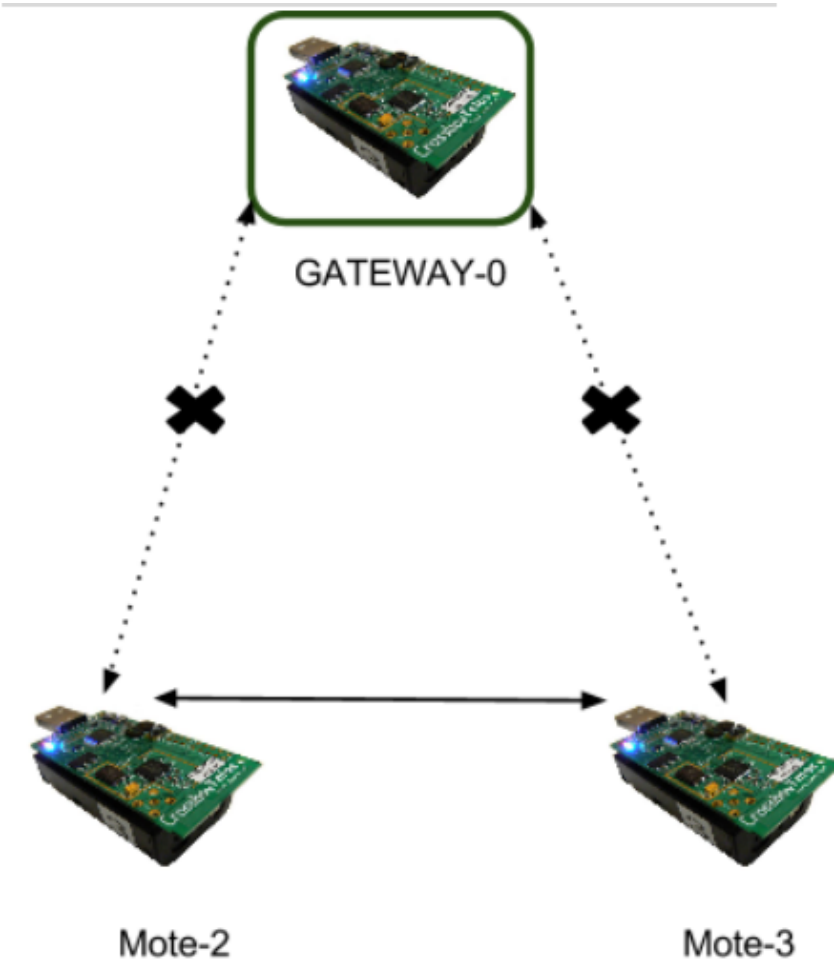


FIGURA 5.9: Error del bucle infinit

5.2.2 Grans pèrdues de paquets en xarxes grans

Durant varies proves s'ha comprovat com el sistema aguanta l'establiment de connexió de 3 motes sense cap problema, però al moment que el número de dispositius ha augmentat s'ha observat com la pèrdua de paquets augmenta linealment a mesura que es va afegint més dispositius a la xarxa. Per a la prova s'ha seguit la mateixa tipologia que la de la figura 5.1 però afegint més nodes. Per tant d'aquesta forma intentem estressar el node base amb les múltiples peticions dels nodes. S'ha mesurat durant un minut els paquets perduts tal com podem veure en la Taula 5.1.

Número Motes	1	2	3	4	5	6
Paquets Perduts (%)	0	0	0	10	25	50

TAULA 5.1: Pèrdues de paquets

La única solució possible es implementar un sistema de cues FIFO [38]. D'aquesta forma quan coincideixen un nombre elevat de paquets s'emmagatzemen a la cua fins que sigui possible processar-los.
La implementació de la cua FIFO senzilla es un array de paquets que quan entra un nou el fica el final de la cua, i quan surt un desplaça els paquets una posició de l'array. No es la solució més eficient i no garanteix una solució definitiva del problema però es suficient per demostrar que una de les possibles causes de pèrdua de paquets es aquesta. Es pot observar en la següent taula 5.2 la millora de les comunicacions una vegada implementada una rudimentària cua senzilla:

Número Motes	1	2	3	4	5	6
Paquets Perduts (%)	0	0	0	5	7	6

TAULA 5.2: Pèrdues de paquets amb cues FIFO senzilles

Igualment la solució més pràctica seria fer ús d'un dels molts protocols d'enrutament ja creats per TinyOS per solucionar adequadament el problema tal com s'explica en el punt 5.3.2

5.2.3 Inestabilitat de la connexió amb potència d'emissió mínima

Durant la fase de proves s'ha observat com en els casos on els TelosB estan molt junts durant la fase de negociació d'energia aquesta es negociada a 1 de potència d'emissió. Tot i que es possible emetre en aquesta potència les connexions son molt inestables i causen que la majoria de paquets siguin incapaços d'arribar a la base.
Les causes d'aquest problema poden ser vàries, des de que l'antena no funciona correctament en potències molt baixes, o ve que al ser emesos els paquets en una energia tant baixa qualsevol moviment o obstacle en l'ambient provoca interferències.
L'única solució és establir una potència mínima on es mantingui la connexió estable i el consum sigui el més reduït possible. Després de més de 10 proves s'ha observat com una potència mínima de 5 es mantenen les connexions de manera estable.

5.3 Millores

En aquesta secció després d'analitzar i corregir els errors del protocol, es passaran a proposar possibles millores en base a l'experiència obtinguda durant les proves i s'intentaran implementar.

5.3.1 Millorar l'establiment de la connexió

Durant les diferents proves s'ha observat com durant la fase de negociació es necessita un gran número de paquets per aconseguir negociar una potència d'emissió estable quan els dos nodes estan separats a petites distàncies. En canvi quan estan a grans distàncies l'intercanvi de paquets es mínim tal com s'observa en la gràfica 5.10.

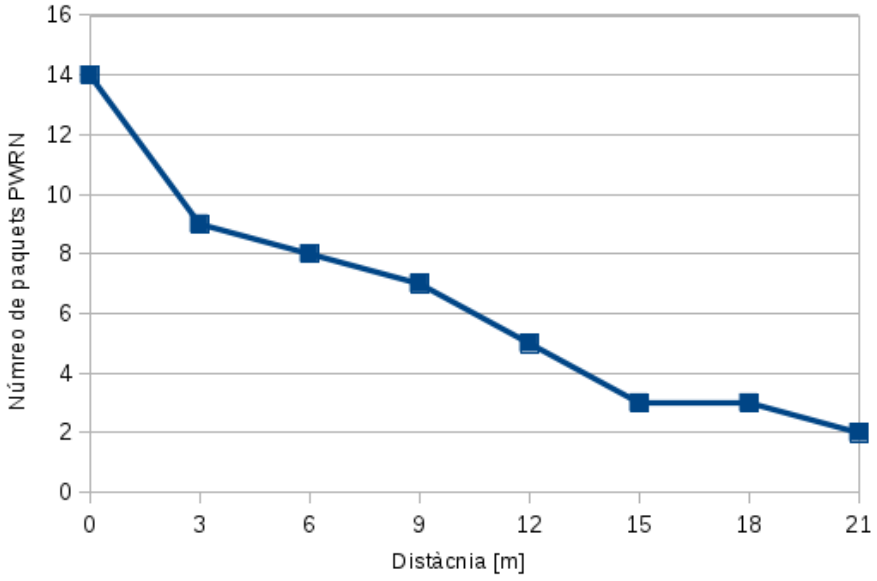


FIGURA 5.10: Gràfica del número de paquets necessaris per establir connexió

Això es deu que l'actual algoritme sempre comença a negociar la potència des de 31 i baixa la potència linealment de 2 en 2 tal com està explicat en el punt 4.3.2.2 . Per tant es ineficient davant distàncies curtes, però eficient en distàncies llargues. Això comporta un gran cost energètic durant la fase d'establiment de la connexió

Per tant una possible millora es modificar l'algoritme de negociació per un que sigui eficient en tots dos casos, amb transmissions a llargues distàncies i en curtes.
Una proposta del nou algoritme sería que enlloc de apropar-te linealment (de 2 en 2) la reducció de potència estaria en funció de la potència rebuda. Per exemple a major qualitat de senyal rebuda més es reduiria la potència. S'ha fet una primera implementació tal com es mostra al codi següent:

```
1  nx_uint16_t negotiationAlgorithm(u_int16_t rssi, u_int16_t power){
2
3    if(MAXDBM-rssi>=10&&power>20){
4      return 10;
5    }
6    if(MAXDBM-rssi>=5&&power>14){
7      return 5;
```

```
8       }
9       return 2;
10    }
```

Índex de codis 5.1: nesC Nou algoritme de negociació

Observem com en distàncies curtes es passa a fer la negociació en 4-5 paquets i en distàncies llargues es fa en 3-4 paquets 5.11. Això es una bona millora ja que s'aconsegueix reduir el nombre mínim de paquets en distàncies curtes un 64% i en distàncies llargues manté el mateix nombre de paquets aproximadament.

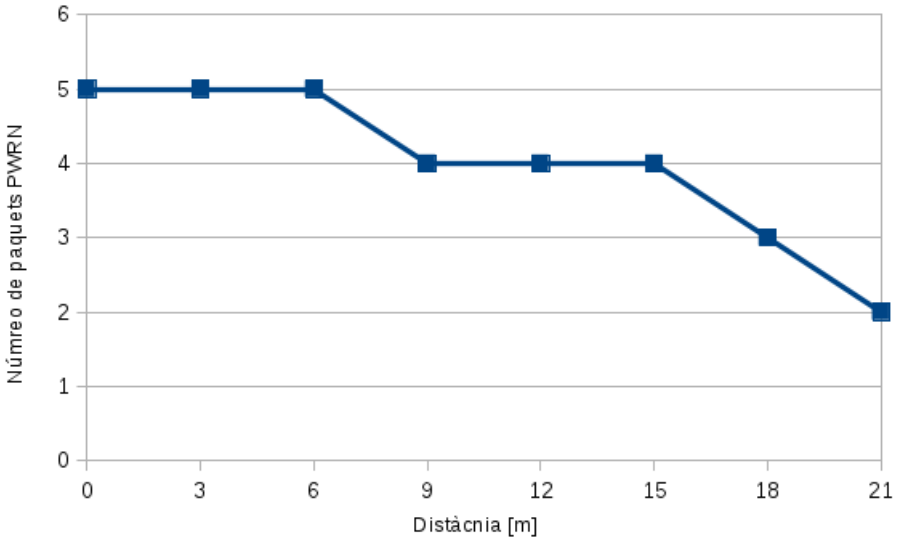


Figura 5.11: Gràfica del número de paquets necessaris per establir connexió

5.3.2 Protocol d'enrutament alternatiu

En l'implementació actual s'utilitzen unes taules d'enrutament molt simples per enrutar els paquets DATA. Aquestes taules no són eficients en xarxes amb més de 9 motes. Això no comporta cap problema per proves del protocol en xarxes petites, però si es vol fer un ús més pràctic per el que està dissenyat el protocol (grans xarxes de sensors) és mostra insuficient. Una solució seria substituir l'actual solució per una més professional i implementar la fase de negociació d'energía. Hi han 2 protocols principals d'enrutament que es podrien implementar:

5.3.2.1 TYMO

Tymo és la implementació en TinyOS del protocol DYMO, un protocol d'enrutament de punt a punt per a xarxes mòbils ad-hoc (MANET). Va ser dissenyat inicialment per

l'IETF per trobar rutes dinàmicament a la part superior de la pila IP. Aquest protocol d'enrutament és simple i relativament fàcil d'implementar[39].

5.3.2.2 TinyRPL

TinyRPL, és la implementació per a TinyOS de IETF's IPv6 Routing Protocol for Low-power and Lossy Networks (RPL). És un protocol d'enrutament per a xarxes amb baixa potència d'emissió, però al ser més complexe que el TYMO i entrar en conflicte amb la part energètica del protocol que es treballa no és el més ideal per el nostre projecte[40].

5.3.3 Energia restant en piles

Durant les proves s'ha observat com a mesura que va baixant la carga de les bateries la potència de la senyal emesa es torna més inestable. Una possible solució seria augmentar l'energia d'emissió per tenir una potència estable a mesura que es detecti que els nivells de bateria baixen per sota d'un mínim. TelosB ofereix la possibilitat de consultar la tensió de la pila, però tot i així el valor que aporta es difícil treure una conclusió sobre si queda molta o poca bateria ja que cada pila es comporta d'una manera diferent.
Per tant la solució més senzilla està en quan es vegi que un node té una gran quantitat de pèrdues de paquets es canvia la pila per una de carregada.

Capítol 6

Conclusions

L'objectiu d'aquest treball ha sigut allargar l'autonomia energètica en les xarxes de sensors sense fils. Per fer això possible s'ha implementat un protocol d'enrutament destinat al estalvi energètic en les comunicacions entre nodes. Aquest protocol s'ha dividit en 3 parts.
Primer s'ha creat una fase d'enrutament on s'intentava que tots els terminals sempre que fos possible es connectessin al node base. La segona fase es la de negociació on l'objectiu principal es establir la potència mínima necessària d'emissió per mantenir una connexió estable entre dos nodes amb un reduït cost energètic. Per acabar l'última fase del protocol es la del manteniment de la connexió perquè sigui estable i la pèrdua de paquets mínima.

Una vegada implementat el protocol en els microcontroladors on s'han fet les proves (TelosB), s'ha observat un estalvi energètic important ja que l'energia d'emissió ha passat de ser la màxima del sensor a una energia que depenia de la qualitat de la senyal que rebia el node destí. Aquesta potència en alguns casos de nodes molt propers s'aconsegueix passar d'emetre a 31 a 5. En cas que els terminals fossin més llunyans la potència d'emissió negociada és més alta i d'aquesta manera menor que la màxima. També s'han optimitzat al màxim el número de paquets necessaris per establir la xarxa de sensors sense fils i per tant aconseguir reduir encara més el consum energètic.
Tot i així caldria matissar que no es possible fer un experiment total del consum del sensor amb el protocol i sense ell, ja que, durant tot el llarg procés de proves cap pila ha estat gastada, i per tant es difícil obtenir resultats en aquest sentit. També cal aclarir que, per fer la comparació s'hauria de construir una xarxa de sensors en un gran espai, ja que el protocol al ser un Data Centric, es difícil veure l'enrutament a no ser que els sensors estiguin situats en distàncies molt grans. També cal aclarir que les piles proporcionen una autonomia diferent depenen de les recarregues que hagin tingut per

culpa de la degradació química de les bateries. Aquests punts fan que sigui complexe demostrar un resultat directe de l'augment de l'autonomia, però les proves que s'han fet demostren que l'estalvi energètic es un fet ja que tal com s'ha explicat es consumeix molta menys energia al emetre els paquets amb el protocol que sense ell.

Durant l'investigació s'han detectat diversos problemes en l'enrutament que provocaven grans pèrdues de paquets en xarxes amb molts nodes. Això succeïx perquè la fase d'enrutament del protocol s'ha implementat des 0 de forma senzilla ja que no era l'objectiu primordial de l'investigació. Amb diferents versions del codi s'ha aconseguit millorar aquests errors, però tot i així el problema no ha estat completament solucionat.

Per tant el següent pas de la recerca sería fer un fork d'un protocol d'enrutament ja implementat en TinyOS com el TYMO y afegir la fase de negociació d'energía. Seguint aquesta investigació seria possible obtenir un protocol robust totalment aplicable de forma pràctica i per tant possibilitant un gran estalvi energètic.

Apèndix A

Apendix

A.1 Instal·lació de la llibreria de TinyOS en Kubuntu 14.04

1. Install tinyos-tools
   sudo apt-get install tinyos-tools-14
2. Instal·lar gcc-msp430
   sudo apt-get install gcc-msp430
3. wget http://github.com/tinyos/tinyos-release/archive/tinyos-2_1_2.tar.gz
   tar xf tinyos-2_1_2.tar.gz
4. vi /home/pol/.bashrc

```
1
2 # Here we setup the environment
3 # variables needed by the tinyos
4 # make system
5
6 export TOSROOT="<local-tinyos-path>"
7 export TOSDIR="$TOSROOT/tos"
8 export CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java
9 export MAKERULES="$TOSROOT/support/make/Makerules"
10 export PYTHONPATH=$PYTHONPATH:$TOSROOT/support/sdk/python
11 echo "setting up TinyOS on source path $TOSROOT"
```

Índex de codis A.1: .bashrc

5. Obrir permisos escritura/lectura del serial USB
   sudo gpasswd –add ¡username¿ dialout
6. Reiniciar sessió.

A.2 Programa més simple per TinyOS

Un exemple del programa més simple que es pot fer per TinyOS. Primer tenim el fitxer de configuració on s'especifiquen els components que es faran servir, després el fitxer del component on s'implementa el codi. I per acabar el Makefile on especifica com s'ha de compilar.

```
configuration SimpleAppC{
}
implementation{
    components SimpleC, MainC;

    SimpleC.Boot -> MainC.Boot;
}
```

Índex de codis A.2: Configuration file

```
module SimpleC{
    uses interface Boot;
}

implementation{
    event void Boot.booted()
    {
        //The entry point of the program
    }
}
```

Índex de codis A.3: Component file

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

Índex de codis A.4: Makefile

Per compilar i flashejar simplement has d'escriure el següent dins de la carpeta del projecte:
make telosb
Cerques el nom del teu telosb(DEVICE) connectat amb:
motelist
I flashejes amb:
make telosb reinstall bsl,DEVICE

A.3 Posar a punt el IDE per Eclipse

Per desenvolupar aplicacions per TinyOS es pot fer mitjançant qualsevol editor de text i compilant i flashejant mitjançant comandas. Aquesta forma es pot fer poder pesada, i per tant existeixen altres solucions molt més comodes com per exemple integrar el tinyOS en un IDE.

1. Instalar eclipse.
   sudo apt-get install eclipse
2. Dins del eclipse instalar el pluguin Yeti 2
   http://tos-ide.ethz.ch/wiki/pmwiki.php?n=Site.Installation
3. Posar l'espai de treball de TinyOS
   Open Prespective-¿TinyOS
4. Configurar el plugin per que et compili i "flasheji" els projectes.
   Run-¿Run Configurations
5. Crear una nova configuració d'arracada
   Selecciona TinyOS Build i clicar "new launch configuration"
6. Anar a extres i seleccionar les dos opcions següents A.1:
7. Ara quan premis run a un projecte te'l compilarà i instalarà automaticament.

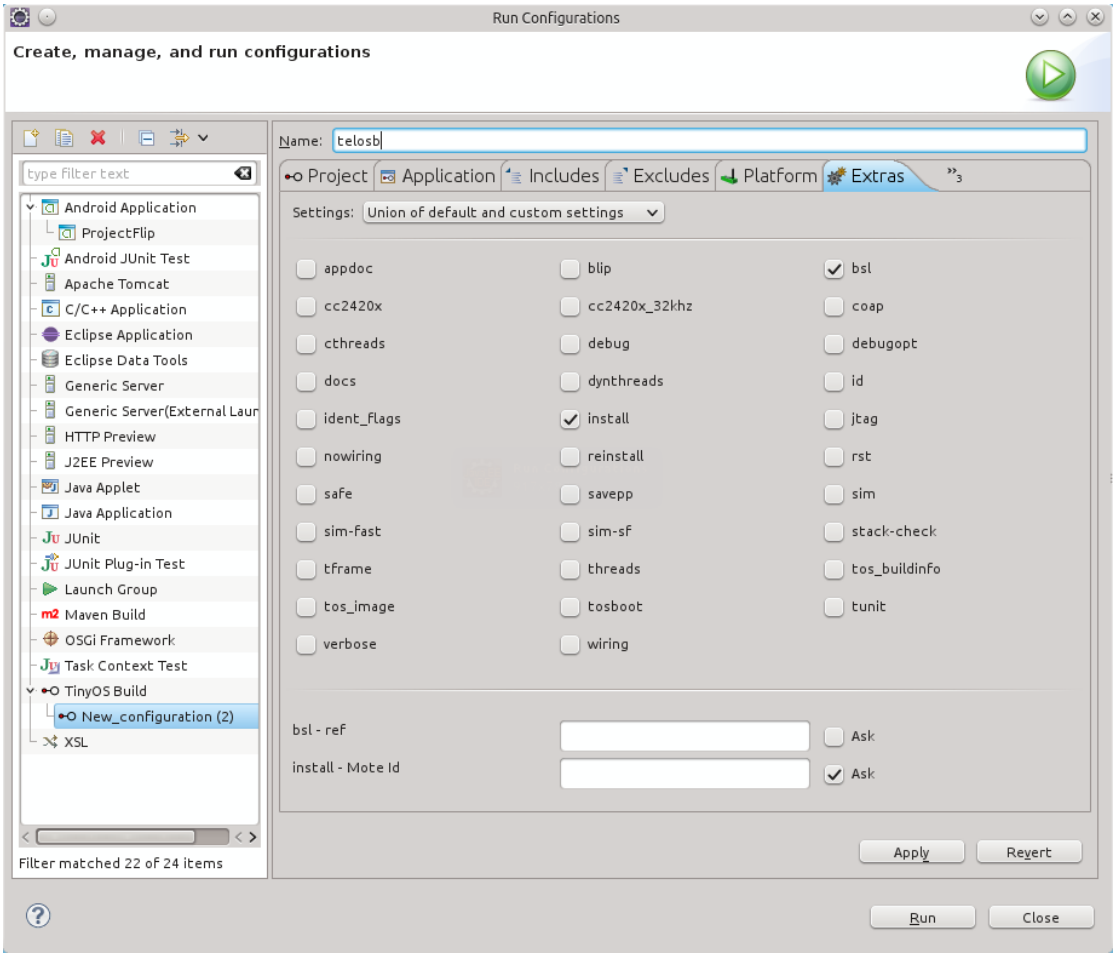


FIGURA A.1: Configuració dels extres pel telosb

Bibliografia

[1] B Van Tuijl, E van Os, and E van Henten. Wireless sensor networks: state of the art and future perspective. In *International Symposium on High Technology for Greenhouse System Management: Greensys2007 801*, pages 547–554, 2007.

[2] Shekhar Borkar. Obeying moore's law beyond 0.18 micron [microprocessor design]. In *ASIC/SOC Conference, 2000. Proceedings. 13th Annual IEEE International*, pages 26–31. IEEE, 2000.

[3] Qiuhua Cao and John A Stankovic. An in-field-maintenance framework for wireless sensor networks. In *Distributed Computing in Sensor Systems*, pages 457–468. Springer, 2008.

[4] Priya Ganapati. Why arduino is a hit with hardware hackers, June 2010. URL http://www.wired.com/2010/07/hardware-hobbyists-arduino/.

[5] Ken Leung. Arduino: A brief history, October 2014. URL http://www.kenleung.ca/portfolio/arduino-a-brief-history-3/.

[6] David Kushner. The making of arduino, October 2011. URL http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino.

[7] Uno r3 mega328p atmega16u2 for arduino compatible, October 2014. URL http://www.aliexpress.com/item/Best-prices-UNO-R3-MEGA328P-ATMEGA16U2-for-Arduino-Compatible-Free-Shipping-Drops909950710.html.

[8] Assembling the really bare bones board (rbbb) arduino clone, August 2008. URL http://www.instructables.com/id/Assembling-the-Really-Bare-Bones-Board-RBBB-Ardu/.

[9] Arduino store, October 2014. URL http://store.arduino.cc/.

[10] George Wong. Build your own prototype raspberry pi minicomputer, October 2011. URL http://www.ubergizmo.com/2011/10/build-raspberry-pi-minicomputer/.

[11] Yocto project, October 2014. URL https://www.yoctoproject.org/.

[12] Johanna Williams. Telosb mote wireless sensor mote platform, December 2013. URL https://telosbsensors.wordpress.com/tag/telosb-mote/.

[13] Craig A Grimes, Elizabeth C Dickey, and Michael V Pishko. *Encyclopedia of sensors*. American Scientific Publishers, 2006.

[14] Xbee. Xbee sepcs, October 2014. URL http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module#specs.

[15] Ren Ping Liu, Gordon J Sutton, and Iain B Collings. Power save with offset listen interval for ieee 802.11 ah smart grid communications. In *Communications (ICC), 2013 IEEE International Conference on*, pages 4488–4492. IEEE, 2013.

[16] Ieee 802-2005, October 2014. URL http://standards.ieee.org/getieee802/download/802.15.1-2005.pdf.

[17] Ble121lr bluetooth smart long range module, October 2014. URL https://www.bluegiga.com/en-US/products/bluetooth-4.0-modules/ble121lr-bluetooth--smart-long/.

[18] Arne Simonsson and Anders Furuskar. Uplink power control in lte-overview and performance, subtitle: principles and benefits of utilizing rather than compensating for sinr variations. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pages 1–5. IEEE, 2008.

[19] Wireless Sensor Tags. Monitor and find everything from internet, October 2014. URL https://www.mytaglist.com/index.html.

[20] Smart Citizen. Smart citizen: Landing, October 2014. URL http://www.smartcitizen.me/.

[21] Martin Atzmueller and Katy Hilgenberg. Towards capturing social interactions with sdcf: An extensible framework for mobile sensing and ubiquitous data collection. In *Proceedings of the 4th International Workshop on Modeling Social Media*, page 6. ACM, 2013.

[22] Luis Javier García Villalba, Ana Lucila Sandoval Orozco, Alicia Triviño Cabrera, and Claudia Jacy Barenco Abbas. Routing protocols in wireless sensor networks. *Sensors*, 9(11):8399–8421, 2009.

[23] Habib M Ammari and Sajal K Das. Promoting heterogeneity, mobility, and energy-aware voronoi diagram in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 19(7):995–1008, 2008.

[24] Jae Duck Yu, Kyung Tae Kim, Bo Yle Jung, and Hee Yong Youn. An energy efficient chain-based clustering routing protocol for wireless sensor networks. In *Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on*, pages 383–388. IEEE, 2009.

[25] Habib M Ammari. Energy sink-hole problem with always-on sensors in two-dimensional deployment fields. In *Challenges and Opportunities of Connected k-Covered Wireless Sensor Networks*, pages 241–269. Springer, 2009.

[26] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, et al. Tinyos: An operating system for sensor networks. In *"Ambient intelligence*, pages 115–148. Springer, 2005.

[27] Tinyos.stanford.edu. Faq - tinyos wiki, 2014. URL http://tinyos.stanford.edu/tinyos-wiki/index.php/FAQ#Where_is_it_useful.3F.

[28] Tinyos.net. Tinyos home page, 2014. URL http://www.tinyos.net/.

[29] Linfo.org. Bsd license definition, 2014. URL http://www.linfo.org/bsdlicense.html.

[30] Per Brinch Hansen. *Classic operating systems: from batch processing to distributed systems*. Springer, 2001.

[31] Robert W Floyd. The paradigms of programming. *Communications of the ACM*, 22(8):455–460, 1979.

[32] Raja Jurdak, Antonio G Ruzzelli, and Gregory MP O'Hare. Radio sleep mode optimization in wireless sensor networks. *Mobile Computing, IEEE Transactions on*, 9(7):955–968, 2010.

[33] Peter Trenkamp. Wireless sensor network platforms datasheets versus measurements, September 2011. URL http://www.comnets.uni-bremen.de/itg/itgfg521/aktuelles/fg-workshop-29092011/Peter-FG_TUHH_PTr.pdf.

[34] Dmoss Sissou. Cc2420, March 2010. URL http://tinyos.stanford.edu/tinyos-wiki/index.php/CC2420.

[35] Mathieu. [tinyos-help] rssi conversion for telosb, October 2007. URL http://mail.millennium.berkeley.edu/pipermail/tinyos-help/2007-October/028840.html.

[36] Jang-Ping Sheu. Tinyos lesson 6 – topology control, October 2014. URL http://hscc.cs.nthu.edu.tw/~sheujp/lecture_note/wsn_lab06_en.pdf.

[37] Paul. cc2420packet setpower and getpower, November 2009. URL http://mail.millennium.berkeley.edu/pipermail/tinyos-help/2009-November/043369.html.

[38] Edward G Coffman Jr and Micha Hofri. A class of fifo queues arising in computer systems. *Operations Research*, 26(5):864–880, 1978.

[39] I Chakeres and C Perkins. Dynamic manet on-demand (aodvv2) routing draft-ietf-manet-dymo-26. Technical report, IETF Internet-Draft, http://tools. ietf. org/id/draft-ietf-manet-dymo-26. txt.

[40] Tim Winter. Rpl: Ipv6 routing protocol for low-power and lossy networks. 2012.