Alarma d'incendis amb detectors sense fils

Sebastià Reinés Sastre
Enginyeria Tècnica d'Informàtica de Sistemes

Consultor: Jordi Bécares Ferrés

12 de juny de 2012

A na Catalina, en Biel, en Pere i na Xisca
per la seva paciència i el seu suport

Resum

L'objectiu d'aquest treball de final de carrera és desenvolupar un sistema d'alarma contra incendis basat en tecnologia WSN (xarxa de sensors sense fils).

Precisament l'us d'aquesta tecnologia és el que aporta la major novetat i el major interès. Permet fer la instal·lació del sistema de seguretat estalviant-se, no tan sols el cost econòmic del cablejat i de l'obra necessària per fer-lo arribar a cada un dels sensors, sinó també el cost estètic. No és factible que per protegir contra incendis un monument, edifici o patrimoni facem malbé el seu valor artístic.

En aquest cas els dispositius emprats per a implementar el prototip són dues motes Cou24 que estan basades en un mòdul de la sèrie ZigBit de la casa Atmel, al qual hi ha connectats un conjunt de sensors, un d'ells de temperatura. Funcionen amb bateries i es comuniquen sense fils mitjançant l'estàndard ZigBee. El programa ha de fer que activin l'alarma d'incendi quan detectin que la temperatura sobrepassa la temperatura llindar prefixada.

Aquest sistema està controlat per una aplicació que corre en un ordinador, constituint la consola de control encarregada d'avisar a l'usuari quan es produeixi una alarma. Des de la consola, l'usuari també pot configurar una sèrie de paràmetres dels sensors, com la temperatura llindar, la periodicitat amb la que el sensor ha de prendre mostra de la temperatura, del nivell de bateria etc.

La mota també disposa d'un botó per si cal activar l'alarma de forma manual i disposa d'una sèrie de leds que ens indiquen en quin estat es troba la mota.

Una de les motes, programada amb la funció d'estació base, actua d'interfície entre l'ordinador i la xarxa de sensors i l'altra porta el programa dissenyat per a actuar com a node sensor. El sistema esta pensat perquè la xarxa tengui una topologia en estrella, de tal forma que els nodes sensors es poden comunicar únicament amb el node base i aquest amb el PC pel port USB, sent la comunicació bidireccional.

Aquests tipus de nodes executen el sistema operatiu TinyOS i per programar les aplicacions que han d'executar s'empra el llenguatge de programació nesC.

Índex de continguts

Índex de figures

1. Introducció

Les xarxes de comunicació sense fils han tingut un ràpid desenvolupament en els darrers anys i han permès, per exemple, que tecnologies com GPRS o WI-FI hagin universalitzat l'accés a Internet des de qualsevol telèfon mòbil, tauleta tàctil o ordinador portàtil.

Enfocades a un altre camp d'aplicació han sorgit també les xarxes WSN (Wireless Sensor Network), xarxes de sensors sense fils, que es basen en dispositius de baix cost i baix consum anomenats nodes o motes, capaços d'obtenir informació del seu entorn, processar-la localment i enviar-la fins a un node central de control.

Paral·lelament a l'aparició de les motes, l'any 1999 la Universitat de Califòrnia, Berkeley va desenvolupar el sistema operatiu TinyOs pensat específicament per a executar-se en aquests tipus de dispositius. Des de llavors ha crescut, s'ha generalitzat el seu ús i s'han creat llibreries, entorns de desenvolupament, simuladors etc. que fan molt més fàcil la programació de les motes amb nesC (**n**etwork **e**mbedded **s**ystems C), que és una variant del llenguatge C pensada per a programar aplicacions sobre la plataforma TinyOs.

Són essencialment aquests dos conceptes, les xarxes de sensors sense fils WSN i la programació de les motes el que constitueixen la base d'aquest treball, l'objectiu del qual és desenvolupar un sistema d'alarma contra incendis basat en una xarxa de sensors sense fils.

1.1. Justificació

Actualment les normatives d'edificació preveuen la instal·lació de sistemes contra incendis en tots els recintes d'accés públic, escoles, hospitals, establiments turístics i en general en tots els llocs on existeixi el risc de que el foc pugui provocar una situació d'emergència.

Aquests sistemes de seguretat necessaris presenten una sèrie de problemes, el més important dels quals és segurament l'elevat cost econòmic de la instal·lació. Els sistemes convencionals requereixen, a més dels sensors, tot un sistema de cablejat per alimentar els sensors i per transmetre la informació cap al sistema centralitzat de monitorització i control d'alarmes. Però a part d'aquest cost econòmic també hi ha el cost estètic que suposa tota l'obra necessària per fer arribar el cablejat fins a cada un dels sensors. Monuments i edificis catalogats pel seu interès artístic han de poder estar protegits contra incendis, però sense que això els faci perdre el seu propi valor artístic.

És per tot això que sorgeix la proposta de desenvolupar un sistema de detecció d'incendis que treballi sobre tecnologia WSN (Xarxes de sensors sense fils). Aquests sistemes tenen un cost d'instal·lació molt baix al no necessitar el cablejat, ni per alimentar els sensors, ja que funcionen amb bateries, ni per transmetre les dades, ja que ho fan mitjançant comunicació sense fils.

1.2. Descripció del projecte

Aquest projecte preveu desenvolupar un sistema de detecció d'incendis, basat amb la detecció per increment de temperatura, que treballi sobre tecnologia WSN (Xarxes de sensors sense fils) per tal d'evitar el cablejat de la instal·lació i abaratir els costos.

Ha de constar d'una aplicació de gestió i control del sistema que s'executarà en un PC i d'una xarxa de nodes sensors, un dels quals serà el node estació base que, a part d'actuar també com a sensor, s'encarregarà d'actuar com a interfície entre l'ordinador i la resta de nodes sensors.

La xarxa de nodes sensors ha de tenir una topologia en estrella, de tal forma que els nodes sensors es podran comunicar únicament amb el node base i aquest amb el PC pel port USB, sent la comunicació bidireccional.

El funcionament general del sistema es basa en que cada node sensor va llegint de forma periòdica el sensor de temperatura. Si en qualque moment la temperatura arriba a superar una temperatura llindar establerta, fa saltar una alarma que senyala mitjançant els leds en el node i que envia a l'estació base, per tal de mostrar-la en el PC. Cada sensor també ha de disposar d'un botó per poder activar l'alarma d'incendi de forma manual.

L'usuari que supervisa l'aplicació, veu l'alarma i té informació de si s'ha disparada de forma automàtica o manual, quin sensor l'ha generada, quina és la temperatura que està mesurant el sensor i a quina hora s'ha generada. Quan vegi l'alarma haurà de confirmar-ne la recepció amb la qual cosa el sistema sabrà que l'alarma ja ha estat visualitzada.

Per tal de garantir el correcte funcionament de tots els sensors de la xarxa, cada un d'ells pren mostra del seu nivell de bateria, si aquest baixa d'un llindar establert ha d'enviar una alarma per tal de notificar-ho i procedir a la substitució de les bateries. Per altre banda el nivell de bateria s'envia de forma periòdica per tal de tenir informació de l'estat del sistema i poder garantir que tots els nodes funcionen correctament; si la consola deixa de rebre aquesta comunicació d'un node durant un temps més gran de l'esperat, ho notificarà a l'usuari a través de l'aplicació del PC.

L'usuari, des de l'aplicació de control, ha de poder configurar els següents paràmetres de la mota:

- La temperatura llindar d'alarma
- El període de lectura de la temperatura per tal de comparar-la amb la temperatura llindar
- El període d'enviament del nivell de bateria
- Si vol rebre o no totes les mesures que efectua la mota i cada quant de temps les vol rebre

La mota ha de senyalitzar mitjançant els leds l'estat en que es troba:

- Sensor apagat. Pot encendre un led periòdicament conforme el node pot funcionar correctament.
- No alarma. Pot encendre un led periòdicament conforme el node funciona correctament.
- Alarma detectada
- Alarma rebuda en estació central
- Alarma reconeguda
- Nivell de bateria baix
- Fora de cobertura

La transmissió dels missatges d'alarma ha de ser amb acusament de recepció ACK per tal de que no es perdi cap alarma.

Les motes han de tenir un sistema de protecció de caigudes, han de fer servir el WatchDog per evitar quedar-se penjades i han de recuperar la configuració de forma automàtica. A més, les motes sensor han de disposar d'un sistema per comprovar que en el lloc on s'instal·len hi ha cobertura amb el node base.

El sistema per activar i desactivar els sensors ha de ser el següent: La mota s'activarà passant un imant dues vegades seguides per sobre del sensor d'efecte Hall. Es desactivarà passant-lo quatre vegades seguides.

Com a funcionalitats extres inclourà també la lectura de la fotocèl·lula i una interfície gràfica d'usuari.

1.3. Objectius

Els requisits enumerats es compliran si s'assoleixen els següents objectius:

- Llegir de forma periòdica el sensor de temperatura
- Comparar la temperatura amb la temperatura llindar d'alarma
- Transmetre els missatges d'alarma de forma segura amb ACK
- Senyalitzar l'alarma de forma local en el node
- Habilitar el botó d'alarma manual
- Llegir de forma periòdica el nivell de bateria
- Comparar el nivell de bateria amb el nivell llindar d'alarma de bateria

- Transmetre els missatges de nivell de bateria cada N segons de forma segura amb ACK
- Transmetre els missatges d'alarma de bateria de forma segura amb ACK
- Senyalitzar l'alarma de bateria baixa de forma local en el node
- Programar el WatchDog per evitar que la mota es quedi penjada
- Recuperar la configuració si la mota es reinicia després d'actuar el WatchDog
- Transmetre les lectures de temperatura, nivell de bateria i lluminositat cada M segons
- Activar la mota passant dos cops un imant per sobre del sensor d'efecte Hall
- Desactivar la mota passant quatre vegades un imant per sobre del sensor d'efecte Hall
- Configurar remotament des de la consola la temperatura llindar d'alarma de cada mota
- Configurar remotament des de la consola la periodicitat de lectura de la temperatura
- Configurar remotament des de la consola la periodicitat d'enviament del nivell de bateria
- Configurar remotament des de la consola la periodicitat d'enviament de les dades
- Mostrar en la consola el missatge d'alarma de foc amb l'hora que s'ha produït l'alarma
- Confirmar des de la consola la recepció d'alarma de foc
- Mostrar en la consola el missatge d'alarma de bateria baixa
- Mostrar en la consola el missatge d'alarma de sensor perdut quan es deixin de rebre els missatges de nivell de bateria d'un node
- Mostrar en la consola totes les dades rebudes
- Mostrar en una interfície gràfica totes les dades i les alarmes
- Configurar des de la interfície gràfica tots els paràmetres de la mota

1.4. Enfocament i metodologia

Es tracte de dissenyar un sistema de seguretat contra incendis el més fiable possible, basat en el monitoratge de la temperatura, assegurant que els sensors estan sempre actius i que quan detectin una alarma aquesta arribarà a la consola d'usuari.

El mètode seguit per desenvolupar l'aplicació encaixa clarament dins el que és un cicle de vida iteratiu i incremental en el qual es distingeixen aquestes quatre etapes:

L'etapa primera o d'inici coincideix amb la publicació de l'enunciat amb la justificació del projecte i la llista de requisits. També en aquesta etapa ha quedat fixades les limitacions temporals i el maquinari a utilitzar.

L'etapa segona o d'elaboració durant la qual se fa l'estudi del problema, es preveuen les necessitats d'informació, s'estableix a grans trets l'arquitectura del programari i es fa una planificació del projecte.

La part més novedosa i desconeguda, i per tant la que genera més necessitats d'informació és la referent a la programació de les motes COU24 que són les que se fan servir en aquest cas. Cal conèixer el maquinari, especialment com interactua el microcontrolador amb els sensors i els leds i també com ho fa amb la radio per enviar i rebre missatges. Per això resulten molt útils els exemples d'aplicació del tutorial de TinyOS.

Pel que fa a l'arquitectura, l'opció triada és crear les configuracions, mòduls i interfícies necessàries per a separar el més possible cada una de les tasques, per exemple llegir els sensors, gestionar les comunicacions etc.

L'etapa tercera o de construcció durant la que es desenvolupa tot el producte de manera iterativa i incremental.

Afrontam primer el processos per llegir els sensors, enviar missatges des de la mota a la consola i, amb una programació bàsica de la interfície d'usuari, presentar les dades rebudes a la pantalla.

Amb aquesta part embastada la prioritat passa a ser el tractament dels valors recollits dels sensors per tal de detectar possibles situacions d'alarma.

Seguidament continuarem amb la part de seguretat, enviament de missatges amb ACK i activació del WatchDog. Després la inclusió del botó d'alarma manual i la senyalització corresponent als diferents estats amb els leds.

La següent passa és afegir a la consola una aplicació que permeti a l'usuari introduir els paràmetres de configuració de la mota, generant i enviant els missatges corresponents. Al mateix temps, a la mota, programar la recepció dels missatges i l'aplicació dels nous paràmetres.

Quedarà pel final desenvolupar la interfície gràfica, que ha de reunir en una sola aplicació dues funcionalitats, monitorar les alarmes i les dades rebudes de la mota i configurar els paràmetres.

La quarta etapa o de transició coincideix amb el lliurament del codi, un codi en el que encara caldria fer-hi retocs com a conseqüència d'errors detectats o de requisits que s'havien passat per alt fins aleshores.

1.5. Planificació del projecte

1.5.1. Planificació inicial

La relació d'objectius definits anteriorment es concreta en la següent relació de tasques. Si les repartim d'acord amb el temps que tenim, limitat per les dates de lliurament del producte, donen com a resultat la planificació inicial i el diagrama de Gantt que podem veure a les figures 1 i 2.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		Estudi documentació i tutorials motes COU24	35 días	mar 20/03/12	dom 06/05/12
2		**Programació dels nodes**	**32 días**	**mié 21/03/12**	**jue 03/05/12**
3		Comunicació node sensor - node base - PC	8 días	mié 21/03/12	vie 30/03/12
4		Lectura automàtica dels sensors	8 días	mar 27/03/12	jue 05/04/12
5		Comunicació de lectures dels sensors	8 días	sáb 31/03/12	mar 10/04/12
6		Detecció de sobrepassament del llindar	6 días	mar 03/04/12	mar 10/04/12
7		Comunicació d'alarma	6 días	dom 08/04/12	vie 13/04/12
8		Activació d'alarma manual i indicació leds	6 días	mar 10/04/12	mar 17/04/12
9		Programació WatchDog	6 días	dom 22/04/12	vie 27/04/12
10		Indicació existència de cobertura	6 días	jue 26/04/12	jue 03/05/12
11		Ensesa i apagada amb imant	4 días	mar 10/04/12	vie 13/04/12
12		**Programació de la cònsola de control**	**28 días**	**mié 21/03/12**	**vie 27/04/12**
13		Per mostrar l'alarma i dades corresponents	12 días	mié 21/03/12	jue 05/04/12
14		Per confirmar la recepció d'una alarma	6 días	vie 06/04/12	vie 13/04/12
15		Per configurar remotament paràmetres	6 días	mié 11/04/12	mié 18/04/12
16		Per mostrar temperatura i bateria dels nodes	6 días	jue 19/04/12	jue 26/04/12
17		Per detectar mal funcionament del node	6 días	dom 22/04/12	vie 27/04/12
18		Lectura i enviament lluminositat	4 días	jue 03/05/12	mar 08/05/12
19		Interfície gràfica	15 días	mié 25/04/12	mar 15/05/12
20		**Preparació del software del TFC**	**45 días**	**mié 21/03/12**	**mar 22/05/12**
21		Lliurament primera versió	1 día	mar 24/04/12	mar 24/04/12
22		Lliurament versió definitiva	1 día	mar 22/05/12	mar 22/05/12

Figura 1: Llistat de tasques de la programació inicial

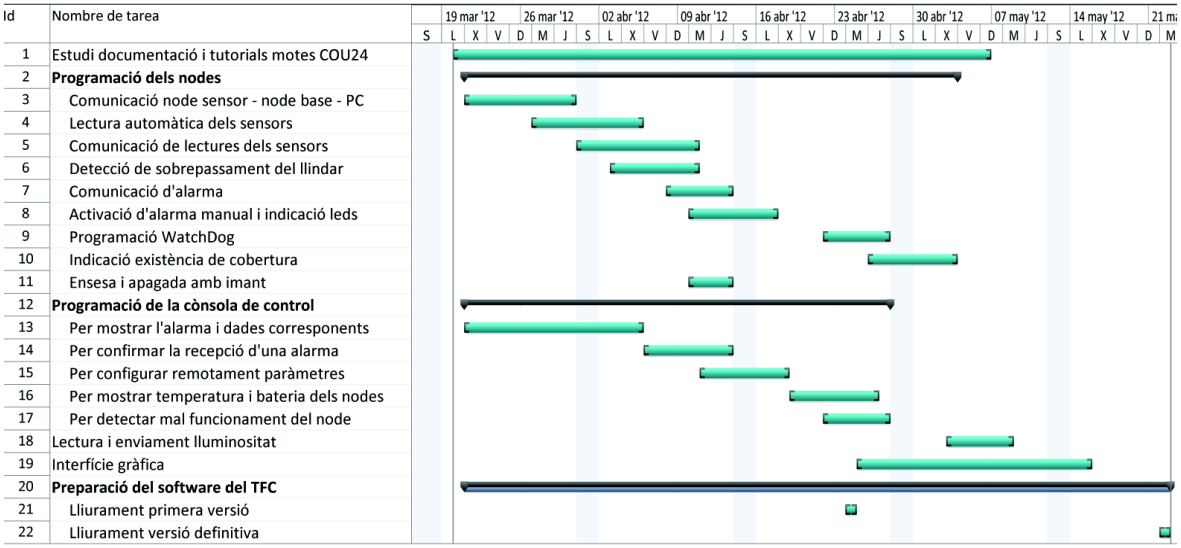

Figura 2: Diagrama de Gantt de la programació inicial

1.5.2. Planificació final

	Nombre de tarea	Duraci	Comienzo	Fin
1	Estudi documentació i tutorials motes COU24	35 días	mar 20/03/12	dom 06/05/12
2	**Programació dels nodes**	**48 días**	**mié 21/03/12**	**sáb 26/05/12**
3	Comunicació node sensor - node base - PC	20 días	mié 21/03/12	mar 17/04/12
4	Lectura automàtica dels sensors	10 días	mar 03/04/12	lun 16/04/12
5	Comunicació de lectures dels sensors	10 días	mar 03/04/12	lun 16/04/12
6	Detecció de sobrepassament del llindar	6 días	dom 08/04/12	vie 13/04/12
7	Comunicació d'alarma	6 días	dom 08/04/12	vie 13/04/12
8	Activació d'alarma manual i indicació leds	3 días	sáb 12/05/12	mar 15/05/12
9	Programació WatchDog	3 días	mié 16/05/12	vie 18/05/12
10	Indicació existència de cobertura	3 días	vie 18/05/12	mar 22/05/12
11	Ensesa i apagada amb imant	4 días	mar 10/04/12	vie 13/04/12
12	Fer que l' Estació Base faci també de Sensor	6 días	lun 21/05/12	sáb 26/05/12
13	**Programació de la còsola de control**	**26,5 días**	**lun 09/04/12**	**mar 15/05/12**
14	Per mostrar l'alarma i dades corresponents	12 días	lun 09/04/12	mar 24/04/12
15	Per confirmar la recepció d'una alarma	6 días	jue 12/04/12	jue 19/04/12
16	Per configurar remotament paràmetres	6 días	vie 13/04/12	vie 20/04/12
17	Per mostrar temperatura i bateria dels nodes	6 días	sáb 14/04/12	vie 20/04/12
18	Per detectar mal funcionament del node	3 días	jue 10/05/12	mar 15/05/12
19	Lectura i enviament lluminositat	4 días	jue 03/05/12	mar 08/05/12
20	Interfície gràfica	23 días	mié 25/04/12	vie 25/05/12
21	**Preparació del software del TFC**	**49 días**	**mié 21/03/12**	**sáb 26/05/12**
22	Lliurament primera versió	1 día	mar 24/04/12	mar 24/04/12
23	Lliurament versió definitiva	1 día	sáb 26/05/12	sáb 26/05/12

Figura 3: Llistat de tasques de la programació final

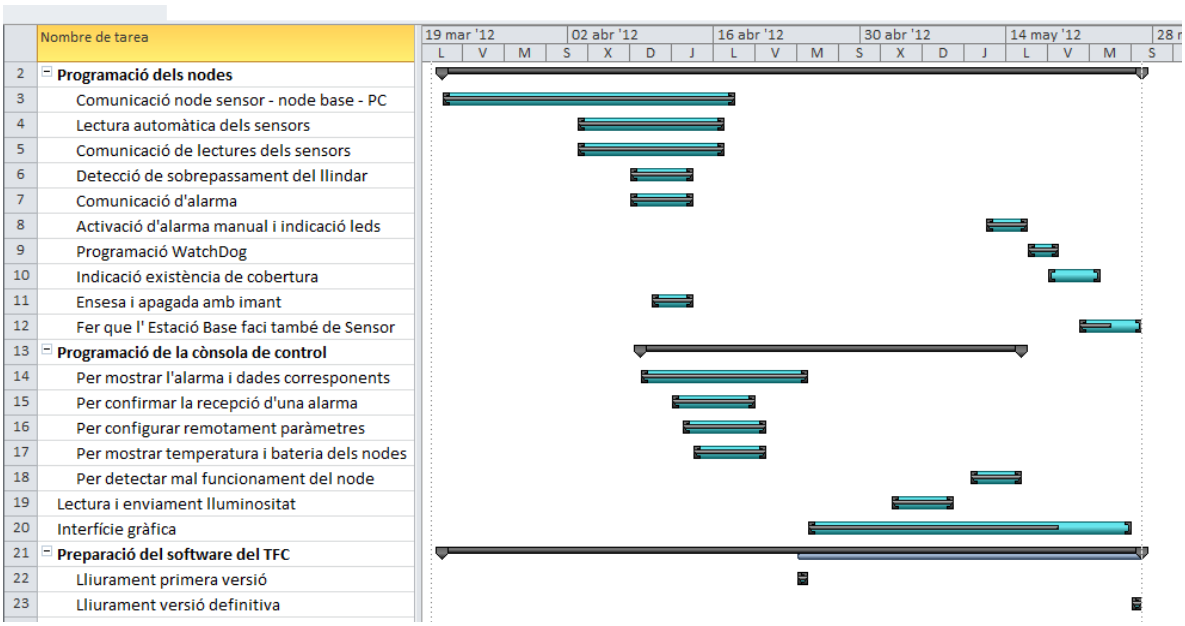

Figura 4: Diagrama de Gantt de la planificació final

1.5.3. Canvis el la planificació

En general es va complir la planificació prevista, només amb el disseny de la interfície gràfica quedaren relegades algunes de les tasques que encara restaven pendents però que finalment es feren dins el plaç. Cal esmentar una tasca no prevista inicialment i que per força s'havia de fer al final; és la de fer que el node estació base actués també com a sensor. A priori no semblava que calgués dedicar-hi molt de temps però la realitat no va ser aquesta i finalment no es pogué deixar acabat pel dia del lliurament. Aquestes modificacions queden reflectides en les figures 3 i 4.

1.6. Recursos emprats

Per desenvolupar aquest projecte s'han fet servir els següents recursos de maquinari i de programari:

1.6.1. Maquinari

Un ordinador amb el S. O. LinuxMint 12, Java 6.26 i el TinyOS 2.1.1 instal·lats.

Dues motes COU24 1_2 com les de la figura 5, basades amb el mòdul ZigBit ATZB-24-A2 de Atmel que incorpora el microcontrolador ATmega1281V i el transceptor AT86RF230. Per a la comunicació via radio fa servir un dels protocols de comunicació més utilitzats per aquest tipus de xarxes, el ZigBee, basat en l'estàndard IEEE 802.15.4 i pensat per a aplicacions que requereixen una taxa de transferència de dades baixa i la maximització de la durada de la bateria.

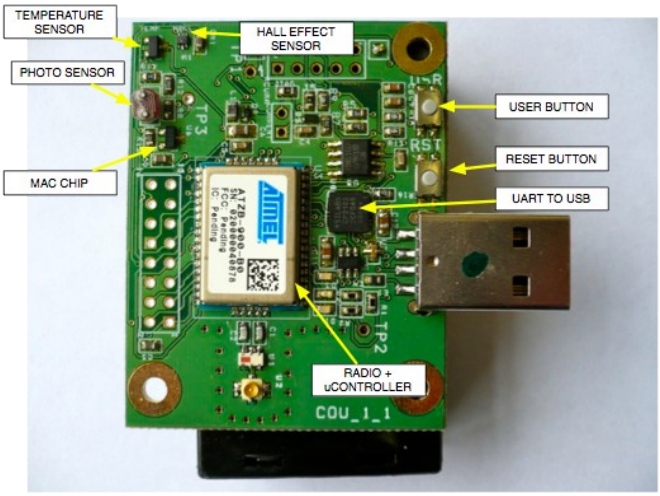

Figura 5: Mota Cou24 1_2

Les motes incorporen un sensor de temperatura MCP9700 de Microchip, un sensor d'efecte Hall BU52011HFT de ROHM i un fotosensor PDV-P9003-1 d'API connectats a distintes entrades ADC

del microcontrolador. Així mateix també disposen de tres leds, vermell, taronja i verd i d'un botó interruptor que es pot configurar per activar l'alarma de forma manual i un altre botó per reiniciar el dispositiu.

1.6.2. Programari

TinyOS 2.1.1 És un sistema operatiu de codi obert basat en components, per a xarxes de sensors sense fils. Està escrit en llenguatge de programació NesC i pensat per a funcionar sota les importants restriccions de memòria pròpies d'aquests tipus de sensors.

NetBeans 7.1.2 És un entorn de desenvolupament integrat lliure de programari, pensat especialment per a java, que incorpora el constructor intuïtiu d'interfícies gràfiques Matisse.

Eclipse Ganymede És la versió 3.4 de l'entorn de desenvolupament lliure de programari Eclipse. No és la darrera versió, però ens assegura plena compatibilitat amb el connector Yeti2 que s'integra en l'Eclipse per poder treballar també amb TinyOS.

Meshprog És una aplicació que serveix per a carregar el programa a la mota, un cop compilat. Se pot incorporar com a aplicació externa a l'Eclipse.

Serial Forwarder Aplicació que s'encarrega d'escoltar el port sèrie USB de la consola, on tenim connectada la mota Estació Base. Tots els paquets que arriben pel port sèrie els reenvia a un sòcol TCP i a la inversa.

1.7. Productes obtinguts

La implementació d'aquest projecte ha resultat en una sèrie de productes de programari:

L'aplicació **FireSensor** que és la que s'executa a la mota sensor. Està implementada en NesC i un cop compilada s'instal·la a la mota amb l'aplicació Meshpro. És la que s'encarrega de llegir els sensors, detectar i senyalitzar les alarmes, enviar i rebre missatges i de la resta de funcionalitats descrites anteriorment.

L'aplicació **FireAlarmJava** que és la que s'executa en el PC. Està implementada en *Java* i constitueix la interfície gràfica d'usuari. S'encarrega de mostrar la informació que li remeten les motes sensor, ja siguin alarmes o la lectura dels sensors. També serveix per enviar a les motes els paràmetres de configuració.

Altres productes obtinguts són la primera versió de **FireAlarmJava**, que no incorpora la interfície gràfica i serveix únicament per mostrar pel terminal els missatges rebuts de les motes, i l'aplicació **SetParameter** que es complementa amb l'anterior i serveix per enviar els paràmetres de configuració a les motes. Ambdues aplicacions estan implementades en *Java*.

1.8. Breu descripció de la resta de capítols de la memòria

Els següents capítols de la memòria parlen de:

- Capítol 2: Quins són els antecedents de la tecnologia WSN i quin és l'estat actual i quines aplicacions similars es poden trobar en el mercat.
- Capítol 3: Les decisions que preses a l'hora d'afrontar el disseny i la descripció funcional.
- Capítol 4: Descripció detallada de cada una de les parts del sistema.
- Capítol 5: Altres motes que es troben en el mercat i compatibilitat amb les emprades en aquest cas.
- Capítol 6: Viabilitat tècnica del projecte, durabilitat, operativitat.
- Capítol 7: Valoració econòmica.
- Capítol 8: Conclusions, objectius assolits i no assolits.
- Capítol 9: Glossari.
- Capítol 10: Bibliografia.
- Capítol 11: Annexos. Manual d'usuari de la consola i instruccions per compilar i instal·lar el programa a les motes.

2. Antecedents

Els orígens de les xarxes de sensors sense fils es remunten al programa DSN (xarxes de sensors distribuïts) del departament de defensa d'Estats Units l'any 1980. Tot i que la idea estava perfectament definida, la tecnologia encara no aportava el material necessari. Els sensors eren massa grossos, alguns eren com una capça de sabates i això limitava les aplicacions. A més, les primeres DSN no es varen associar preferentment amb la connectivitat sense fils.

No va ser fins a finals dels noranta, que els avenços tecnològics en informàtica, comunicacions, microelectrònica i transductors varen portar a un canvi significatiu en la investigació de les xarxes WSN i van permetre pensar en que es podrien aconseguir aquells objectius definits en la concepció original.

De llavors ençà l'interès en la tecnologia WSN no ha parat de créixer. Els avenços aconseguits, fruit de la cooperació internacional, en camps com el de les xarxes de comunicació, del processat de la informació en xarxa i especialment en el dels nodes sensors amb recursos limitats, han permès estendre'n l'ús civil a àrees com ara la vigilància del medi ambient, les xarxes de sensors en vehicles o en el cos humà.

L'any 2003 l'IEEE va definir l'estàndard IEEE 802.15.4 per a xarxes d'àrea local sense fils de baixa velocitat, i basant-se en aquest, la *ZigBee Alliance* va publicar a finals de 2004 l'estàndard *ZigBee* que especifica un conjunt de protocols de comunicació d'alt nivell per a xarxes de sensors sense fils.

Actualment WSN es percep com una de les tecnologies amb més futur. Països com la Xina han inclòs les xarxes de sensors sense fils dins els seus programes nacionals de recerca estratègica.

2.1. Estat de l'art

Una WSN està formada per un conjunt de sensors distribuïts en un espai. Cada un d'ells és capaç de realitzar de forma autònoma tasques de detecció, processament i transmissió de la informació a una unitat de processament centralitzat. Els components de maquinari habituals d'un node inclouen un transceptor de ràdio, un processador, memòria interna i pot ser externa i un o més sensors.

En el node, la funció del processador és programar les tasques, processar les dades i controlar el funcionament dels altres components de maquinari. D'entre les opcions que actualment trobaríem per a fer aquesta funció, microcontroladors, Processadors de Senyal Digital (DSP), Circuit Integrats per Aplicacions Específiques (ASIC), FPGAs etc. L'opció més àmpliament utilitzada entre les motes que actualment es troben en el mercat és la del microcontrolador, degut a la seva versatilitat per a connectar-se a altres dispositius i al seu baix preu. Per exemple, el més nou que podem trobar de Texas Instruments, el mòdul CC2531 utilitza el microcontrolador 8051 i en el cas de les motes

utilitzades en aquest projecte, porten un mòdul ATZB-24-A2 com el que podem veure a la figura 6, que incorpora un microcontrolador ATMega1281V.

Pel que fa a la comunicació sense fils, d'entre les distintes possibilitats que trobaríem, radiofreqüència, infrarojos, làser etc. la que s'adapta millor a la majoria d'aplicacions WSN és la radiofreqüència (RF). Actualment trobem per exemple el transceptor CC2520 de Texas Instruments o l'AT86RF230 d'Atmel, que és el que porta integrat el mòdul ATZB-24-A2 en les motes COU24.

Figura 6: Mòdul ATZB-24-A2

La memòria en els nodes es pot quedar només en la que porten integrada els microcontroladors, per exemple el nostre ATMega1281V porta 128 KB de memòria flaix i 8 KB de RAM, o es pot ampliar amb memòria externa. El xip AT45DB041B d'Atmel permetria afegir 4Mb de memòria flaix addicional.

En les xarxes WSN el consum d'energia dels nodes és un aspecte crític, ja que han de poder funcionar el màxim temps possible de forma autònoma. Per tant la minimització del consum ha de ser un objectiu que s'ha de tenir sempre present tant en el disseny dels components com en la gestió de l'energia mitjançant la programació. Normalment les motes s'alimenten amb bateries, la majoria de les que actualment es troben ho fan amb dues bateries tipus AA com és el cas de la COU24. En l'actualitat es treballa per a obtenir energia del medi ambient (solar, eòlica) per alimentar les motes i fer-les així del tot autònomes.

Un altre dels components de les motes que també ha evolucionat molt són els sensors. Aquests elements produeixen un senyal elèctric en resposta a un canvi mesurable en una magnitud física tal com la temperatura, la pressió, la humitat, la quantitat de llum etc. El senyal analògic detectat pels sensors es digitalitza i processa en el microcontrolador. Com que el node disposa d'una font d'energia limitada, els sensors connectats, a part de tenir una mida petita, han de tenir un consum d'energia extremadament baix. La mota COU24 porta un sensor de temperatura MCP9700 de Microchip com el que podem veure a la figura 7, un sensor d'efecte Hall BU52011HFT de ROHM i un fotosensor PDV-P9003-1 d'API. Tots ells constitueixen un petit exemple dels molts que podem trobar en el mercat.

Figura 7: Sensor de temperatura MCP9700

Pel que fa als protocols emprats, l'estàndard IEE 802.15.4 especifica la capa física i la capa MAC per a xarxes sense fils d'àrea personal i velocitats de transmissió baixes. Les característiques principals són:

- Velocitats de transmissió de 20, 40 i 250 Kbps
- Dos modes d'adreçament: curt de 16 bits i adreçament IEEE de 64 bits
- Suport per a dispositius de latència crítica, com ara palanques de comandament “joysticks”
- Accés al canal CSMA-CA (Carrier Sense, Multiple Acces, Collision Avoidance) Accés múltiple per detecció de portadora amb evitació de col·lisions
- Establiment automàtic de la xarxa pel coordinador
- Protocol amb conformitat de connexió “handshaking” per a total fiabilitat de la transferència
- Administració d'energia per assegurar un baix consum energètic
- 16, 10 i 1 canals respectivament a les bandes ISM de 2,4 GHz 915 MHz i 868 MHz

És la base per a les especificacions *ZigBee* i *WirelessHART* que ofereixen una solució de xarxa completa mitjançant el desenvolupament de les capes superiors que no estan cobertes per la norma IEE 802.15.4.

ZigBee va ser concebut i actualment és mantingut per la *ZigBee Alliance,* un gran consorci d'indústries. Els camps d'aplicació típics de *ZigBee* inclouen: monitorització d'energia intel·ligent, monitorització de constants clíniques, control remot, automatització d'edificis, domòtica, etc. La pila de protocols corresponent es veu a la figura 8.

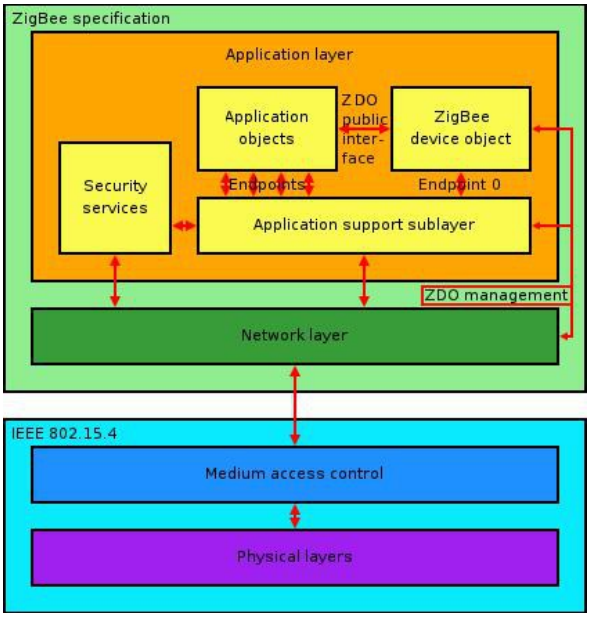

Figura 8: Pila del protocol ZigBee

WirelessHART és un protocol estàndard obert. A la capa d'enllaç de dades utilitza la tecnologia TDMA (Accés Múltiple per Divisió de Temps) per arbitrar i coordinar les comunicacions entre els dispositius. Proporciona comunicacions segures mitjançant l'ús d'algoritmes de xifrat AES-128. És compatible amb la capa d'aplicació de l'estàndard HART (Highway Addressable Remote Transducer) i és compatible per tant amb els instruments, aplicacions i tecnologia d'integració de sistemes HART. Altres característiques que cal remarcar són l'excel·lent fiabilitat i l'escalabilitat.

2.2. Estudi de mercat

Actualment hi ha en el mercat alguns sistemes semblants a l'aplicació desenvolupada, un dels més semblants és el sistema de la companyia francesa **CLEODE** *www.cleode.fr*.

Com a sensor de foc utilitza el dispositiu **ZGas** de la figura 9, que detecta fum i monòxid de carboni. Funciona sense fils i es comunica amb l'estàndard ZigBee. Funciona amb una pila de liti de 9V i té un consum en repòs inferior a 0,1 mA.

Figura 9: Detector de foc sense fils ZGas

Com a estació base utilitza el dispositiu USB **Ubee** de la figura 10, que incorpora com a transceptor el xip CC2530 de Texas Instruments.

Figura 10: Node estació base Ubee

L'aplicació per a PC **CleoBee** permet als usuaris veure els dispositius de la xarxa ZigBee i interactuar-hi sense tenir cap coneixement de programació. Permet veure una representació topogràfica de la xarxa, com es veu a la figura 11, a part de mostrar i configurar els paràmetres de cada dispositiu i definir els escenaris de comportament.

Figura 11: Interfície d'usuari CleoBee

Un altre sistema similar, però més obert és el desenvolupar per l'empresa Libelium, creada com a companyia Spin Off de la Universidad de Zaragoza. El seu producte WaspMote de la figura 12 és un dispositiu modular i versàtil al que es poden acoblar una cinquantena de sensors distints. El mòdul transceptor ZigBee també pot ser distint en funció de la potència necessària.

Figura 12: Node sensor WaspMote de Libelium

Figura 13: Node estació base de Libelium

Amb aquests dispositius l'empresa DIMAP ha desenvolupat i instal·lat el sistema de Seguiment i Vigilància Ambiental (SISVIA), basat en el desplegament d'una xarxa de sensors sense fils per a la monitorització contínua de variables ambientals, amb l'objectiu d'actuar com a dispositiu de detecció d'incendis forestals.

2.3. Sectors a qui va dirigit el sistema desenvolupat

El sistema desenvolupat en aquest projecte està pensat per a instal·lacions contra incendis en recintes o espais relativament petits ja que al tractar-se d'una xarxa amb configuració en estrella no es poden instal·lar nodes molt allunyats de l'estació base.

Una altra característica important és que cal vigilància permanent, ja que les alarmes només es visualitzen en el propi sensor que l'ha disparada i en la consola de control.

Aquests dos condicionants limiten els llocs on es podria instal·lar el sistema; si que serviria per a grans superfícies, recintes i edificis oficials , centres d'oci, museus etc.

3. Descripció funcional

Abans d'iniciar la descripció funcional de cada una de les parts, val la pena comentar algunes decisions preses a l'hora de fer el disseny sobre qüestions que, o be no venien especificades en els requisits o no hi quedaven definides amb total precisió.

Després d'iniciar l'aplicació de la consola de control, aquesta roman a l'espera de que arribi algun missatge de qualque node de la xarxa. El moment que això succeeix crea una instància de la classe sensor per a aquesta mota, on hi anirà guardant tots els paràmetres de configuració, les lectures, l'estat, les alarmes i un registre de tots els missatges que arriben de la mota o s'hi envien.

En encendre la mota, el primer que fa és enviar un missatge a la consola per saber en quin estat es troba. Això es fa així perquè hi ha la possibilitat de que la mota estàs activa i ara s'estigui reinicialitzant després de que hagi actuat el WatchDog. Si fos així, la consola li envia tots els paràmetres de configuració i es queda activa. Si no estava activa o no rep cap resposta de la consola roman inactiva.

En el moment que s'activa la mota s'activa el temporitzador de lectura del sensor de temperatura, pel qual s'han fixat un valor mínim d'un segon i un valor màxim de set segons. Aquest valor màxim ve condicionat pel fet de que cada cop que s'activa aquest temporitzader es refresca el temporitzador del WatchDog, el qual té un període de vuit segons. Cada vegada que es llegeix el sensor de temperatura es llegeixen també el nivell de bateria i el fotosensor. El motiu de fer-ho d'aquesta manera és que un període màxim de lectura de set segons és prou petit com per no haver de fer lectures específiques a l'hora d'enviar el nivell de bateria o totes les dades pel *debug*.

Pels temporitzadors d'enviament de nivell de bateria i de dades també s'han fixat uns valors màxim i mínim. El mínim en ambdós casos és el valor que s'hagi fixat pel temporitzador de lectura de la temperatura, que és el que marca la lectura de tots els sensors; i el valor màxim és de dos i cinc minuts respectivament. Si no es volen rebre totes les dades perquè no cal depurar el sistema *debug*, el valor que s'ha d'introduir per aquest paràmetre és 0, que és el valor per defecte.

Quan s'envien totes les dades, no tan sols s'envia el valor de la temperatura, sinó que s'envien totes les lectures i tots els paràmetres de configuració de la mota. Quan es modifica qualque paràmetre, la mota envia un missatge amb totes les dades que serveix a l'operador de la consola per confirmar que el canvi s'ha produït.

3.1. Sistema total

El sistema està format per un PC que actua com a interfície d'usuari i una xarxa de sensors sense fils que té una topologia en estrella, de tal forma que els nodes sensors es podran comunicar únicament amb el node estació base i aquest amb el PC via port USB, sent la comunicació bidireccional. La figura 14 en representa un esquema general.

Figura 14: Esquema funcional del sistema

El funcionament general del sistema es basa en que cada node sensor va llegint de forma periòdica el sensor de temperatura, el nivell de la bateria i el fotosensor. Si en qualque moment la temperatura arriba a superar una temperatura llindar establerta, o si el nivell de bateria baixa per sota d'un nivell preestablert, farà saltar una alarma que es mostrarà mitjançant els leds en el node i també s'enviarà a l'estació base, per tal de mostrar-la en el PC.

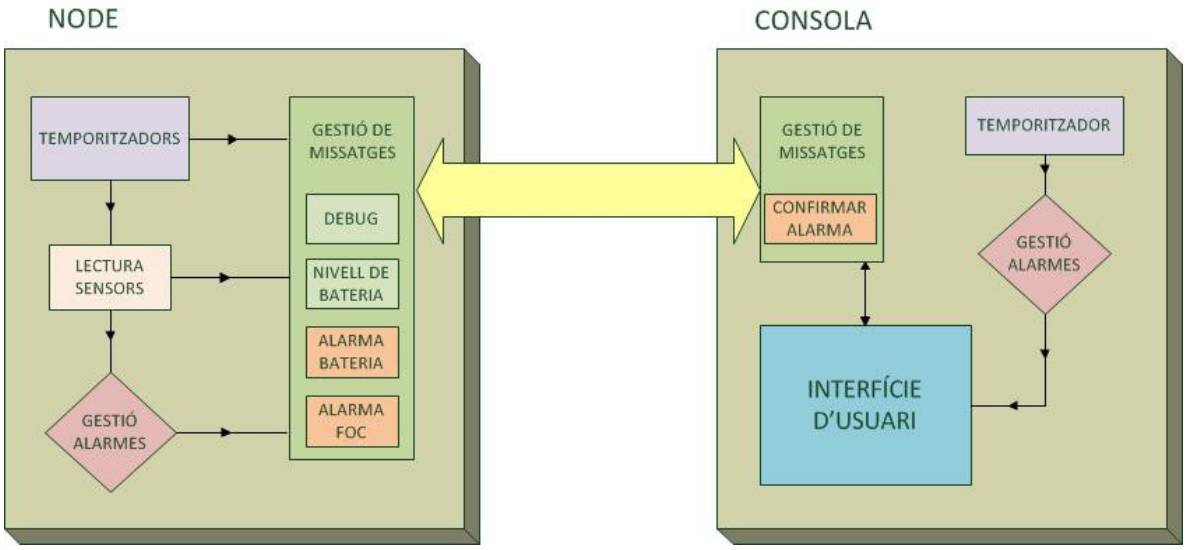

Figura 15: Esquema de blocs del sistema

Des de la consola es poden veure els paràmetres i les dades llegides de cada sensor. També es poden configurar els paràmetres i si es dona el cas, confirmar la recepció del missatge d'alarma de foc. A la figura 15 es pot veure un esquema de blocs del sistema.

3.2. PC, Interfície d'usuari

En el PC s'executa l'aplicació que fa la funció d'interfície d'usuari. Està implementada amb el llenguatge de programació Java del qual s'utilitzen les llibreries Swing per a la interfície gràfica.

Inicialment no hi ha cap sensor donat d'alta a l'aplicació. Quan rep un missatge el primer que fa és comprovar si el node sensor del qual prové ja està registrat, si no ho està el registra.

Després, actuarà d'una o altra manera segons els missatge rebut, tal com mostra la figura 16.

StateReqMsg: Missatge que sempre envia un node sensor quan es reinicia per conèixer l'estat en el que es trobava abans de reiniciar-se. En rebre'l la consola li envia un missatge StateMsg amb tota la informació de configuració.

BatMsg: Missatge que envia el node sensor periòdicament perquè la consola de control tingui constància de que continua actiu. En rebre'l la consola reinicia el temporitzador que vigila la activitat de cada mota. Si per a un node sensor aquest temporitzador arriba al final, la consola mostra una alarma indicant que s'ha perdut la comunicació amb la mota.

DataMsg: Quan es rep aquest tipus de missatge es rep tota la informació d'un node sensor, tant la lectura de la temperatura, nivell de bateria i lluminositat com els paràmetres actuals de configuració. Quan el rep, la consola actualitza totes les dades que guarda d'aquell node.

BatAlarmMsg: És el missatge d'alarma de bateria. Qual el rep, la consola mostra la indicació en pantalla de l'alarma amb l'hora a la que s'ha produït.

FireAlarmMsg: És el missatge d'alarma de foc. Qual el rep, la consola mostra la indicació en pantalla de l'alarma amb l'hora a la que s'ha produït. També activa la tecla per enviar el missatge de confirmació d'alarma rebuda cap a la mota.

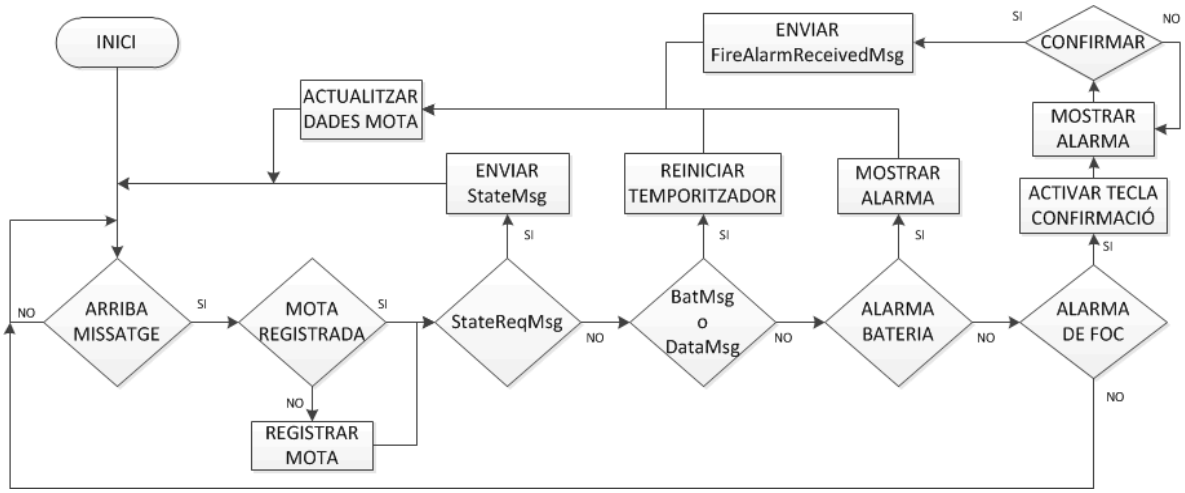

Figura 16: Diagrama de flux de les tasques de monitorització de la consola

La interfície gràfica de la consola de control mostra els valors de la temperatura, dels nivells de bateria i lluminositat. Els dos primers són de color verd quan els paràmetres que monitoritzen estan per sota del llindar, i canvien a vermell quan el sobrepassen. Quan els mostra en color gris vol dir que en el darrer missatge rebut no hi havia informació per actualitzar la dada i que, per tant el valor pot no ser l'actual. Això succeeix amb la dada de temperatura i lluminositat quan només es reben missatges de nivell de bateria.

La interfície també ens dona més informació, per exemple, quins són els valors actuals dels paràmetres de configuració o quin és l'estat de la mota:

Missatge	**Significat**
Sensor inactiu	Sensor registrat a la consola de control, però en stand-by
Sensor actiu	Sensor registrat a la consola de control que transmet dades
Alarma manual HH:MM:SS	Alarma manual activada a l'hora HH:MM:SS
Alarma automàtica HH:MM:SS	Alarma automàtica activada a l'hora HH:MM:SS
Bateria baixa	Tensió de la bateria per sota del 90% del valor nominal
Sensor perdut	Perduda la comunicació amb el sensor

L'aspecte de la interfície gràfica dissenyada és la que es pot veure a la figura 17.

Figura 17: Interfície gràfica d'usuari. Monitorització

A la interfície, s'han agrupat dins la pestanya Configuració els controls que permeten configurar els distints paràmetres del node sensor. Quan introduim un nou valor per a un paràmetre, l'aplicació comprova que estigui dins els valors previstos. Si el valor està fora dels marges permesos indica un error. Si el valor és correcte envia un missatge a la mota que segons el paràmetre a modificar serà:

TempThdMsg: Missatge per a fixar una nova temperatura llindar d'alarma de foc per al node sensor.

TempTimerMsg: Missatge per a fixar un nou període de lectura de la temperatura al node sensor.

BatTimerMsg: Missatge per a fixar un nou període d'enviament de la lectura del nivell de bateria.

DataTimerMsg: Missatge per a fixar un nou període d'enviament de totes les dades del node sensor, lectures de la temperatura, del nivell de bateria, de la lluminositat i tots els paràmetres de configuració per tal de que l'usuari pugui verificar el correcte funcionament.

DataReqMsg: És un missatge que s'envia automàticament després d'un missatge de modificació de la configuració. Amb aquest missatge li demanam al node destinatari que envii un missatge DataMsg amb tota la informació, lectures i configuració, per comprovar que els canvis en la configuració s'han fet correctament.

Aquestes opcions es poden veure representades en la figura 18.

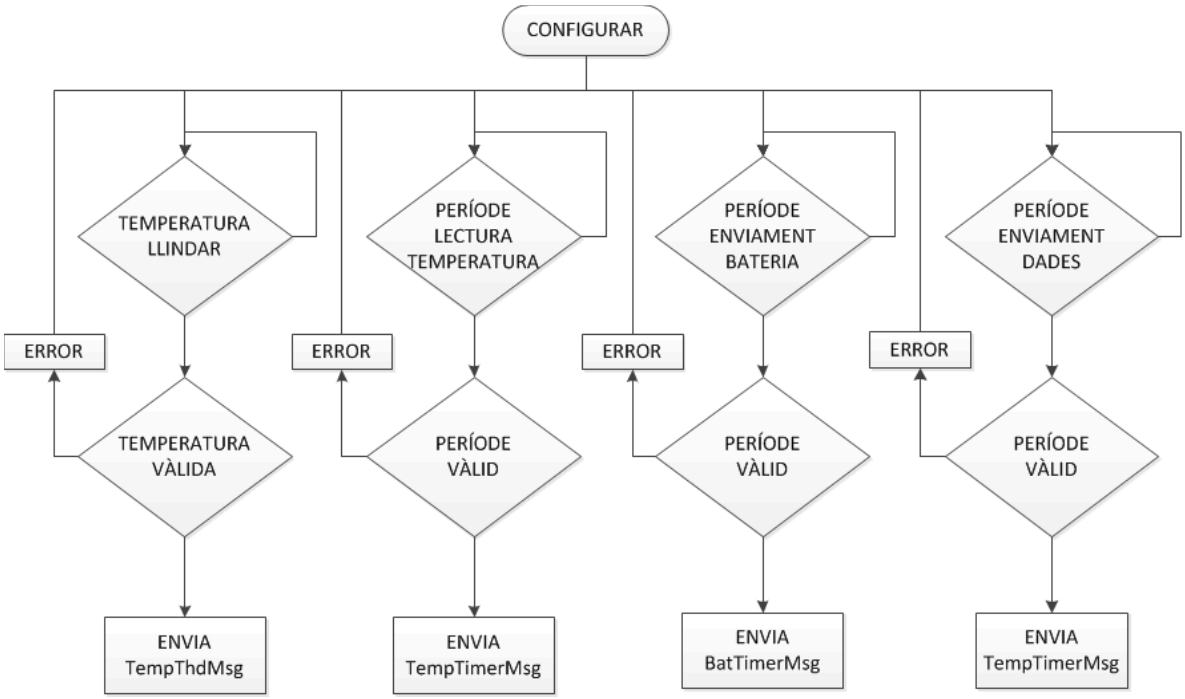

Figura 18: Diagrama de flux de les tasques de configuració de la consola

Els valors vàlids per als distints paràmetres són:

Paràmetre	Unitats	Rang de valors
Temperatura llindar d'alarma	ºC	30 - 100
Període lectura temperatura	ms	1000 - 7000
Període enviament bateria	ms	Període lectura temperatura - 120000
Període enviament dades	ms	Període lectura temperatura - 300000 o bé 0

3.3. Node Sensor

Quan es connecta l'alimentació a una mota, el primer que fa és encendre la radio i enviar un missatge **StateReqMsg** a la consola de control. Com ja s'ha comentat, això ho fa perquè es podria donar el cas que la mota s'estàs recuperant després d'una substitució de les bateries o després de que hagués actuat el WatchDog. Si aquest és el cas, la consola de control la te registrada com a mota activa i guarda tots els seus paràmetres de configuració. Si el missatge ha arribat a la consola, en breus instants rebrà un missatge StateMsg que li dirà si ha de quedar inactiva o si ha de recuperar l'activitat i amb quins paràmetres de configuració ho ha de fer. Si al cap de 3 segons no ha rebut resposta es queda inactiva.

La mota s'activa passant dues vegades seguides un imant per sobre del sensor d'efecte Hall, i es desactiva passant-lo quatre vegades.

Al activar la mota s'habilita el WatchDog amb una temporització de 8 segons i s'engeguen els temporitzadors d'enviament del nivell de bateria i de lectura de la temperatura, el qual podrà tenir un període màxim a 7 segons ja que es fa servir també per reiniciar el WatchDog.

Cada vegada que es fa la lectura de la temperatura s'aprofita també per fer les lectures del nivell de bateria i del fotosensor. Amb les noves dades es comprova si hi ha qualque nivell que transgredeixi els llindars establers i calgui activar alguna alarma.

Si la temperatura sobrepassa la de llindar d'alarma la mota envia el missatge **FireAlarmMsg** a la consola. Aquest enviament es fa amb ACK de maquinari amb el node Estació Base, i amb ACK de programari amb la consola.

Quan la consola rep un missatge d'alarma de foc, automàticament contesta amb el missatge **FireAlarmAckMsg** a la mota, fent-li saber que l'alarma l'hi ha arribat.

L'alarma d'incendi també se senyalitza a la mota amb el led vermell encès de forma intermitent. Un sol pols curt indica alarma detectada i dos polsos curts seguits indica que l'alarma ja ha estat rebuda en la consola de control.

En aquest punt, l'usuari té l'opció de confirmar la recepció de l'alarma des del terminal. Si ho fa, la consola envia el missatge **FireAlarmReceivedMsg**. En una aplicació real concreta, aquesta acció activaria tots els dispositius d'alerta i extinció. En aquest prototip, el missatge arriba a la mota que ha generat l'alarma, la qual atura tots els temporitzadors i senyala aquest fet amb el led vermell lluint sense intermitència. Tal com ha quedat el disseny, aquesta circumstància dura fins que el WatchDog, al cap de vuit segons, reinicialitza la mota.

Senyalització de l'alarma de foc en els nodes sensors		
Alarma detectada	**Alarma rebuda en estació central**	**Alarma reconeguda**
Led vermell intermitent. 1 pols curt	Led vermell intermitent 2 polsos curts seguits	Led vermell encès sense intermitència
		

Si el que es detecta és alarma de bateria la mota envia el missatge **BatAlarmMsg** a la consola. L'enviament es fa amb ACK de maquinari fins al node estació base. Si no rep el justificant de recepció ACK repetirà l'enviament fins a 20 vegades.

El sensor indica localment aquesta circumstància amb dos polsos curts seguits del led taronja de forma intermitent.

Si amb la lectura dels sensors no s'ha produït cap alarma, la mota periòdicament anirà enviant el valor del nivell de bateria amb el missatge **BatMsg**. Si aquest missatge és el que serveix a la consola per saber que un sensor continua actiu, en el sensor ens serveix per indicar aquesta mateixa circumstància mitjançant el led verd. Cada cop que s'envia el missatge el led verd s'encén de forma intermitent, un pols curt.

La tramesa es produeix amb requeriment de justificant de recepció ACK. Si després d'enviar el missatge rep l'ACK de l'estació base, en lloc d'un pols, el led verd s'encén dos polsos curts seguits, senyalitzant també d'aquesta forma que està dins la zona de cobertura de l'estació base.

Quan la mota està inactiva ho indica amb el led taronja lluint de forma intermitent un pols curt cada dos segons.

Senyalitzacions en els nodes sensors			
Alarma de bateria	**No alarma**	**No alarma + Cobertura**	**Sensor apagat**
Led taronja intermitent 2 polsos curts seguits	Led verd intermitent 1 pols curt	Led verd intermitent 2 polsos curts seguits	Led taronja intermitent 1 pols curt cada 2 s
			

A la mota es guarden en variables els valors dels distints paràmetres susceptibles de ser modificats. Quan s'inicia la mota s'hi guarden els valors per defecte. Posteriorment des de la consola es poden modificar aquests valors amb els missatges que vists en l'apartat anterior.

En aquest cas els valors per defecte són:

Temperatura llindar d'alarma	Nivell de bateria llindar d'alarma	Període lectura temperatura	Període enviament bateria	Període enviament dades
30 ºC	2,7 V	2 s	5 s	0 = no s'envia

En la figura 19 podem veure una representació en forma de diagrama de flux de l'aplicació que s'executa en el node sensor.

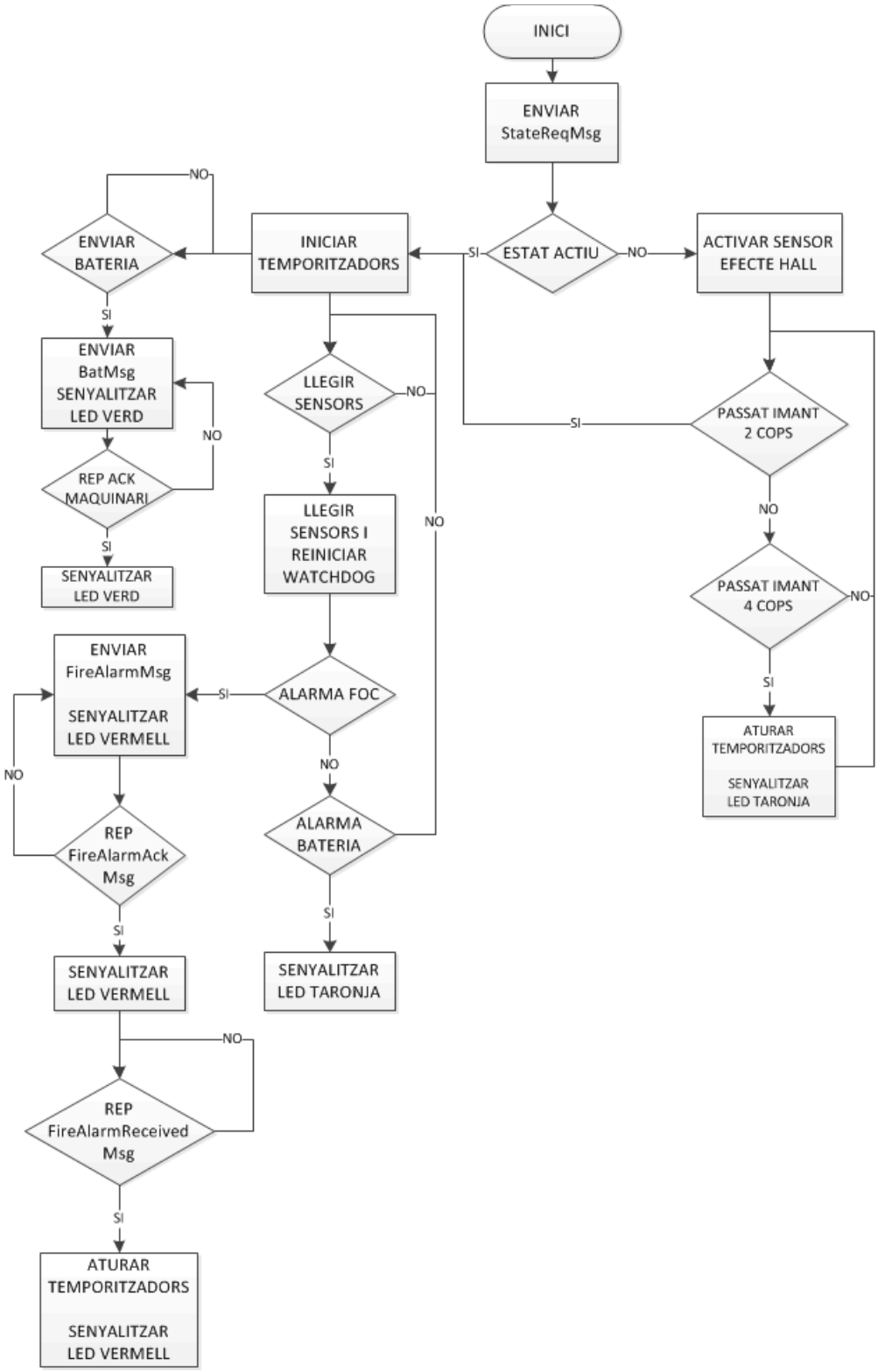

Figura 19: Diagrama de flux de l'aplicació del node sensor

3.4. Node Estació Base

El node Estació Base és el que fa de pont entre la consola de control i la xarxa de nodes sensors. En aquest node s'executa l'aplicació BaseStation que el que fa és enviar cap al port sèrie USB tots els missatges que li arriben via radio i al revés, enviar via radio tots els missatges que li arriben del port sèrie.

L'aplicació fa parpellejar el led vermell cada vegada que envia un paquet via radio i el led taronja cada vegada que envia un paquet al port serie. El led verd parpelleja quan es perd qualque paquet degut a que els rep a més velocitat que la que els pot enviar.

No ha calgut implementar aquesta aplicació ja que es troba entre el conjunt d'aplicacions que venen en l'arbre del sistema operatiu TinyOS.

4. Descripció Detallada

Tal com s'ha vist fins ara, l'objecte principal d'aquest projecte ha estat el disseny de l'aplicació que s'executa a la consola de control i la que s'executa en el node sensor.

4.1. Consola de control

L'aplicació de la consola de control s'ha implementat en Java. S'ha creat la classe **Sensor** de la qual se'n crea una instància per a cada un dels nodes sensors de la xarxa. A cada una s'hi guarden, actualitzats, els valors de tots els paràmetres i de les lectures, un registre de tots els missatges que s'han rebut o s'hi han enviat amb anotació del dia i hora i també un temporitzador per detectar si es deixa de rebre el missatge de nivell de bateria amb la periodicitat correcte, per si cal, generar l'alarma de pèrdua de comunicació amb la mota.

La classe principal és la **FireAlarmJava** que implementa la interfície **MessageListener**. S'encarrega de gestionar la recepció i l'enviament de missatges així com de crear i mantenir actualitzades les instàncies de **Sensor**.

La classe **controlConsole** és la que proporciona la interfície gràfica. Tota la informació que mostra l'obté de la instància del sensor corresponent.

Cada vegada que arriba un missatge d'un node, **FireAlarmJava** mira si ja en té una instància creada a la llista de sensors i si no hi és la crea. Després n'actualitza les dades i finalment crida el procediment *updateFrame(Sensor)* de la **controlConsole** per a que actualitzi les dades que està mostrant.

Per cada tipus de missatge que arriba o s'envia a la mota s'ha creat la classe corresponent. Totes i cada una d'aquestes classes hereten de la classe **Message** i proporcionen els mètodes *get* i *set* per a cada un dels paràmetres que conté el missatge.

L'estructura de cada un d'ells és la següent:

```
typedef nx_struct DataMsg
        nx_uint8_t      moteId;
        nx_uint16_t     countsTemp;
        nx_uint16_t     countsPhoto;
        nx_uint16_t     countsBat;
        nx_uint16_t     vrefmilivolts;
        nx_uint16_t     tempThd;
        nx_uint32_t     tempTimer;
        nx_uint32_t     batTimer;
        nx_uint32_t     debugTimer;

typedef nx_struct TempMsg
        nx_uint8_t      moteId;
        nx_uint16_t     countsTemp;
        nx_uint16_t     vrefmilivolts;

typedef nx_struct BatMsg
        nx_uint8_t      moteId;
        nx_uint16_t     countsBat;
        nx_uint16_t     vrefmilivolts;

typedef nx_struct TempThdMsg
        nx_uint16_t     countsTemp;

typedef nx_struct TempTimerMsg
        nx_uint32_t     tempTimerPeriod;

typedef nx_struct BatTimerMsg
        nx_uint32_t     batTimerPeriod;

typedef nx_struct DataTimerMsg
        nx_uint32_t     dataTimerPeriod;

typedef nx_struct FireAlarmReceivedMsg
        nx_bool         recAlarm;

typedef nx_struct FireAlarmMsg
        nx_uint8_t      moteId;
        nx_bool         manualAlarm;
        nx_uint16_t     countsTemp;
        nx_uint16_t     vrefmilivolts;
```

```
typedef nx_struct BatAlarmMsg
        nx_uint8_t      moteId;
        nx_uint16_t     countsBat;
        nx_uint16_t     vrefmilivolts;

typedef nx_struct FireAlarmAckMsg
        nx_bool         ackAlarm;

typedef nx_struct StateReqMsg
        nx_uint8_t      moteId;

typedef nx_struct StateMsg
        nx_bool         onStby;
        nx_uint16_t     tempThd;
        nx_uint32_t     tempTimerPeriod;
        nx_uint32_t     batTimerPeriod;
        nx_uint32_t     debugTimerPeriod;

typedef nx_struct DataReqMsg
        nx_uint8_t      seq;

typedef nx_struct RssiMsg
        nx_int16_t      rssi;
```

Una altra de les coses que permet la interfície d'usuari és configurar una sèrie de paràmetres de la mota. A la pestanya *Configuració* hi apareixen els valors que tenen actualment en uns quadres de text que es poden editar. Quan introduïm un nou valor i polsam la tecla *Actualitzar*, el primer que fa és comprovar que el valor és correcte i si ho és crida a una de les funcions que proporciona la classe **FireAlarmJava** per enviar el missatge de configuració al node.

setTempThreshold(double Temp)	Per modificar la temperatura llindar
setTempTimer(int teTi)	Per modificar el període de lectura de la temperatura
setBatTimer(int baTi)	Per modificar el període d'enviament del nivell de bateria
setDebugTimer(int daTi)	Per modificar el període d'enviament de les dades

Un altre dels procediments que crida és *confirmAlarm()* que serveix per enviar el missatge de confirmació de la recepció d'una alarma de foc.

S'ha dit abans que cada instància de **Sensor** guarda una relació de tots els missatges que s'intercanvien la consola de control i el node sensor al que representa. Ho fa creant, per cada un, una instància de la classe **SensorMessage** on hi guarda l'hora i el dia, el missatge i si ha arribat o s'ha enviat. Aquesta instància la guarda després a la llista *messages*.

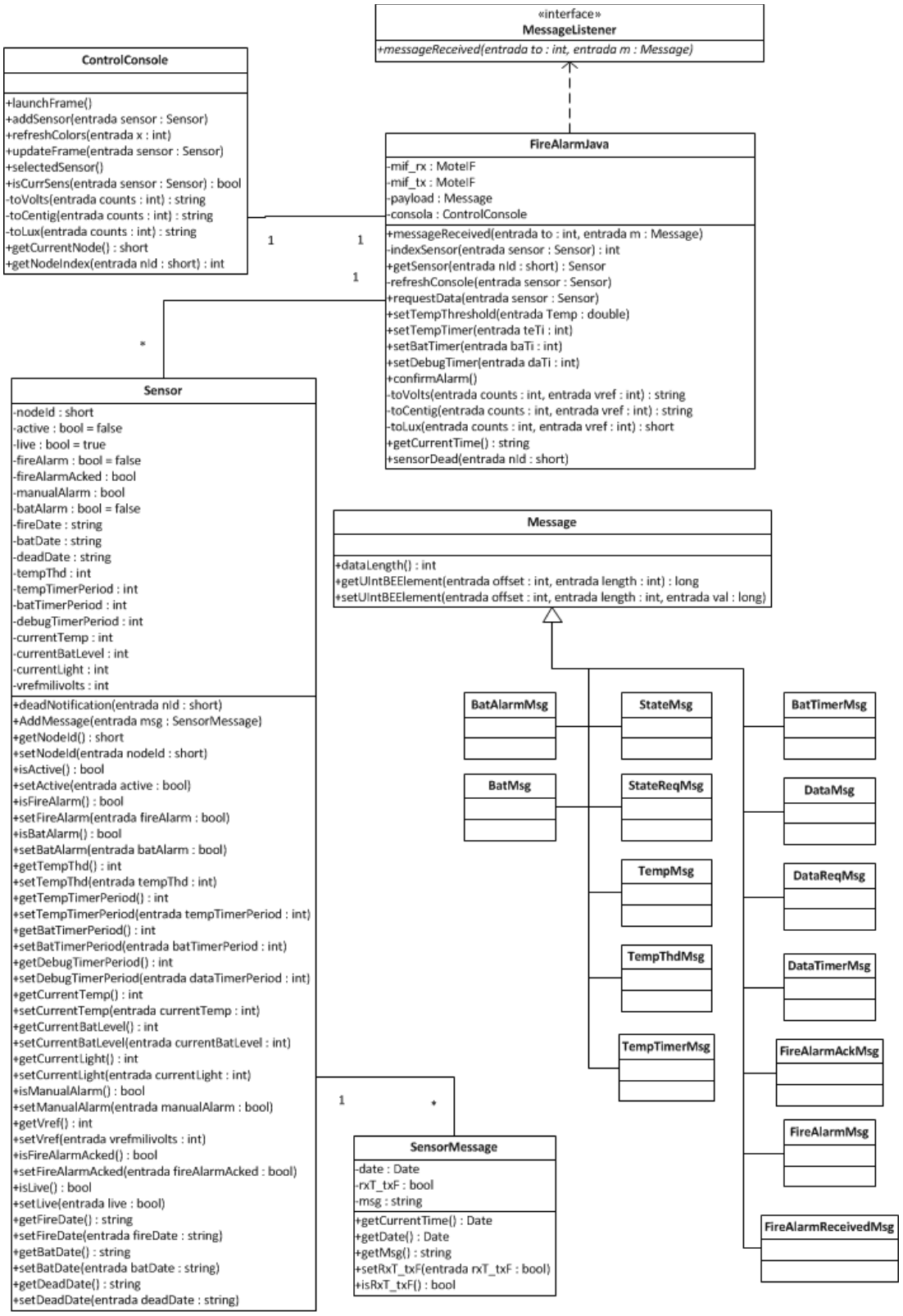

Figura 20: Diagrama de classes de l'aplicació de la consola

4.2. Node Sensor

L'aplicació **FireSensor** és l'encarregada de fer que la mota actuï com a node sensor. L'aplicació s'ha implementada amb nesC per al sistema operatiu TinyOS i s'han creat una sèrie de components amb les configuracions, els mòduls i les interfícies corresponents per a separar les distintes tasques que ha de realitzar:

FireSensorAppC Especifica la configuració del component principal, que és el que s'encarrega d'activar els temporitzadors, d'ordenar la lectura dels sensors, de comprovar els valors llegits amb els llindars per veure si cal generar alguna alarma, de senyalitzar les alarmes amb els leds etc. Aquest component s'implementa amb el mòdul FireSensorC i fa servir un grapat de components ja implementats a la llibreria de TinyOS que es poden veure a la figura 21:

MainC: Proveeix la interfície *Boot* que genera l'esdeveniment *booted* que ens serveix per a iniciar l'aplicació.

LedsC: Que proveeix la interfície *Leds* i permet encendre i apagar els leds.

TimerMilliC(): Que proveeix els distints temporitzadors.

ActiveMessageC: Amb la interfície *powerSerie* que ens permet encendre i apagar la radio.

HplAtm128GeneralIOC Amb la interfície *ButtonPin* per assignar un port del microcontrolador al botó d'alarma manual.

HplAtm128InterruptC Amb la interfície *ButtonInt* per assignar una interrupció del microcontrolador al botó d'alarma manual.

Apart d'aquests també fa servir els components nous implementats per a aquesta aplicació:

SensorsReaderC S'encarrega d'activar els sensors i llegir-los.

ComControlC S'encarrega de la comunicació, enviar i rebre missatges.

HallEffectC S'encarrega d'activar el sensor d'efecte Hall i detectar el pas de l'imant.

WchDogC S'encarrega de gestionar el WatchDog.

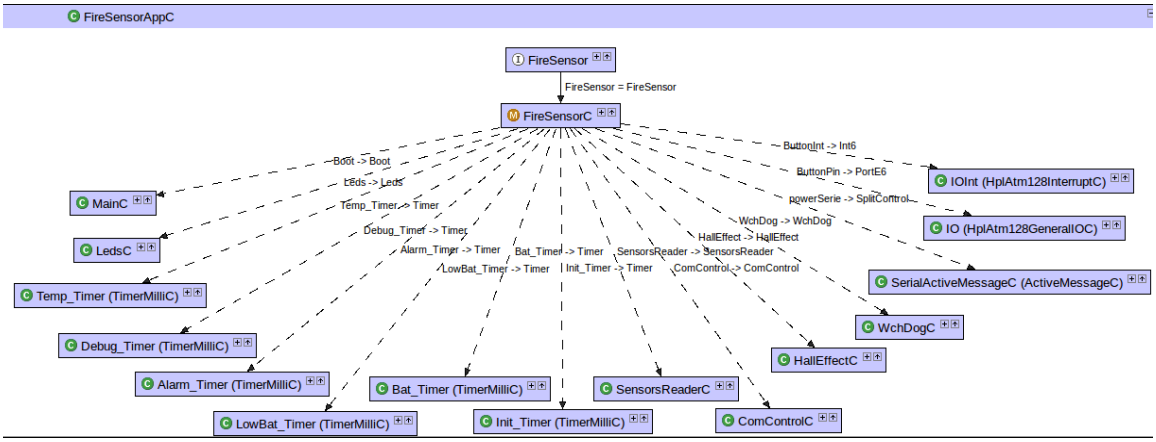

Figura 21: Diagrama de components de FireSensorApp

El mòdul **FireSensorC** és el que implementa aquest component. Proveeix la interfície **FireSensor** i a través d'ella les següents funcionalitats:

turnOn(): Encén el node, que en aquest cas vol dir activar els temporitzadors de lectura de la temperatura, d'enviament del nivell de bateria i també habilitar el WatchDog.

turnOff(): Aturar els temporitzadors i el WatchDog.

newValues(): Compara els nous valors llegits amb els llindars per veure si es produeix un cas d'alarma.

sendData(): Activa l'ordre d'enviar les dades.

setTempThd(uint16_t): Assigna un nou valor al paràmetre temperatura llindar.

getTempThd(): Retorna el valor actual del paràmetre temperatura llindar.

setTempTimer(uint32_t): Assigna un nou valor al paràmetre període de lectura de la temperatura.

getTempTimer(): Retorna el valor actual del paràmetre període de lectura de la temperatura.

setBatTimer(uint32_t): Assigna un nou valor al paràmetre període d'enviament del nivell de bateria.

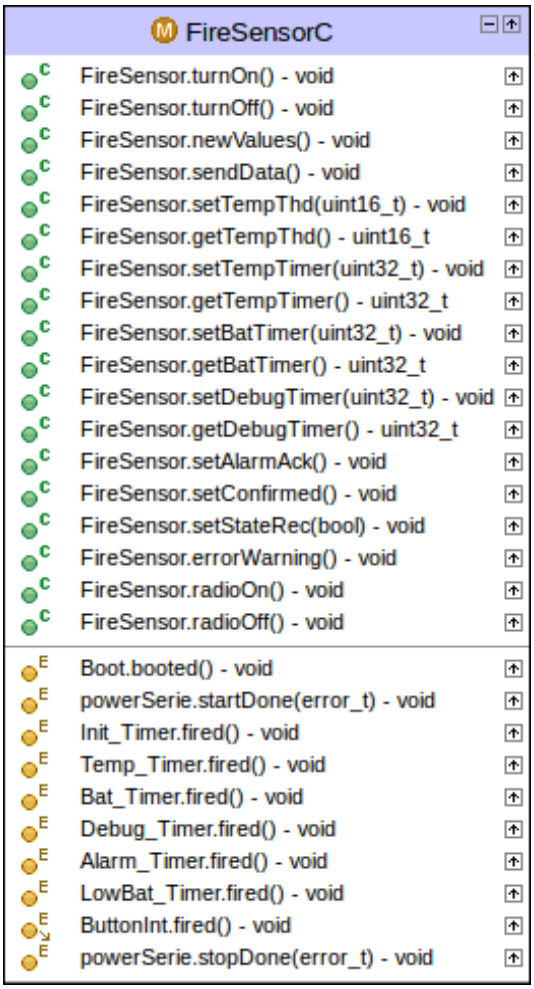

Figura 22: Mòdul FireSensorC

getBatTimer(): Retorna el valor actual del paràmetre període d'enviament del nivell de bateria.

setDebugTimer(uint32_t): Assigna un nou valor al paràmetre període d'enviament de les dades.

getDebugTimer(): Retorna el valor actual del paràmetre període d'enviament de les dades.

setAlarmAck(): Posa a cert el valor de la variable d'estat que indica que el missatge d'alarma de foc ha arribat a la consola.

setConfirmed(): Posa a cert el valor de la variable d'estat que indica que el missatge d'alarma de foc ha estat confirmat per l'usuari.

setStateRec(): Posa a cert el valor de la variable d'estat que indica que està en estat de recuperació després d'una aturada no prevista.

errorWarning(): Tractament d'errors.

radioOn() i **radioOff()**: Encén i apaga la radio respectivament.

Així mateix també preveu el codi que s'ha d'executar quan es produeix un esdeveniment generat per alguna de les interfícies d'altres components que fa servir. Són els que figuren al final de la llista de la figura 22.

SensorsReaderC és la configuració del component que s'encarrega d'activar els sensors i llegir-los. La implementació d'aquest component es concreta en el mòdul **SensorsReaderP** i fa servir els següents components de la llibreria de TinyOS representats en l'esquema de la figura 23:

TimerMilliC(): Que proveeix un temporitzador.

AdcReadClientC(): Proveeix l'accés als ports analògics del microcontrolador mitjançant la interfície *AdcReading*.

HplAtm128GeneralIOC: Que proveeix les interfícies d'assignació i configuració dels ports d'entrada i sortida del microcontrolador.

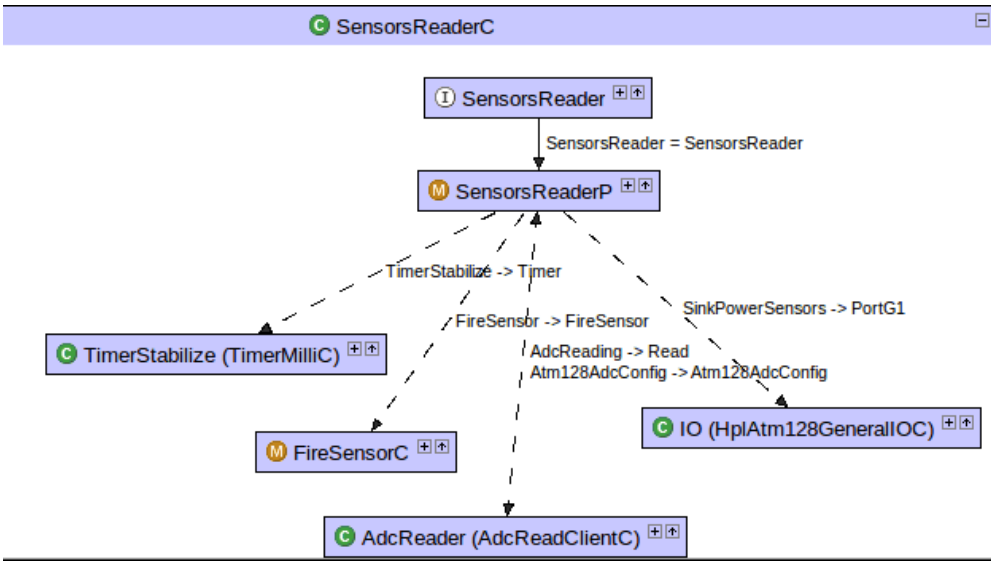

Figura 23: Diagrama de components de SensorsReaderC

El mòdul **SensorsReaderP** és el que implementa aquest component i a través de la interfície **SensorsReader** proveeix les següents funcionalitats, resumides en la llista de la figura 24:

ReadSensors(): Inicia la tasca que s'encarrega de fer la lectura dels sensors.

getChannel(): Retorna el port analògic que s'ha de llegir depenent de si es vol mesurar la temperatura, el nivell de bateria o la quantitat de llum.

getTemp(): Retorna el valor de temperatura llegit.

getPhoto(): Retorna el valor mesurat amb el fotosensor.

getBat(): Retorna el nivell de bateria mesurat.

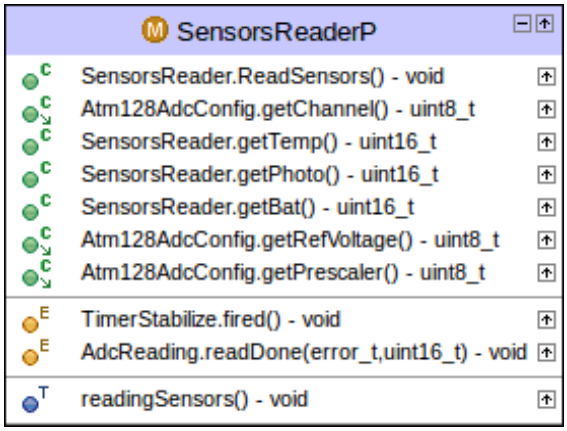

Figura 24: Mòdul SensorsReaderP

getRefVoltage(): Retorna el valor de la tensió de referència emprada per l'ADC.

getPrescaler(): Retorna el factor d'escala de l'ADC.

La tasca **readingSensors()** és la que se'n cuida d'activar el sensors, llegir-los un darrera l'altra i guardar els valors llegits. Finalment quan ja els ha llegit tots els desactiva i crida el comando *newValues()* de la interfície *FireSensor*.

ComControlC és la configuració del component que s'encarrega de la comunicació, d'enviar i rebre missatges. La implementació d'aquest component es fa en el mòdul **ComControlP** i fa servir els següents components de la llibreria de TinyOS representats el l'esquema de la figura 25:

LedsC: Que proveeix la interfície *Leds* i permet encendre i apagar els leds.

TimerMilliC(): Que proveeix temporitzadors.

AMSenderC(): Permet enviar missatges via radio amb les interfícies Packed i AMSend.

AMReceiverC(): Permet rebre missatges via radio amb la interfície Receive.

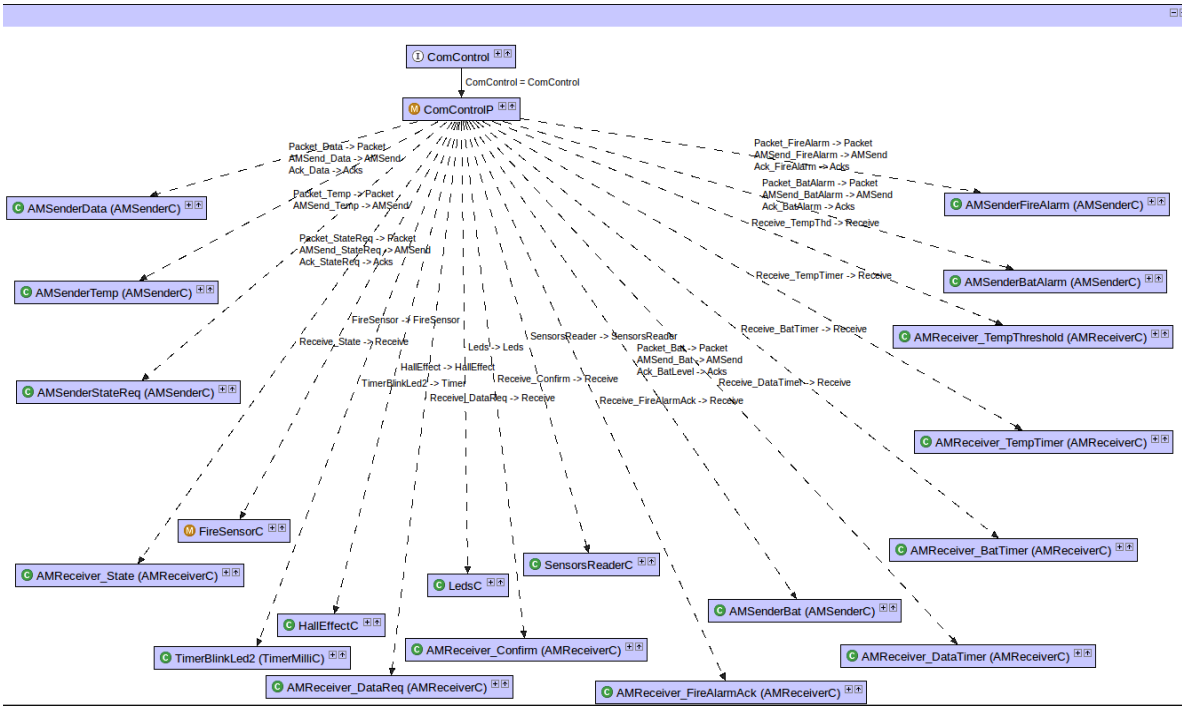

Figura 25: Diagrama de components de ComControlC

El mòdul **ComControlP** a traves de la interfície **ComControl** proveeix les següents funcionalitats:

sendStateReq(): Crida la tasca sendStateReqMsg() que envia el missatge a la consola per a saber en quin estat s'ha de posar, actiu o inactiu.

resetSender(): Fa que deixi de persistir en l'enviament d'un missatge.**sendData()**: Crida la tasca **sendDataMsg()** que envia el missatge amb totes les dades del node sensor a la consola.

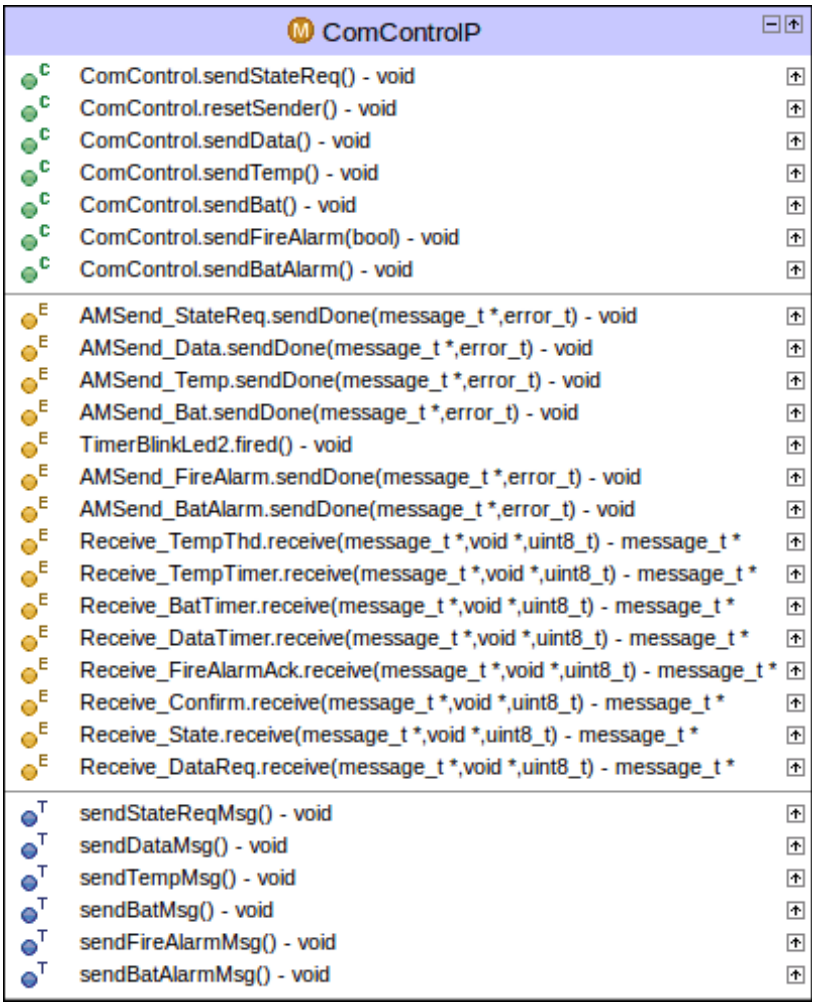

Figura 26: Mòdul ComControlP

sendTemp(): Crida la tasca **sendTempMsg()** que envia el missatge amb la lectura de la temperatura a la consola.

sendBat(): Crida la tasca **sendBatMsg()** que envia el missatge amb el nivell de bateria a la consola.

sendFireAlarm(): Crida la tasca **sendFireAlarmMsg()** que envia el missatge d'alarma de foc a la consola.

sendBatAlarm(): Crida la tasca **sendBatAlarmMsg()** que envia el missatge d'alarma de bateria a la consola.

També preveu el codi que s'ha d'executar quan es produeix un esdeveniment generat per alguna de les interfícies d'altres components que fa servir i que també apareixen en la llista de la figura 26.

HallEffectC és la configuració del component que s'encarrega de gestionar l'estat inactiu del node. Activa el sensor d'efecte Hall i roman a l'espera que es produeixi una interrupció deguda al pas d'un imant per damunt el sensor. Quan això succeeix, inicia un comptador i un temporitzador de dos segons. Si abans de que transcorri aquest temps es produeixen altres interrupcions degudes al pas de l'imant, el temporitzador es reinicia i el comptador s'incrementa. Quan transcorren més de dos segons sense que s'hagi produït cap interrupció, si el comptador val dos s'activa la mota i si val quatre es desactiva. El led vermell canvia d'estat cada vegada que es produeix una interrupció.

El mòdul **HallEffectP** és el que implementa aquest component que fa servir els següents components de la llibreria de TinyOS, representats en l'esquema de la figura 27:

LedsC: Que proveeix la interfície *Leds* i permet encendre i apagar els leds.

TimerMilliC(): Que proveeix temporitzadors.

HplAtm128GeneralIOC: Que proveeix les interfícies d'assignació i configuració dels ports d'entrada i sortida del microcontrolador.

HplAtm128InterruptC Per assignar una interrupció del microcontrolador al sensor d'efecte Hall.

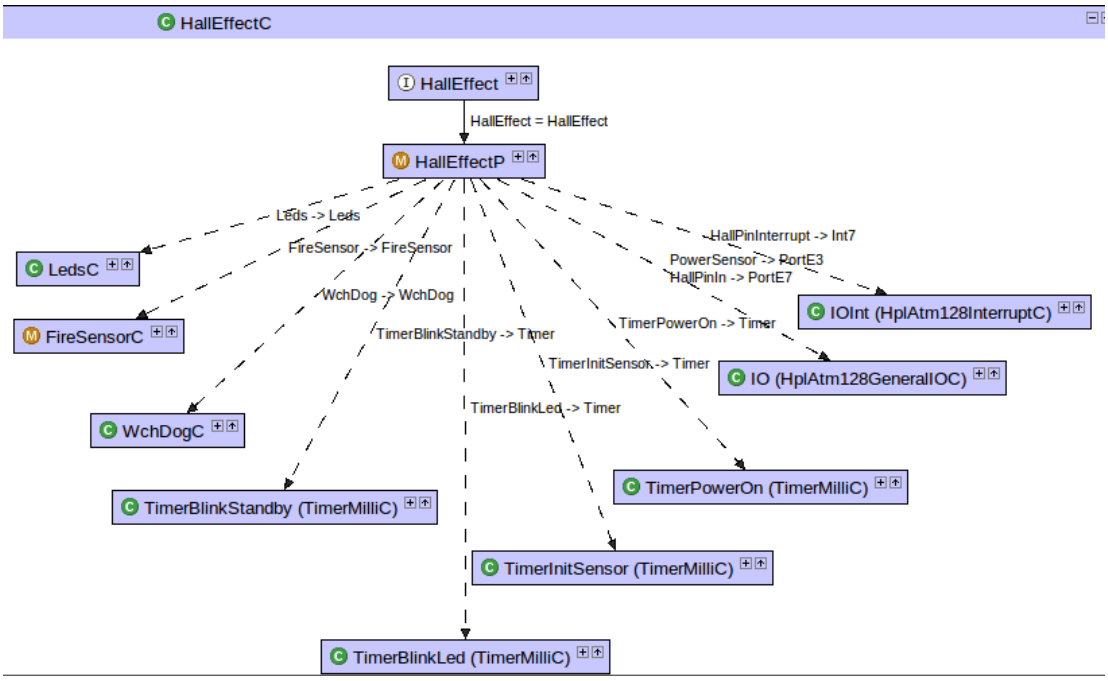

Figura 27: Diagrama de components de HallEffectC

El mòdul **HallEffectP** a través de la interfície **HallEffect** proveeix les següents funcionalitats:

EnableHall(): Activa el sensor d'efecte Hall. Configura les entrades corresponents del microcontrolador i inicia un temporitzador de tres segons per assegurar l'estabilització del

transductor. Transcorregut aquest temps es queda en l'estat inactiu indicant-ho amb el led taronja lluint un període curt cada dos segons, tot esperant l'activació amb l'imant.

La tasca **onOff()** reinicia el temporitzador de dos segons per comptar les vegades que es passa l'imant per damunt el sensor.

També preveu el codi que s'ha d'executar quan es produeix un esdeveniment generat per alguna de les interfícies d'altres components que fa servir i que també apareixen en la llista de la figura 28.

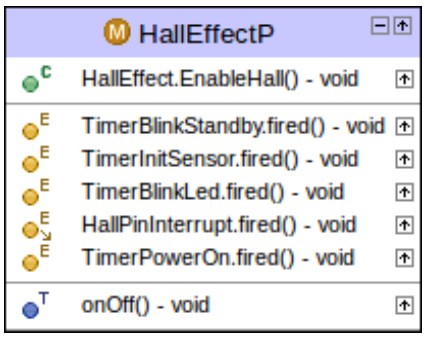

Figura 28: Mòdul HallEffectP

Finalment, **WchDogC** és la configuració del component que gestiona el WatchDog. El mòdul **WchDogP** l'implementa i amb la interfície **WchDog** proveeix les següents funcionalitats:

stop(): Desactiva el WatchDog.

start(uint32_t): Activa el temporitzador del WatchDog amb la temporització indicada.

reset(): Reinicia el temporitzador del WatchDog.

Figura 30: Mòdul WchDogP

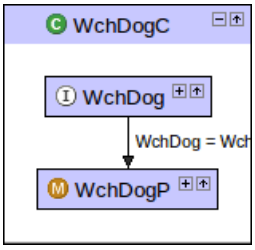

Figura 29: Components de WchDogC

5. Altres motes i compatibilitat

La llista de nodes sensors sense fils existents en el mercat és cada dia més llarga. Bàsicament, el que distingeix una plataforma d'una altra és el microcontrolador i el transceptor que utilitzen.

Les motes BitBean™ , figura 32, empren el mateix bloc ATZB-24-A2 de Atmel i les IRIS, figura 31, un bloc basat amb el mateix microcontrolador i el mateix transceptor, per tant serien les més compatibles.

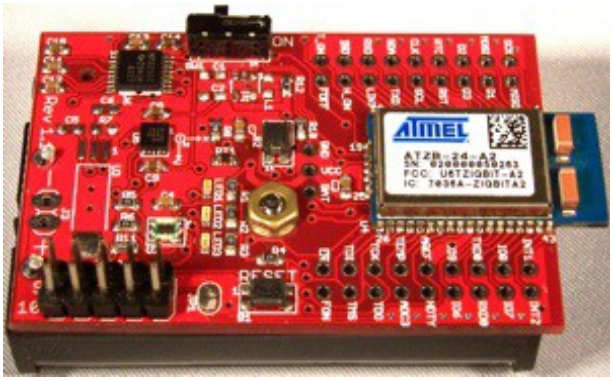

Figura 32: Mota BitBean

Figura 31: Mota IRIS

Altres plataformes com la MICAz, figura 34, es basen en el microcontrolador ATmega128L, pràcticament compatible, però en aquest cas el transceptor està basat en el CC2420 de Texas Instruments. La Waspmote, figura 33, empra l'ATmega1281 i com a transceptor un mòdul Xbee.

Figura 34: Mota MICAz

Figura 33: Mota WaspMote

Altres plataformes, com la TelosB, figura 35, i la Kmote, figura 36, es basen en el microcontrolador MSP430 i el transceptor CC2420, ambdós de Texas Instruments.

Figura 35: Mota TelosB

Figura 36: Mota KMote

6. Viabilitat tècnica

El prototip desenvolupat pràcticament compleix tots els requisits inicials del projecte però, per afirmar que és viable tècnicament cal abans verificar una sèrie de factors:

- Durabilitat: Per fer el prototip s'han utilitzat les motes COU24, unes motes genèriques que caldria adequar per a que poguessin instal·lar-se com a sensors, amb una càpsula o sistema de protecció que assegurassin la durabilitat del maquinari. Pel que fa al programari, aquest tipus de mota permet una fàcil actualització si es deixa accessible el port USB, per tant això n'asseguraria la vigència davant canvis de normativa o d'exigències de la instal·lació.
- Operativitat: Al tractar-se d'un sistema de seguretat, cal aconseguir que sigui totalment fiable, que mai doni falses alarmes i que mai es passi per alt una situació d'alarma real. Per això l'aplicació s'hauria de depurar al màxim per tal que no es produïssin casos imprevists. També calen millores per aconseguir un sistema contra incendis més efectiu, s'haurien d'afegir als nodes altres tipus sensors com ara detectors de fum o de flama. Altres millores que el farien més operatiu podrien ser incorporar nodes actuadors per disparar sistemes d'extinció automàtica, o per avisar via GSM de l'alarma.

Com s'ha pogut veure en l'apartat d'estudi de mercat, hi ha sistemes similars a aquest, que incorporen millores com les que acabam d'esmentar, que estan implementats i es comercialitzen. Per tant es pot considerar que el projecte és tècnicament viable.

7. Valoració econòmica

El que resulta més imprecís a l'hora de fer la valoració econòmica del disseny és el cost de desenvolupament, ja que una bona part del temps que s'hi ha dedicat ha servit per adquirir uns coneixements generals de la tecnologia WSN de xarxes de sensors sense fils, del sistema operatiu TinyOS i del llenguatge de programació nesC. Aquest cost no es pot imputar tot a aquest projecte, ja que el normal seria que aquest bagatge servís també per a projectes posteriors.

Suposant que s'ha de fer una instal·lació de 10 sensors amb un contracte de manteniment de dues revisions anuals.

Concepte	Unitats	Preu unitari	Total
Hores desenvolupament projecte	200	20	4.000
Motes TelosB CM5000 Advanticsys	11	77	847
PC amb Linux	1	350	350
Hores instal·lació	12	20	320
Hores manteniment anual (2 revisions)	2 x 6 = 12	20	320
TOTAL			**5.837** €

8. Conclusions

8.1. Assoliment d'objectius

Si repassam la llista d'objectius que s'havien d'assolir, per tal de complir els requisits exigits, veiem:

Pel que fa a l'alarma de foc, el sensor llegeix periòdicament el sensor de temperatura, compara el valor llegit amb la temperatura llindar d'alarma i si el sobrepassa envia de forma segura missatges d'alarma de foc a la consola de control, senyalitza l'alarma amb els leds i per altra part també té habilitat un botó per activar l'alarma manualment. El missatge d'alarma que envia el node inclou el valor de la temperatura, la consola identifica el node que li envia l'alarma, guarda l'hora a la que li ha arribat i mostra tota la informació per pantalla. També la consola envia de forma automàtica un missatge de reconeixement de l'alarma de foc i habilita l'usuari per confirmar-la. El sensor senyalitza totes aquestes circumstàncies amb els leds.

Si analitzam ara els objectius relacionats amb l'alarma de bateria, la mota llegeix periòdicament el nivell de bateria, el compara amb el nivell llindar i si està per sota envia de forma segura missatges d'alarma de bateria a la consola de control. També de forma segura i amb una periodicitat

reconfigurable, envia missatges a la consola, amb el nivell mesurat, per tal de que aquesta constati l'activitat de la mota. Per la seva part, la consola mostra el nivell de bateria de la mota i si es produeix alarma ho indica juntament amb l'hora que s'ha produït. La mota també ho indica amb els leds.

Un altre objectiu assolit és que la mota s'activi i es desactivi passant l'imant per damunt el sensor d'efecte Hall, dues o quatre vegades seguides respectivament. També el WatchDog evita que la mota es quedi penjada i si fos així recuperaria la configuració.

L'usuari pot fer des de la consola que la mota envii periòdicament totes les lectures i les dades de configuració i de la mateixa manera pot modificar tots els paràmetres de configuració, temperatura llindar, freqüència de la lectura i dels enviaments.

Un altre dels requisits inicials era proporcionar un sistema per comprovar si on s'instal·la el node hi ha cobertura amb el node base. Malgrat si ens atenem estrictament a l'enunciat, l'objectiu estaria complit no ho està de la forma que segurament preveia l'enunciat. Cada cop que la mota envia un missatge de nivell de bateria ho fa de forma segura amb ACK de maquinari i el led verd parpelleja una vegada. Si rep el justificant de recepció ACK de l'estació base torna parpellejar, per tant si tenim el node estació base en marxa, just activar el node sensor començarà a enviar missatges de nivell de bateria. Si el parpelleig del led verd és doble podem assegurar que on ens trobem hi ha cobertura. La forma esperada de fer-ho era utilitzant la dada de nivell de RF que proporciona el transceptor.

L'enunciat especificava que el node estació base havia de fer també la funció de node sensor i això no s'ha aconseguit. Calia esperar a que el node sensor tingués totes les funcionalitats operatives per després implementar-les en el node base, per tant era una de les tasques a deixar pel final. Després però no va resultar tan fàcil com semblava implementar les dues funcions en el node i el temps es va esgotar sense aconseguir-ho.

8.2. Proposta de Millores

Tal com s'ha exposat en l'apartat d'estudi de la viabilitat tècnica, caldria incorporar algunes millores per a que el sistema fos més efectiu, s'haurien d'afegir als nodes altres tipus sensors com ara detectors de fum o de flama i guanyaria operativitat si s'incorporassin nodes actuadors per disparar sistemes d'extinció automàtica o per avisar via GSM de l'alarma. Una altra millora podria ser crear una interfície d'usuari per a dispositius mòbils.

8.3. Autoavaluació

De les dues vessants que té el treball de final de carrera, la de implementació possiblement m'ha arribat a absorbir massa temps, pot ser per la dificultat d'enfrontar-me a temes nous, com són el sistema operatiu TinyOS i el llenguatge de programació nesC o per l'interessant que em resultava i no volia donar una passa sense sabes exactament el que estava fent. No vaig ser conscient de l'altra vessant fins que a la primera PAC s'ens va demanar que lliuréssim el pla de treball del TFC . Pot ser per arribar a assolir més pràctica i coneixements de gestió, documentació i presentació m'hagués calgut un projecte més senzill d'implementar.

De totes formes no em penedeixo gens d'haver triat aquesta àrea, ans al contrari, pot ser m'ha agradat massa i també he de dir que malgrat no ser el primer TFC que faig, l'altra ja en fa molts d'anys, no recordo haver vist mai el tema de gestió de projectes, tot plegat molt interessant.

9. Glossari

- ACK : Justificant de recepció
- ADC : Convertidor analògic digital
- Eclipse IDE : Entorn integrat de desenvolupament de codi obert multiplataforma
- GUI : Interfície gràfica d'usuari
- IEEE 802.15.4 : Estàndard que defineix el nivell físic i el control d'accés al medi de xarxes sense fils d'àrea personal amb taxes baixes de transmissió de dades.
- Java : Llenguatge de programació orientat a l'objecte.
- Led : Díode electroluminescent
- Messhprog : Aplicació emprada per a programar microcontroladors a traves d'USB.
- Microcontrolador : Microprocessador especialitzat en controlar equips electrònics que inclou en un sol xip les tres unitats funcionals d'un ordinador, CPU, memòria i unitat d'entrada i sortida.
- Mota : Node sensor d'una xarxa de sensors sense fils.
- NesC : Variant del llenguatge de programació C optimitzat per a les limitacions de memòria dels sensors.
- Sensor : Dispositiu capaç de transformar magnituds físiques en senyals elèctrics.
- SerialForwarder : Programa que llegeix i escriu paquets de dades en un port sèrie i els remet a través d'una connexió TCP/IP, actuant com un servidor intermediari, de manera que altres aplicacions s'hi poden connectar en lloc de connectar-se al port sèrie.
- TinyOS : Sistema operatiu de codi obert basat en components per a xarxes de sensors sense fils.
- Transceptor : Dispositiu que realitza, dins un mateix encapsulament, funcions tant de transmissió com de recepció, utilitzant components i circuits comuns per ambdues funcions.
- USB : Estàndard de bus que, mitjançant l'ús d'un sol cable simplifica la integració automàtica dels perifèrics que es connecten a un ordinador.
- WSN : Xarxes de sensors sense fils.
- ZigBee : Nom de l'especificació d'un conjunt de protocols d'alt nivell de comunicació sense fils, basat en l'estàndard IEEE 802.15.4, pensat per a la comunicació segura en sistemes amb baixa taxa d'enviament de dades i minimització del consum.

10. Bibliografia

Advanced Photonix. Documentació del fotosensor PDV-P9003-1. Versió en PDF disponible a:
<http://www.advancedphotonix.com/ap_products/pdfs/PDV-P9003-1.pdf>

Atmel. Documentació del microcontrolador Atmega1281V. Versió en PDF disponible a:
<http://www.atmel.com/Images/doc2549.pdf>

Atmel. Documentació del transceptor AT86RF230. Versió en PDF disponible a:
<http://www.atmel.com/Images/doc5131.pdf>

ETH Zurich. Documentació del plugin Yeti2 per a l'entorn de programació Eclipse per treballar amb TinyOS disponible en línia.
<http://tos-ide.ethz.ch/wiki/index.php>

Levis, Philip; Gay, David (2009). *TinyOS Programming*. Cambridge: Cambridge University Press

Microchip. Documentació del sensor de temperatura MCP9700. Versió en PDF disponible a:
<http://ww1.microchip.com/downloads/en/DeviceDoc/21942e.pdf>

Rohm. Documentació del sensor d'efecte Hall BU52011HFT. Versió en PDF disponible a:
<http://www.rohm.com/products/databook/sensor/pdf/bu52001gul-e.pdf>

Termcat. Base de dades de terminologia disponible en línia.
<http://www.termcat.cat/>

TinyOS. Documentació de la versió 2.x de TinyOS i de les llibreries ce components i interfícies de les distintes plataformes disponible en línia.
<http://www.tinyos.net/tinyos-2.x/doc/>

TinyOS. Javadoc de TinyOS en línia.
<http://www.tinyos.net/dist-2.0.0/tinyos-2.x/doc/javadoc/>

TinyOS. Tutorial de TinyOS disponible en línia.
<http://docs.tinyos.net/tinywiki/index.php/TinyOS_Tutorials>

UOC. Informació sobre la mota COU24 i com programar-la disponible en línia.
<http://cv.uoc.edu/app/mediawiki14/wiki/**>**

11. Annexos

11.1. Manual d'usuari

Abans d'iniciar la interfície gràfica del sistema contra incendis cal connectar el node Estació Base a un port USB del PC i executar l'aplicació SerialForwarder que és un programa que ens fa de pont entre la consola i el port USB on hi hem connectat la mota.
Ho farem teclejant la següent comanda:

java net.tinyos.sf.SerialForwarder -comm serial@/dev/ttyUSB0:19200

Quan s'executa s'obri una finestra com aquesta en la que hem de configurar, si no ho estan, els paràmetres tal com es mostren en la figura 37:

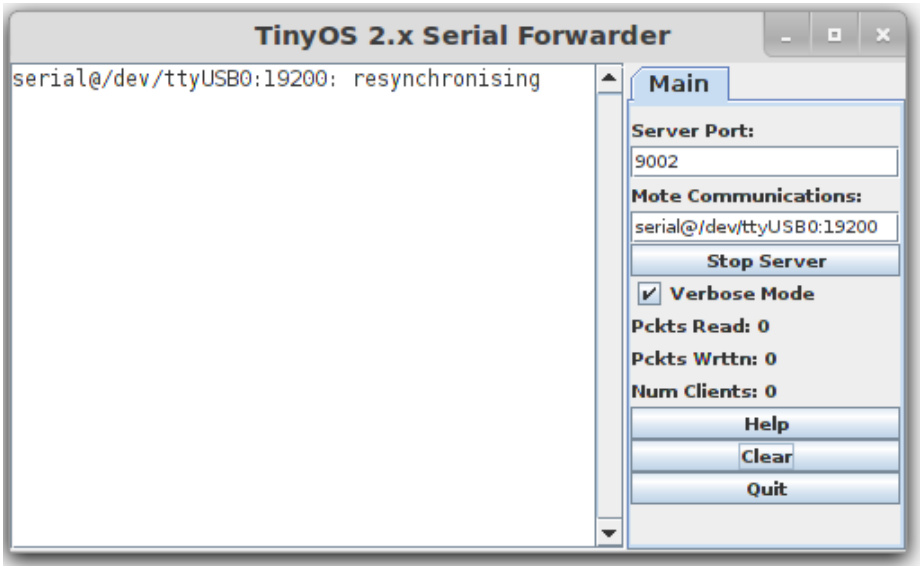

Figura 37: Configuració de l'aplicació SerialForwaeder

Ara ja podem executar l'aplicació FireAlarm. Des d'un terminal obert en el directori on tenim l'aplicació teclejam:

java -jar FireAlarm.jar

A la finestra inicial podem veure que no hi ha cap sensor registrat i no mostra cap lectura.
A partir d'ara, en l'instant que s'activa un sensor, l'aplicació el registra i apareixen les dades de les lectures i els paràmetres actuals de configuració del node. Simultàniament, en el terminal des del qual hem iniciat l'execució de la interfície gràfica s'aniran mostrant tots els missatges que s'envien en un i altre sentit.

Cada vegada que la consola rep un missatge actualitza les dades amb els valors nous que li acaben d'arribar mostrant en color verd els valors actualitzats si estan dins els marges correctes i en vermell si són valors d'alarma. En cas de que el valor no s'hagi actualitzat es mostra en color gris. Això succeeix amb la dada de temperatura i lluminositat quan només es reben missatges de nivell de bateria. La pantalla de monitorització de la consola de control es pot veure a la figura 38.

Figura 38: Pantalla de monitorització de la consola d'usuari

Al quadre de text de la part esquerra es pot veure l'estat del sistema:

Missatge	**Significat**
Sensor inactiu	Sensor registrat a la consola de control, però en stand-by
Sensor actiu	Sensor registrat a la consola de control que transmet dades
Alarma manual HH:MM:SS	Alarma manual activada a l'hora HH:MM:SS
Alarma automàtica HH:MM:SS	Alarma automàtica activada a l'hora HH:MM:SS
Bateria baixa	Tensió de la bateria per sota del 90% del valor nominal
Sensor perdut	Perduda la comunicació amb el sensor

A la dreta del quadre de text hi podem veure els valors de la configuració actual dels paràmetres, i a sota el botó per confirmar l'alarma per part de l'operador.
Si el que volem fer és modificar qualque paràmetre, podem fer-ho des de la pestanya de configuració.

Figura 39: Pantalla de configuració de la consola de control

Aquesta segona pestanya, tal com es veu a la figura 39, mostra els valors actuals i permet modificar-los respectant els marges següents:

Paràmetre	**Unitats**	**Rang de valors**
Temperatura llindar d'alarma	ºC	30 - 100
Període lectura temperatura	ms	1000 - 7000
Període enviament bateria	ms	Període lectura temperatura - 120000
Període enviament dades	ms	Període lectura temperatura - 300000 o bé 0

Com podem veure a la taula anterior, el període de lectura de la temperatura, el període d'enviament de dades i el període d'enviament del nivell de bateria no necessàriament han de coincidir. Cada paràmetre té la seva funcionalitat:

- El “període d'enviament bateria” ens indica cada quan es refresca la informació de l'estat de la bateria en el panell gràfic de la consola de control.
- El “període d'enviament dades” ens indica cada quan es refresca la informació de temperatura i lluminositat en el panell gràfic de la consola de control.
- El període de lectura de la temperatura ens indica cada quan es pren una mostra de la temperatura del sensor, a més d'una mesura del nivell de bateria i de lluminositat. Cada cop que es fa la lectura es comprova si cap dada sobrepassa el llindar establert. Si es produeix una situació d'alarma (bateria baixa, temperatura massa alta), s'envia immediatament un missatge indicant-ho a la consola de control. En cas contrari, no es fa res.

Aquest últim camp pot prendre un valor màxim de 7 segons, ja que el sistema de protecció de caigudes (WatchDog) està configurat a 8 segons. És a dir, si en 8 segons la mota no ha realitzat cap lectura es reinicia automàticament per evitar situacions de caiguda del dispositiu. Un cop reiniciada, demana els paràmetres que tenia configurats abans de reiniciar-se a la consola de control.

Així com les dades de temperatura s'usen per determinar si hi ha o no un incendi, les dades d'estat de la bateria tenen una doble funcionalitat. Per una part, aquest senyal ens permet dur a terme una monitorització del nivell de tensió de les bateries, i per altra ens serveix per determinar si s'ha perdut la comunicació amb el sensor. El programa determina que s'ha perdut la comunicació amb un sensor si està més de dos períodes (definits al camp “període d'enviament bateria”) sense rebre informació de l'estat de la bateria.

11.2. Execució i compilació

El primer que ens cal és disposar del PC amb Linux i amb el TinyOS instal·lat per poder compilar els programes de les motes. Un cop tenim configurades les variables d'entorn ja podem compilar els programes de les motes. En aquest cas Tenim l'aplicació ***BaseStation*** per a la mota que fa d'Estació Base i ***FireSensor*** per a la mota Sensor. Per a cada un s'adjunta una carpeta amb el codi font i tots els arxius necessaris.

Un cop copiades en el PC obrirem un terminal en la que vulguem compilar i teclejarem "*make cou24 install,1*" pel cas de la ***BaseStation*** i "*make cou24 install,2*" pel cas de ***FireSensor***, que compilarà les aplicacions i les deixarà en les corresponents subcarpetes *./buid/cou24*.
El que ens cal fer després és carregar els corresponents programes a les motes i per fer-ho cal utilitzar el programa **meshprog**. Primer connectam la mota que volem programar al port USB del PC i seguidament, del mateix terminal que teníem obert en la carpeta de l'aplicació que volem instal·lar teclejam:

```
meshprog -t/dev/ttyUSB0 -f./build/cou24/main.srec.out-1
```

pel cas de ***BaseStation*** i

```
meshprog -t/dev/ttyUSB0 -f./build/cou24/main.srec.out-2
```

pel cas de ***FireSensor*** i acte seguit polsam el botó *reset* de la mota.

Quan la carrega finalitza ens ho indica en el terminal i ja podem desconnectar la mota i connectar la següent.
Quan ja estan programats tots els nodes ja es pot iniciar l'aplicació de la consola seguint les instruccions de l'apartat anterior.

Ja només ens cal posar les bateries al node Sensor i activar-lo passant l'imant dos cops per sobre del sensor d'efecte Hall i al cap d'uns instants s'aniran mostrant els missatges.