Sistema detector d'incendis WSN (Wireless Sensor Network)

Estudiant: Domínguez López, Emiliano
Enginyeria Técnica en Telecomunicació especialitat Telemàtica

Consultor: Jordi Bécares Ferrés

Data de lliurament 12 de Juny de 2012

Agraïments:

L'últim tram de la carrera ha estat una tasca molt feixuga, amb moltíssim esforços, sense descansar gaire en molts moments, per això, amb aquest treball vull agrair a totes les persones que m'han fet costat mentre jo anava cursant una assignatura darrera d'una altra, donant-me la sensació, de vegades, que això no acabaria mai.

En primer lloc vull agrair molt especialment el suport i l'empenta rebuda per a començar aquesta carrera a la meva dona **Esther Cebrián**, i sobretot per les estones que ha passat amb mi sense poder sortir de casa per tal d' acabar la feina a temps i per tenir sempre el seu recolzament anímic.

Agrair als meus pares i al meu germà, la paciència que han tingut quan han estat llargues temporades sense veure'm a causa de la càrrega de feina que ha suposat la carrera.

Als meus sogres i cunyats pels suports i ànims que m'han donat durant tots els anys de carrera.

Als companys de feina (Ivan, David, JC,...) que a l'últim tram de la carrera m'han donat molts ànims i m'han ajudat molt per a poder acabar aquest projecte.

I com no, agrair al consultor Jordi, per la paciència que ha demostrat i el suport rebut durant tot el projecte.

En resum, a tot el món que m'hagi donat ànims per a portar a terme aquesta carrera.

Moltes gràcies a TOTS!

RESUM

Amb aquest projecte, que pertany a l'àrea dels sistemes encastats (embedded systems), s'ha desenvolupat una xarxa de sensors sense fils (Wireless Sensor Networks) per a crear un **sistema detector d'incendis.**

S'han fet servir dos dispositius composats d'un microcontrolador, un sistema per a comunicar-se via ràdio i via port sèrie (USB), i diversos sensors que han proporcionat els perifèrics necessaris per a realitzar aquest sistema de forma satisfactòria.

Els dos dispositius s'han programat amb un sistema operatiu (tinyOS) creat per a aquest tipus de dispositius, i que és molt adient ja que és capaç d'adaptar el codi generat per a executar-lo en sistemes amb pocs recursos hardware.

La xarxa creada amb els dos nodes utilitza el protocol 802.15.4 per a comunicar-se. Aquest protocol pertany a xarxes del tipus WPAN (Wireless Personal Area Network), similar al protocol que es fa servir per a Bluetooth.

Els dos dispositius es controlen a través d'una aplicació central instal·lada en un PC amb sistema operatiu Linux, i proporciona totes les funcionalitats per a comunicar-se amb els dos dispositius.

Índex de continguts

Índex de figures

1. Introducció.

Un sistema de seguretat que no necessiti gaire instal·lació elèctrica o mecànica és molt interessant per a controlar aspectes de seguretat en edificis antics o monuments amb algun interès arquitectònic o cultural.

Per poder preservar aquests edificis o monuments i adaptar-los a la vigent normativa de seguretat, es fa molt recomanable pensar en sistemes de detecció d'intrusions o d'incendis que no necessitin realitzar gaires canvis estructurals com ara la instal·lació de cablejat, punts d'accés a alimentació, etc.

En un edifici que es vol preservar de la forma més original possible, es a dir, sense afegir-hi aspectes que no tenen res a veure amb l'arquitectura de l'edifici o monument, pot ser de gran vàlua disposar d'un sistema de detecció d'incendis amb el qual es pugui controlar la seguretat de l'edifici de forma pràcticament inapreciable.

1.1. Justificació.

Encara que al mercat existeixen productes similars al que s'ha desenvolupat, la realització d'aquest projecte aprofundirà molts conceptes treballats al llarg de la carrera i a més ens permetrà realitzar un projecte totalment pràctic. Interactuarem directament implementant un firmware i executant-lo sobre maquinari real,o sigui, no serà de forma simulada, sinó de manera pràctica.

El projecte finalitzat, podrà donar un servei òptim en llocs on sigui difícil realitzar qualsevol tipus de instal·lació, ja sigui perquè es tracti d'un monument històric i estigui prohibit alterar cap aspecte físic, o bé perquè l'accés sigui molt complicat i no es pugui dur a terme una instal·lació amb cable.

La tecnologia sense fils(WSN) emprada en aquest projecte té com a principal avantatge la mobilitat dels seus sensors i el seu baix consum.
Per tant, el fet de no disposar de connexió física per a comunicar-se amb un sensor remot, ens permet realitzar la instal·lació d'aquest sensor en el lloc que es vulgui i en el moment que es vulgui, es a dir, aquest sistema no té perquè ser un sistema estàtic, amb

una única ubicació, sinó que els nodes poden ser moguts en el moment desitjat per a controlar altres llocs.
Aprofitant aquesta mobilitat que ens proporciona la tecnologia WSN, el ventall de possibilitats per a aquest projecte és molt gran, ja que es podria pensar en que el sensor remot fos mòbil dins d'una mateixa sala, per exemple, així amb aquesta mobilitat, el sensor remot podria estudiar els diferents estats de la sala que s'estigués controlant i en posicions diferents en cada instant de temps, llavors controlant la posició dins la sala i la temperatura mesurada, es podria disposar d'un mapa termogràfic d'aquesta sala.

La tecnologia WSN s'ha fet servir en molts àmbits diferents com per exemple en domòtica, agricultura, salut, medi ambient, etc.

1.2. Descripció del projecte.

Aquest projecte dóna una solució pràctica ,econòmica i de baix consum, d'un sistema de detecció d'incendis mitjançant uns nodes electrònics controlats per un microcontrolador, que poden estar alimentats amb bateries de 1'5 volts o a través d'un port USB d'ordinador.

Aquests nodes es poden comunicar entre ells "sense fils" i proporcionen una sèrie de sensors i perifèrics que donen informació de l'espai on es troben ubicats (nivell de temperatura, nivell de llum, etc.).

Per realitzar aquesta aplicació, inicialment s'han fet servir dos nodes inal·làmbrics, per a controlar la detecció d'incendis en dues sales diferents, però es podrien fer servir més nodes, i controlar més sales, en un futur.

S'ha realitzat una aplicació a mida per a PC, on es poden controlar diverses informacions que poden ser útils: la temperatura de les sales llegides periòdicament amb un interval de temps establert per l'usuari, control del nivell de la bateria del node que estigui alimentat per aquestes bateries, si hi ha algun tipus d'alarma manual (polsador tipus SOS d'alguna sala, polsat per un usuari), avis de l'augment de temperatura establert a l'aplicació, etc. Aquesta aplicació emmagatzema els històrics, des del moment que s'inicia l'aplicació fins que es deixa d'utilitzar, amb els registres de totes les accions que rebi tant per part de l'usuari com dels nodes.

S'han realitzat 2 firmwares per als nodes, un què correspon al node que anirà connectat a l'ordinador i farà de node central, i un altre firmware per al node que es comunicarà amb el node central via ràdio, que serà el node remot.
El firmware de cada node realitza diferents accions: comunicació entre ells i l'aplicació central, emissió de dades a cada interval de temps establert per l'usuari, visualització dels diferents estats que pugui tenir el node mitjançant els seus díodes led, i opcions per a realitzar les instal·lacions d'una forma més còmode.

Per a la realització del firmware dels nodes s'ha aprofitat un sistema operatiu dissenyat específicament per a sensors d'aquest tipus, TinyOS.
I per a la realització de l'aplicació per al PC s'han aprofitat les llibreries Qt que es programen amb el llenguatge orientat a objectes C++.
L'escenari en funcionament quedaria de la següent manera:

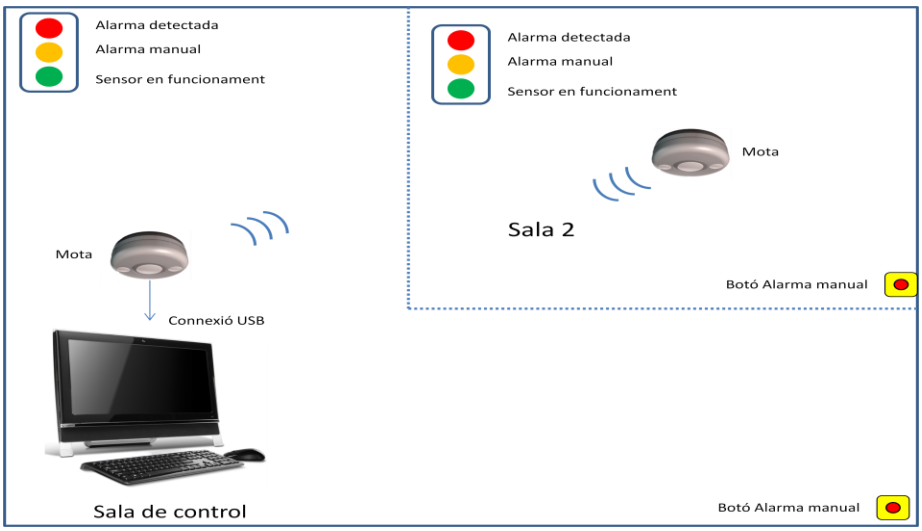


Figura 1. Escenari d'aplicació

1.3. Objectius del TFC.

L'objectiu principal d'aquest projecte és **poder detectar incendis en dues sales mitjançant una xarxa de sensors sense fils** i crear una aplicació per a instal·lar en un PC de la sala de control. Els possibles incendis es podran detectar gràcies a la visualització de l'augment de la temperatura per sobre d'un cert llindar. Aquest llindar l'establirà l'usuari a l'aplicació central per a qualsevol de les dues sales.

Per a complir aquest objectiu principal es creen una sèrie d'objectius secundaris (descrits a continuació) que conjuntament donen com a resultat l'objectiu principal:

- Creació d'una xarxa de sensors sense fils (WSN)
- Monitorar el nivell de temperatura en les sales que es volen controlar
- Monitorar l'estat del nivell de bateria dels sensors
- Activació / desactivació dels nodes mitjançant un sensor magnètic
- Visualització dels senyals (Leds dels nodes)
- Comunicació entre els nodes i l'aplicació central
- Proporcionar una interfície gràfica senzilla per a l'usuari
- Enregistrament dels successos
- Eines de suport per a la instal·lació dels sensors
- Sistema anticaigudes (WatchDog)

1.3.1. Creació d'una xarxa de sensors sense fils.

S'instal·larà a cada sala un node COU24, un d'ells estarà connectat a un PC ubicat a la sala de control i farà les funcions de node central, mentre que l'altre node estarà ubicat en altra sala i es comunicarà amb el node central (primer node) via ràdio.

1.3.2. Monitorar el nivell de temperatura en les sales a controlar.

Cada node disposa d'un termistor que proporciona una tensió, en volts, proporcional a la temperatura que detecta. Aquesta tensió la mesura el microcontrolador mitjançant un convertidor Analògic/Digital.

Per als intervals de temps definits per l'usuari a la interfície gràfica de l'aplicació principal, els nodes, els quals estan contínuament mesurant la temperatura, enviaran per port USB o per ràdio, en funció del node que sigui, central o remot respectivament, el valor de temperatura de la sala on es trobin.

1.3.3. Monitorar l'estat del nivell de bateria dels sensors.

El node mesura contínuament el nivell de bateria i envia per port USB o per ràdio en funció del node que sigui, central o remot respectivament, el valor actual del nivell de bateria.

1.3.4. Activació/desactivació dels nodes mitjançant sensor magnètic.

El sensor d'efecte Hall, emet un senyal quan s'apropa un camp magnètic al seu costat.
Aquest sensor no està connectat a cap convertidor Analògic/Digital, només ens dona un estat On/Off, per aquest motiu el sensor està connectat a una interrupció del microcontrolador.
Quan el node s'encén, es detectaran les vegades que s'ha activat el sensor d'efecte Hall per a saber si l'usuari vol activar o desactivar el node.

1.3.5. Visualització dels senyals.

Els diferents estats en que es trobi el node (activació/desactivació, detecció d'alarmes de nivell de bateria, detecció de temperatura elevada, o alarma manual d'usuari) es podran visualitzar mitjançant els 3 leds dels que disposa cada node, cadascun d'ells indica un estat diferent en funció de les seves intensitats lumíniques amb un total de set estats possibles .

1.3.6. Comunicació entre els nodes i l'aplicació central.

El node central anirà connectat per port USB i es comunicarà amb l'aplicació central mitjançant el protocol de transmissió sèrie.
El node remot es comunicarà aplicant el protocol 802.15.4 (WPAN - Wireless Personal Area Network), amb el node central. Un cop el node central rebi dades del node remot, se les retransmetrà a l'aplicació central mitjançant la comunicació sèrie.

Els nodes enviaran diversos missatges amb les dades obtingudes com per exemple: petició de connexió/desconnexió, nivell de temperatura i bateria actual, possibles alarmes produïdes, etc. Aquests missatges es rebran a l'aplicació central, la qual farà amb cada missatge el que cregui oportú.

1.3.7. Proporcionar una interfície gràfica senzilla per a l'usuari.

Per a poder controlar les dues sales amb els corresponents nodes, es proporcionarà una aplicació gràfica i intuïtiva amb la que l'usuari podrà veure de manera molt ràpida i fàcil en quin estat actual es troben cadascuna de les dues sales i realitzar diferents configuracions.

1.3.8. Enregistrament periòdic dels successos.

Tan bon punt s'inicïi l'aplicació principal es crearà un fitxer amb l'històric dels esdeveniments succeïts fins al moment del tancament de l'aplicació. Cada esdeveniment quedarà enregistrat amb la data i l'hora en que ha ocorregut.

1.3.9. Eines de suport per a la instal·lació dels sensors.

L'aplicació central i el firmware de cada node proporcionen eines de suport per a poder realitzar la instal·lació del sistema de forma correcta, es a dir, per a poder realitzar una configuració personalitzada segons les necessitats. Aquestes eines són: les proves de cobertura entre tots dos nodes, i la visualització del nivell de bateria i de la temperatura actual (sense necessitat de l'activació manual dels nodes) per a poder configurar la temperatura d'alarma.

1.3.10 Sistema anticaigudes (WatchDog)

El firmware dels nodes ha de proporcionar un sistema que eviti els possibles estats incorrectes del funcionament d'algun dels nodes.

Si qualsevol dels dos nodes detecta un mal funcionament, es reinicia automàticament i l'aplicació central li proporcionarà les dades de configuració que tenia fins el moment del reinici.

1.4. Enfocament i mètode seguit.

Un cop es van rebre els requisits que s'havien de complir (com si d'un client es tractés), es va estudiar la millor manera d'assolir tots els requisits demanats per tal de fer una primera proposta de projecte.

En una primera fase, es van estudiar les possibilitats que proporcionaven els nodes i la seva programació per a poder definir unes primeres pautes del funcionament del sistema i un aprofitament òptim dels perifèrics que proporcionen els nodes.
D'aquest primer estudi es va generar el document "Proposta de TFC", en el qual s'especificava el funcionament de tot el sistema que es pretenia desenvolupar.

A continuació, es va plantejar de quina manera es realitzaria tot el procediment d'implementació de codi per assolir les dues dates de lliurament més importants, llavors es va realitzar un primer cronograma en el qual es van dividir totes les tasques especificades en el document "Proposta de TFC" en dues fites.

La primera fita pretenia aconseguir el desenvolupament de tot el treball previst però a grans trets, es a dir, dins la primera fita es van assolir les següents tasques:

- Les comunicacions entre el node central i l'aplicació central.
- Les comunicacions entre el node remot i el node central.
- Lectura en tots dos nodes dels sensors de temperatura i de nivell de bateria.
- Visualització dels diferents estats mitjançant els leds de cada node.
- Interfície gràfica senzilla per a poder controlar els intervals de temps i d'alarmes.
- Petit conjunt de proves de funcionament.

Dins la segona fita, es pretenia solucionar els problemes que poguessin haver sorgit del primer desenvolupament i realitzar les tasques pendents de la proposta, què eren:

- Implementació d'un sistema anticaigudes (WatchDog) als dos nodes.
- Aplicació per a portar a terme la configuració dels nodes (prova de cobertura).
- Finalització de la interfície gràfica de l'aplicació central.
- Conjunt de proves definitiu de funcionament.

Amb l'ajuda del pla de treball realitzat, on es diferenciava cadascuna de les tasques per senzilla que fos, i amb la proposta, es van anar desenvolupant els mòduls necessaris per al funcionament de tot el sistema, sempre intentant seguir amb cura el pla de treball.

Per a la realització de la memòria, s'han anat prenent notes durant tot el desenvolupament del codi, d'aquesta manera al finalitzar la implementació del codi s'ha pogut redactar la memòria amb l'ajuda de tots els apunts recopilats fins al moment.

1.5. Planificació del projecte.

Per a desenvolupar tots els requisits demanats en aquest projecte, es disposava de 3 mesos aproximadament.

Un cop feta la proposta que es pretenia desenvolupar, es va estudiar com funcionava l'entorn de programació que es faria servir (TinyOS-2.1.1), el llenguatge que s'havia d'utilitzar (nesC), i les possibilitats que proporcionaven tant els nodes com l'entorn de programació.

Coneixent quines possibilitats proporciona tant l'entorn i el llenguatge com els dispositius, es va realitzar una identificació més acurada de les tasques que s'havien de portar a terme, l'ordre en que aniria cadascuna, i els objectius clarament identificats del que havia de fer cada tasca.

A continuació es mostra el primer cronograma que es va dur a terme amb totes les tasques definides anteriorment.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Desenvolupament 1ª Fase**	**26 días**	**mar 20/03/12**	**mar 24/04/12**
2	**Activació Leds**	**2 días**	**mar 20/03/12**	**mié 21/03/12**
3	Led verd ON continu	1 día	mar 20/03/12	mar 20/03/12
4	Led vermell ON continu	1 día	mar 20/03/12	mar 20/03/12
5	Led taronja ON continu	1 día	mar 20/03/12	mar 20/03/12
6	Led verd intermitent (0.5s)	1 día	mar 20/03/12	mar 20/03/12
7	Led taronja intermitent (0.5s)	1 día	mar 20/03/12	mar 20/03/12
8	Led vermell intermitent (0.5s)	1 día	mar 20/03/12	mar 20/03/12
9	Led vermell intermitent (0.25s)	1 día	mié 21/03/12	mié 21/03/12
10	3 Leds intermitents (0.5s)	1 día	mié 21/03/12	mié 21/03/12
11	3 Leds seqüencials	1 día	mié 21/03/12	mié 21/03/12
12	**Adquisició polsador usuari**	**1 día**	**jue 22/03/12**	**jue 22/03/12**
13	Estat polsador	1 día	jue 22/03/12	jue 22/03/12
14	**Adquisició sensors**	**2 días**	**vie 23/03/12**	**dom 25/03/12**
15	Lectura del sensor de temperatura	2 días	vie 23/03/12	dom 25/03/12
16	Lectura del sensor d'efecte Hall	2 días	vie 23/03/12	dom 25/03/12
17	Lectura del nivell de bateries	2 días	vie 23/03/12	dom 25/03/12
18	**Enviament dades via radio**	**3 días**	**lun 26/03/12**	**mié 28/03/12**
19	Enviament dades via radio	3 días	lun 26/03/12	mié 28/03/12
20	**Recepció dades via radio (Node remot)**	**3 días**	**jue 29/03/12**	**dom 01/04/12**
21	Recepció dades via radio	3 días	jue 29/03/12	dom 01/04/12
22	**Programa Principal (Node remot)**	**5 días**	**lun 02/04/12**	**vie 06/04/12**
23	Programa Principal	5 días	lun 02/04/12	vie 06/04/12
24	**Enviament dades via USB(Node central)**	**3 días**	**sáb 07/04/12**	**mar 10/04/12**
25	Comunicació USB (Aplicació-Node central)	3 días	sáb 07/04/12	mar 10/04/12
26	**Recepció dades via USB (Node central)**	**3 días**	**mié 11/04/12**	**vie 13/04/12**
27	Recepció dades via USB(Node central)	3 días	mié 11/04/12	vie 13/04/12
28	**Programa Principal (Node central)**	**5 días**	**sáb 14/04/12**	**jue 19/04/12**
29	Programa principal (node central)	5 días	sáb 14/04/12	jue 19/04/12
30	**Aplicació central**	**10 días**	**mié 11/04/12**	**mar 24/04/12**
31	Aplicació central	10 días	mié 11/04/12	mar 24/04/12
32	**Desenvolupament 2ª Fase**	**21 días**	**mar 24/04/12**	**mar 22/05/12**
33	**Revisió tasques fase anterior**	**5 días**	**mar 24/04/12**	**dom 29/04/12**
34	Revisió tasques fase anterior	5 días	mar 24/04/12	dom 29/04/12
35	**WatchDog**	**4 días**	**lun 30/04/12**	**jue 03/05/12**
36	WatchDog	4 días	lun 30/04/12	jue 03/05/12
37	**Configuració cobertura dels nodes**	**2 días**	**vie 04/05/12**	**lun 07/05/12**
38	Configuració cobertura dels nodes	2 días	vie 04/05/12	lun 07/05/12
39	**Interficie gràfica per l'aplicació central**	**11 días**	**mar 08/05/12**	**mar 22/05/12**
40	Interficie gràfica per l'aplicació central	11 días	mar 08/05/12	mar 22/05/12

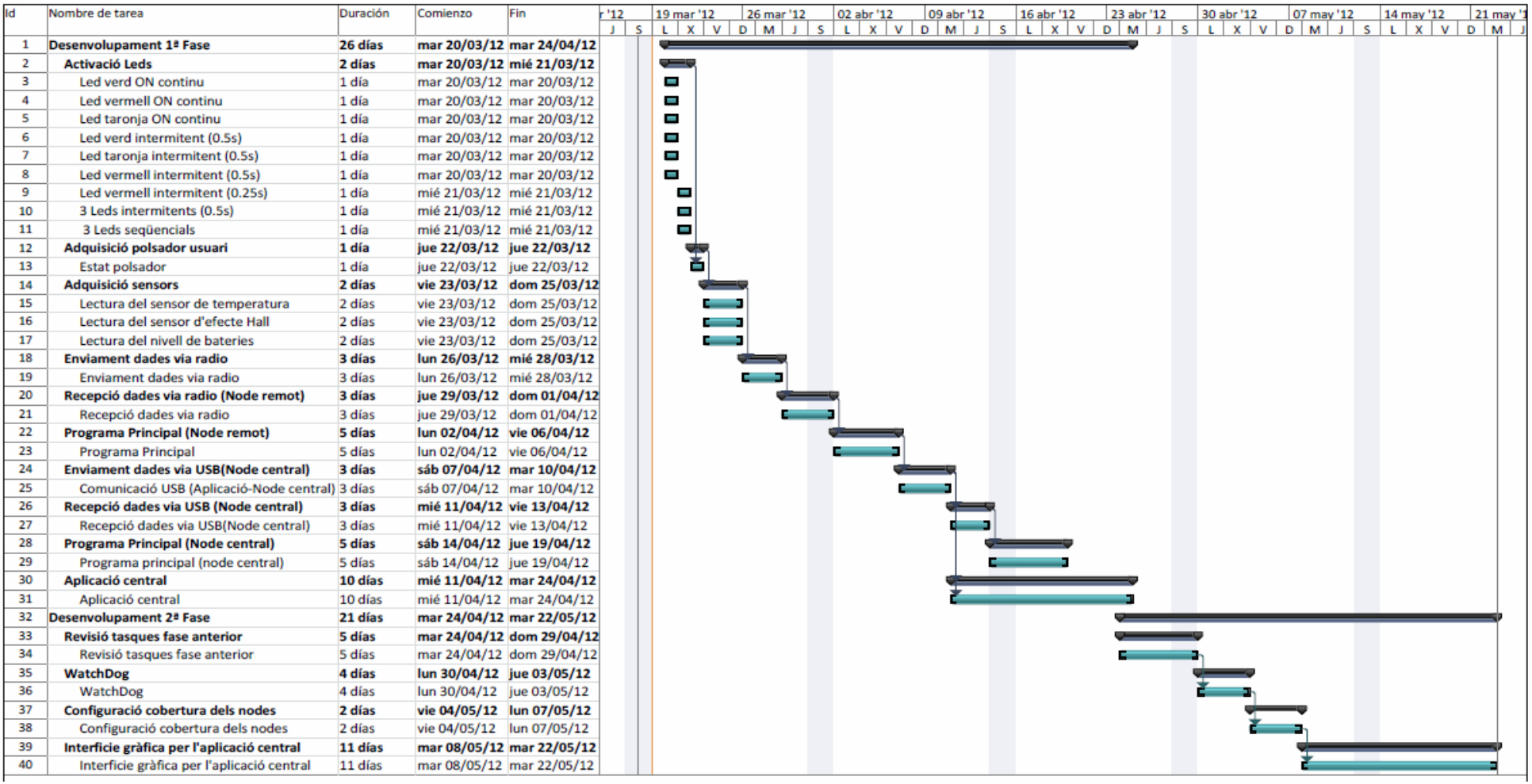


Figura 2. Cronograma inicial

Després del desenvolupament de la primera fase de codi, i amb el suggeriment del consultor del projecte, es van haver de modificar els temps previstos en el cronograma, a causa del canvi substancial del codi en el node central, que no feia ús d'un mòdul molt interessant per a la implementació de la comunicació sèrie/ràdio.

L'adaptació d'aquest mòdul al codi del node central ja implementat (va variar el 80% del codi del node central) va causar un cert endarreriment en la segona fase de desenvolupament.

Al cronograma final, referent al desenvolupament de tot el projecte, s'han afegit com a tasques finals, la creació d'aquesta memòria, i la presentació del projecte.

S'ha d'indicar, que encara que no s'ha començat la memòria en les dates indicades pel pla docent de l'assignatura, s'han anat prenen notes de forma continuada durant la implementació de tot el projecte, d'aquesta manera, s'ha disposat d'un petit esborrany de la memòria final.

A continuació es mostra el cronograma final que s'ha realitzat amb les dates reals:

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Desenvolupament 1ª Fase**	**26 días**	**mar 20/03/12**	**mar 24/04/12**
2	**Activació Leds**	**2 días**	**mar 20/03/12**	**mié 21/03/12**
3	Led verd ON continu	1 día	mar 20/03/12	mar 20/03/12
4	Led vermell ON continu	1 día	mar 20/03/12	mar 20/03/12
5	Led taronja ON continu	1 día	mar 20/03/12	mar 20/03/12
6	Led verd intermitent (0.5s)	1 día	mar 20/03/12	mar 20/03/12
7	Led taronja intermitent (0.5s)	1 día	mar 20/03/12	mar 20/03/12
8	Led vermell intermitent (0.5s)	1 día	mar 20/03/12	mar 20/03/12
9	Led vermell intermitent (0.25s)	1 día	mié 21/03/12	mié 21/03/12
10	3 Leds intermitents (0.5s)	1 día	mié 21/03/12	mié 21/03/12
11	3 Leds seqüencials	1 día	mié 21/03/12	mié 21/03/12
12	**Adquisició polsador usuari**	**1 día**	**jue 22/03/12**	**jue 22/03/12**
13	Estat polsador	1 día	jue 22/03/12	jue 22/03/12
14	**Adquisició sensors**	**2 días**	**vie 23/03/12**	**dom 25/03/12**
15	Lectura del sensor de temperatura	2 días	vie 23/03/12	dom 25/03/12
16	Lectura del sensor d'efecte Hall	2 días	vie 23/03/12	dom 25/03/12
17	Lectura del nivell de bateries	2 días	vie 23/03/12	dom 25/03/12
18	**Enviament dades via radio**	**3 días**	**lun 26/03/12**	**mié 28/03/12**
19	Enviament dades via radio	3 días	lun 26/03/12	mié 28/03/12
20	**Recepció dades via radio (Node remot)**	**3 días**	**jue 29/03/12**	**dom 01/04/12**
21	Recepció dades via radio	3 días	jue 29/03/12	dom 01/04/12
22	**Programa Principal (Node remot)**	**5 días**	**lun 02/04/12**	**vie 06/04/12**
23	Programa Principal	5 días	lun 02/04/12	vie 06/04/12
24	**Enviament dades via USB(Node central)**	**3 días**	**sáb 07/04/12**	**mar 10/04/12**
25	Comunicació USB (Aplicació-Node centra	3 días	sáb 07/04/12	mar 10/04/12
26	**Recepció dades via USB (Node central)**	**3 días**	**mié 11/04/12**	**vie 13/04/12**
27	Recepció dades via USB(Node central)	3 días	mié 11/04/12	vie 13/04/12
28	**Programa Principal (Node central)**	**5 días**	**sáb 14/04/12**	**jue 19/04/12**
29	Programa principal (node central)	5 días	sáb 14/04/12	jue 19/04/12
30	**Aplicació central**	**33 días**	**mié 11/04/12**	**vie 25/05/12**
31	Interficie gràfica per l'aplicació central	33 días	mié 11/04/12	vie 25/05/12
32	Aplicació central	33 días	mié 11/04/12	vie 25/05/12
33	**Desenvolupament 2ª Fase**	**23 días**	**jue 26/04/12**	**sáb 26/05/12**
34	**Revisió tasques fase anterior**	**16 días**	**jue 26/04/12**	**jue 17/05/12**
35	Revisió tasques fase anterior	13 días	jue 26/04/12	lun 14/05/12
36	**Configuració cobertura dels nodes**	**5 días**	**vie 18/05/12**	**jue 24/05/12**
37	**WatchDog**	**1 día**	**vie 25/05/12**	**vie 25/05/12**
38	WatchDog	1 día	vie 25/05/12	vie 25/05/12
39	**Creació documentació final projecte**	**14 días**	**sáb 26/05/12**	**jue 14/06/12**
40	Creació primera versió memoria del project	7 días	sáb 26/05/12	sáb 02/06/12
41	Modificació i lliurament memoria del projec	8 días	dom 03/06/12	mar 12/06/12
42	Creació presentació del projecte	10 días	dom 03/06/12	jue 14/06/12
43				
44				

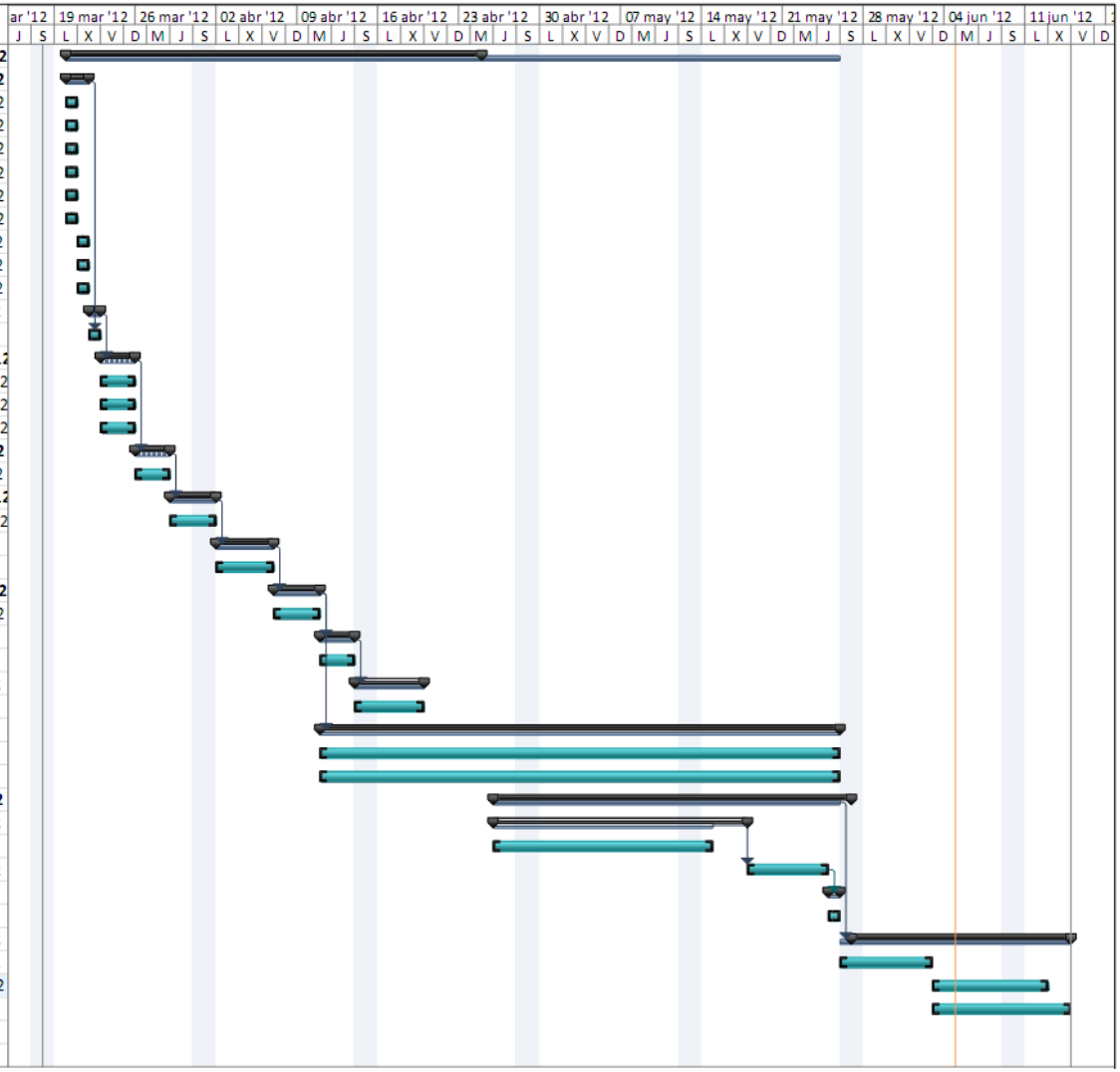


Figura 3. Cronograma final

1.6. Recursos emprats.

Per portar a terme aquest projecte, s'han fet servir els següents elements:

Hardware:

- 2 motes COU24

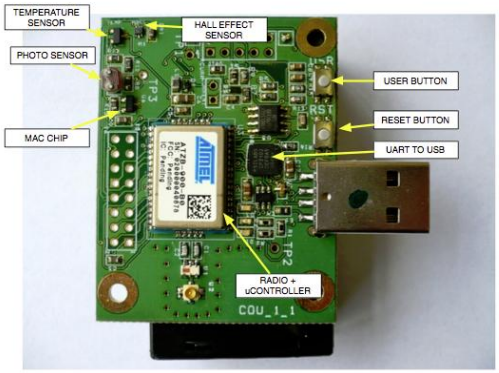


Figura 4. Mota COU24

- PC

Figura 5. Ordinador PC amb S.O. Linux

Software:

- Sistema operatiu Linux ubuntu 10.04

Sistema operatiu open source, basat en el sistema operatiu UNIX i el llenguatge de programació C. El primer Kernel de Linux el va implementar Linus Torvalds, que encara avui continua fent aportacions a les diferents versions de Kernel.
És el sistema operatiu que es recomana per a instal·lar i fer servir TinyOS. Per altra banda, la recomanació per a una bona comunicació sèrie amb els nodes i el sistema operatiu és que aquest estigui correctament instal·lat a l'ordinador i no s'executi sobre una màquina virtual.
Amb l'experiència d'aquest projecte he comprovat que la comunicació sèrie no és gaire eficient dins d'una màquina virtual, encara que ho sembli.

- Sistema operatiu TinyOS 2.1.1

Sistema operatiu orientat a esdeveniments, basat en el llenguatge de programació nesC i dissenyat per a nodes amb pocs recursos (RAM/ROM,...).
Una aplicació implementada amb llenguatge nesC es composa de mòduls i configuracions. En els mòduls es codifiquen totes les "eines" que s'usaran o es proveiran a la resta de mòduls, i la implementació corresponent al funcionament d'aquestes eines. En les configuracions es realitzen el que es denomina connexions, que indiquen quin mòdul és proveïdor d'algunes eines i quin mòdul en farà ús d'aquestes.

- IDE Eclipse Ganymede 3.4.2

És un entorn de desenvolupament integrat open source que ens servirà, conjuntament amb el plugin Yeti2, per a realitzar les implementacions de codi sota l'entorn de TinyOS i el llenguatge nesC.
Aquest IDE també proporciona diferents entorns per a programar en altres llenguatges de programació com son C, C++ i Java.

- Plugin Yeti2 per a Eclipse

Aquest plugin, desenvolupat per la universitat de Zurich, ens permetrà poder introduir codi d'una forma més simple, ja que ens proporciona una ajuda constant conforme es van teclejant instruccions o mòduls ja implementats.

- IDE Qt Creator

Entorn de desenvolupament que fa servir les llibreries Qt i que proporciona unes interfícies gràfiques molt enriquidores. El llenguatge de programació que es fa servir és el C++.

- Meshprog

Aquest programa ens permet carregar el firmware implementat amb TinyOS i nesC dins de la memòria de programa dels nodes. El seu funcionament requereix que el microcontrolador del node disposi d'un programa "Bootloader", que s'encarrega d'escoltar el port sèrie després d'un Reset.
Si en mig segon no rep el missatge indicant-li que carregui un nou programa, aleshores el microcontrolador executa el programa d'usuari. En canvi, si en el mig segon rep el missatge conforme el programa "meshprog" li envia un nou firmware per a carregar en memòria, llavors, es comença la càrrega del nou programa en la posició de memòria establerta per als programes d'usuari.
Amb aquest tipus d'utilitat i el Bootloader dins del microcontrolador, ens estalviem tenir hardware extern dedicat exclusivament a la programació dels microcontroladors.

1.7. Productes obtinguts.

Com a resultat d'aquest projecte s'han obtingut 3 productes:

- Firmware del node central: Aplicació que s'executarà en un dels 2 nodes dels que es disposa. Farà les funcions de node central, i serà el responsable de establir comunicació per USB(comunicació sèrie) amb l'aplicació central, per via ràdio amb el node remot, i realitzar les operacions de lectures dels sensors, activació/desactivació del node i visualització de les diferents alarmes activades.

- Firmware del node remot: Aplicació que s'executarà en el node *no central*. Farà les mateixes operacions que el node central, però només contempla la comunicació via ràdio amb el node central i no la comunicació sèrie.

- Aplicació central: Aplicació que s'executarà sota entorn Linux. Proporcionarà una interfície gràfica, mitjançant la qual es visualitzaran les diferents dades que poden proporcionar-nos els nodes (temperatura, nivell de bateria o possibles alarmes que siguin actives).
  A més de la comunicació amb els nodes, l'aplicació s'encarrega de guardar un històric amb el registre de les actuacions que es duguin a terme per part de l'usuari, així com dels possibles esdeveniments que indiquin els nodes, i per últim, permetrà una correcta instal·lació dels nodes en el seu lloc de funcionament.

1.8. Descripció dels altres capítols de la memòria

Un cop es té una idea del treball que s'ha realitzat, la temporització de les tasques, i els productes que s'han desenvolupat, als següents capítols de la memòria s'aprofundeix en altres aspectes que han contribuït al complet desenvolupament del projecte.

El capítol 2 ens dóna una perspectiva tecnològica de l'estat actual de la tecnologia que s'ha fet servir per al desenvolupament d'aquest projecte, així com de les alternatives que existeixen al mercat tant per als tipus de sensors que podem trobar actualment, com per a les eines de programació d'aquests dispositius.

Els capítols 3 i 4 donen una visió més tècnica del funcionament del sistema desenvolupat, així com del procediment que s'ha dut a terme per a desenvolupar aquestes aplicacions.

Als capítols 5 i 6 s'explica el punt de vista segons la viabilitat tècnica del sistema desenvolupat així com un pressupost on es podrà veure quant costaria desenvolupar i instal·lar un sistema com aquest.

I els últims capítols es dediquen a conclusions, glossari de paraules utilitzades en aquesta memòria, autoavaluació , bibliografia i annexes.

2. Antecedents.

2.1. Estat de l'art.

Per a la realització d'aquest projecte, s'han fet servir diferents tecnologies, en quant a Hardware i Software es refereix.

En la part Hardware, s'han fet servir dos dispositius anomenats "motes" que es troben dins el grup denominat "dispositius Zigbit". Aquesta tecnologia permet crear xarxes sense fils tipus Bluetooth o Wifi, però de baix cost (podem trobar xips microcontroladors amb interfície ràdio al voltant dels 30€).

Les diferencies dels dispositius d'aquest tipus que podem trobar al mercat es basen en les característiques del microcontrolador que governa tot el sistema, i de la interfície ràdio que es fa servir .

Cada microcontrolador té diferents característiques com són el numero de ports d'entrada/sortida que proporciona, quantitat de memòria RAM i memòria ROM, velocitat d'execució, quantitat de convertidors A/D,etc.

Els dispositius d'aquest tipus tenen una estructura com la següent:

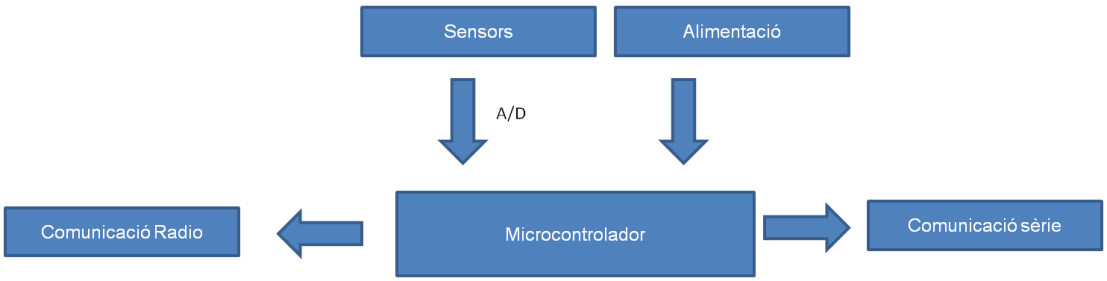


Figura 6. Estructura interna dels nodes

Buscant sistemes encastats del mateix estil que l'utilitza't en aquest projecte s'han trobat un parell de taules que mostren quines característiques presenten els dispositius:

Sensor Node	iCube	Isense	Micaz	BTnode
Developer	TUC Telecom Lab	Coalesenses	Crossbow	ETH
		MCU		
IC	Silabs C8051F321	Jennic JN5148	ATmega128L	ATmega128L
Speed (MHz)	1.5 - 24	4 - 32	up to 8	up to 8
Architecture	8-bit 8051	32-bit RISC	8-bit RISC	8-bit RISC
Flash/ROM (KB)	16	128	128 + 4(EEPROM)	128 + 4(EEPROM)
RAM (KB)	2.304	128	4	4
Consumption Active (mA)	0.41/MHz	0.28/MHz + 1.6	5 @ 4MHz	5 @ 4MHz
		Radio		
IC	CC2500	Jennic JN5148	CC2420	CC1000
Interface	SPI	(Single IC)	SPI	SPI
Max Data-rate (Kb/s)	500	250	250	76.8
Modulation	OOK, FSK, MSK	O-QPSK (802.15.4)	O-QPSK, MSK	FSK
Frequency Band (GHz)	2.4	2.4	2.4	0.868
RSSI/LQI	yes	yes	yes	yes
Consumption TX (mA)	10 @-22dBm, 21.2 @0dBm	15 @0.5dBm	8.5 @-25dBm, 17 @0dBm	8.6 @-20dBm, 16 @0dBm
Consumption RX (mA)	13.3@250Kb/s	17.5@250Kb/s	18.8@250Kb/s	11.8@76.8Kb/s
Consumption Idle (mA)	1.5	-	0.42	0.096
Max Transmit Power (dBm)	+1	+2.5	+0	+10
Max Sensitivity (dBm)	-104@2.4Kb/s	-95@250Kb/s	-95@250Kb/s	-110@0.6Kb/s
Sensitivity@250Kb/s (dBm)	-89	-95	-95	-96@76.8Kb/s (max rate)
		Node Capabilities		
ADC	14 channels	4 channels	8 channels	8 channels
Recommnended RTOS	Keil RTXtiny	custom	TinyOs	TinyOs
OTAP	yes (custom bootloader)	yes	yes	yes
RTOS-independent OTAP	yes (custom bootloader)	no	no	no
Batteries	2xAA	2xAA	2xAA	2xAA
		MCU+Radio Current		
CPU Sleep/Radio Off (μA)	0.1	0.1	15	15
CPU Idle/Radio Off (mA)	0.285	0.0012	4	4
Cpu@8MHz/Radio Off (mA)	3.28	3.84	10	10
Cpu@8MHz/Radio TX@250kbps (mA)	24.5 @0dBm	18.84 @0.5dBm	27 @0dBm	26 @0dBm
Cpu@8MHz/Radio RX@250kbps (mA)	16.58	21.34	28.8	21.8

Figura 7. Taula comparativa de diferents nodes(1)

Mote Type Year	*WeC* 1998	*René* 1999	*René 2* 2000	*Dot* 2000	*Mica* 2001	*Mica2Dot* 2002	*Mica 2* 2002	*Telos* 2004
Microcontroller								
Type	AT90LS8535		ATmega163		ATmega128			TI MSP430
Program memory (KB)	8		16		128			48
RAM (KB)	0.5		1		4			10
Active Power (mW)	15		15		15		60	0.5
Sleep Power (W)	45		45		75		75	2
Wakeup Time (s)	1000		36		180		180	6
Nonvolatile storage								
Chip	24LC256				AT45DB041B			ST M24M01S
Connection type	I^2C				SPI			I^2C
Size (KB)	32				512			128
Communication								
Radio	TR1000				TR1000	CC1000		CC2420
Data rate (kbps)	10				40	38.4		250
Modulation type	OOK				ASK	FSK		O-QPSK
Receive Power (mW)	9				12	29		38
Transmit Power at 0dBm (mW)	36				36	42		35
Power Consumption								
Minimum Operation (V)	2.7		2.7		2.7			1.8
Total Active Power (mW)	24				27	44	89	38.5
Programming and Sensor Interface								
Expansion	none	51-pin	51-pin	none	51-pin	19-pin	51-pin	10-pin
Communication	IEEE 1284 (programming) and RS232 (requires additional hardware)							USB
Integrated Sensors	no	no	no	yes	no	no	no	yes

Figura 8. Taula comparativa de diferents nodes(2)

Per a escollir un node en concret, vist l'ampli ventall del que es disposa al mercat, s'haurien de tenir en compte paràmetres com són el consum del dispositiu, la capacitat de memòria de programa (per si fa falta un codi molt extens en l'aplicació a desenvolupar), la forma de programar-lo, la velocitat d'execució del microcontrolador, l'abast de la interfície radio...

Pel que fa a les eines software que podem trobar per a programar aquests dispositius, trobem una sèrie de sistemes operatius que proporcionen un bon aprofitament del hardware dels dispositius, per a integrar el firmware dins de pocs recursos RAM/ROM.

Entre els diferents sistemes operatius dissenyats per a la programació d'aquest tipus de dispositius, podem trobar:

- **TinyOS**, que és el que s'ha fet servir en aquest projecte.
- **eCOS**, sistema operatiu open source de temps real, dissenyat per a sistemes encastats, que ofereix uns bons rendiments al hardware on es fa servir.
- **LiteOS**, dissenyat per la universitat de Illinois, és tracta d'un sistema operatiu en temps real, semblant al sistema Unix. L'ultima versió d'aquest sistema es va alliberar al juliol de 2011.
- **Contiki**, es tracta d'un sistema multitasca, que permet un bon aprofitament dels recursos dels dispositius.

Per últim, podem disposar de diferents protocols per a la implementació de xarxes de sensors WSN, com són:

- **802.15.4** : Aquest estàndard, del qual son responsables del seu desenvolupament el IEEE 802.15, defineix 3 possibles bandes de freqüències de treball que són 868 - 868'8 MHz, 902 - 928 MHz i 2400 - 2483'5 MHz.
  L'estàndard defineix la capa física i la capa d'accés al medi així com les topologies permeses per a les comunicacions, els mètodes de transport, etc.
- **Zigbee RF4CE**: Aquest protocol està basat en el protocol 802.15.4 i a la banda de freqüència 2'4GHz. Proporciona el mecanisme d'estalvi d'energia i suporta múltiples opcions per a la transmissió i molts perfils d'aplicació. Pot implementar AES-128 per a realitzar comunicacions segures.
- **MiWi**: És un protocol de comunicació creat per la marca de microcontroladors Microchip, de codi obert amb la limitació que s'han de fer servir els seus xips tant de processament com de interfície radio. És compatible amb dispositius que utilitzin el protocol 802.15.4 i permet establir velocitats de comunicació de 250Kbits/s.
- **WirelessHart**: Es tracta d'un protocol de comunicació digital sense fils per a ús industrial, que pot comunicar-se amb una aplicació central i petits instruments. Aquest protocol pot proporcionar accés a diagnòstics, configuració i processament de dades. Des de la versió 7, aquest protocol també incorpora la comunicació amb el protocol 802.15.4.

- **ISA100.11A**: Aquest protocol de comunicació va ser desenvolupat per la Societat Internacional de Automatització (ISA). Està basat en el protocol 802.15.4 i a la banda de 2'4GHz, i està caracteritzat com un protocol robust a les interferències

2.2. Estudi de mercat.

Al mercat es poden trobar sistemes detectors d'incendis semblants al desenvolupat en aquest projecte i amb un ampli ventall d'avantatges.

Per exemple de la marca Bosch, hi ha la central d'alarmes FPD-7024, i de la marca Morley-Ias (HoneyWell) hi han diferents centrals com la VSN-12-LT, que presenten com a avantatges respecte a un sistema com el desenvolupat en aquest projecte,

- Poden incrementar el seu numero de sensors sense cap modificació en el software.
- Poden incorporar diferents tipus de sensors, no solament de detecció d'incendis, sinó també detectors de fum, de presencia, de CO2 etc.
- I permeten la instal·lació de diferents perifèrics com son alarmes sonores, alarmes visuals, pantalles on veure les zones afectades,etc.

Com a contres ,el projecte ha estat un producte exclusivament desenvolupat sota els requeriments del "client", s'ha realitzat una adaptació més a mida, per a donar totalment el suport desitjat.

Seria una mica més costós permetre, amb els nodes dels que disposem, donar suport a diferents sensors, però això no significa que no es pugui.
El microcontrolador del que disposen els nodes utilitzats encara tenen moltes entrades/sortides sense utilitzar i queden molts convertidors analògics/digitals lliures, per aquest motiu, modificant aquest hardware, es podria donar un suport més ampli del que s'ha donat inicialment.

3. Descripció funcional.

Com ja s'ha descrit en el punt 1.7, s'han desenvolupat 3 productes, un firmware per a cada node del sistema, i una aplicació central que s'executarà en un PC sota l'entorn Linux.

Tot seguit es descriu el funcionament complet del sistema, el funcionament de cada node i el de l'aplicació central.

3.1. Sistema total.

El disseny del sistema total, s'ha plantejat com una xarxa de sensors en topologia estrella, es a dir, hi ha un node central que està connectat directament a l'ordinador el qual disposarà de l'aplicació central. La resta de nodes s'han de comunicar amb el node central i aquest retransmetrà les dades dels nodes remots a l'aplicació central.

Com que en el nostre sistema només disposem de dos nodes no és evident que la topologia utilitzada sigui *l'estrella*, per això es podria confondre amb un sistema Multi-Hop, en el qual es realitza un reenviament de missatges entre nodes propers per a que el missatge arribi a un node central; aquest sistema seria més apropiat per a cobrir distàncies grans.

La comunicació ràdio que es realitza entre els dos nodes, és una comunicació que segueix el protocol 802.15.4 (referit a xarxes personals sense fils - Wireless Personal Area Network).

A la figura 9 es veu clarament com es representa aquesta tipologia:

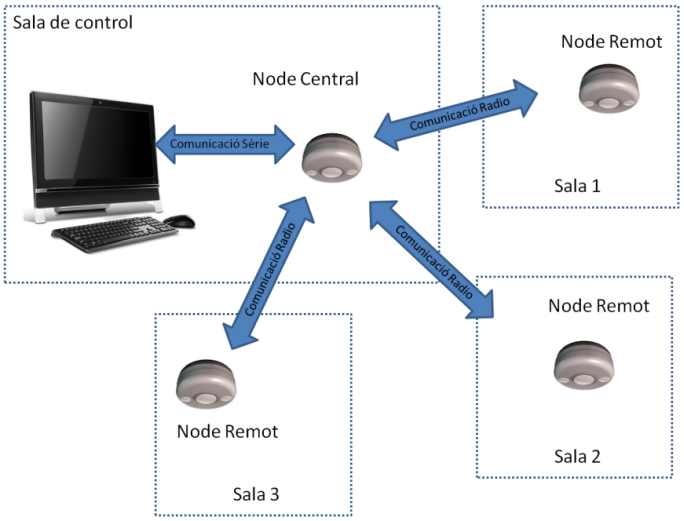


Figura 9. Topologia tipus estrella del sistema

El funcionament del sistema general, es pot descompondre en el diagrama de blocs de la figura 10, per tal d'entendre d'una forma clara com funciona el sistema:

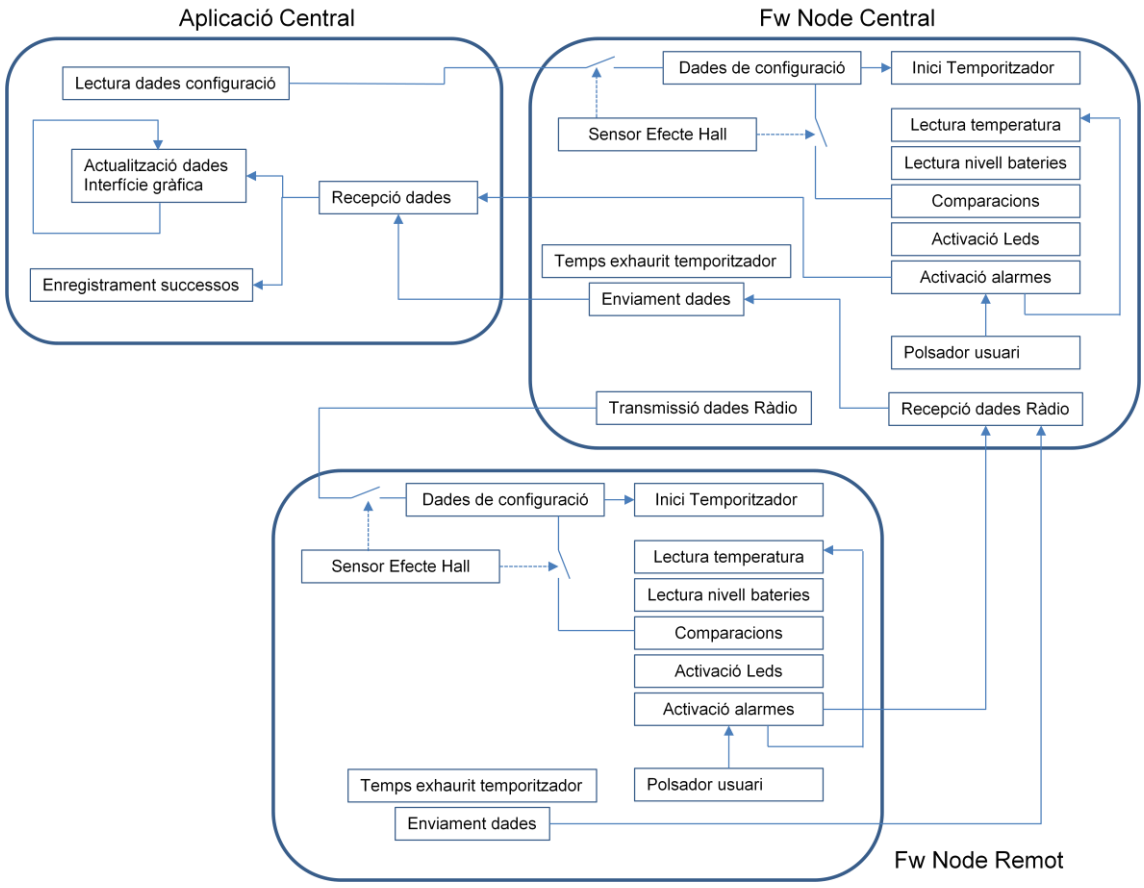


Figura 10. Diagrama de blocs del sistema

Per una banda, l'aplicació central enviarà les dades de configuració, que obté de la interfície gràfica, als nodes quan aquests siguin activats mitjançant el sensor d'efecte Hall.

De forma paral·lela, l'aplicació central està llegint contínuament les dades que hi puguin haver en el port sèrie (USB), amb les quals anirà actualitzant la interfície gràfica (mostrant temperatures, nivell de bateria, alarmes, registres de successos,...).
Les dades que rep l'aplicació central, són enviades pel node central mitjançant una comunicació sèrie a través del port USB. L'aplicació central només pot rebre dades enviades d'aquesta manera, ja que l'únic node que està connectat al PC de forma directa és el node central.
Les dades que hagi d'enviar el node remot han de passar pel node central, i aquest proporcionar-les a l'aplicació central.

El funcionament del node central i del node remot són molt semblants, ja que tots 2 realitzen les mateixes tasques, diferenciant-se entre ells en la forma de comunicar-se.

L'usuari configura els límits de temperatura i nivell de bateria a partir dels quals ha de rebre una alarma, i la periodicitat amb què vol rebre les dades. Els nodes, un cop han estat activats i han rebut la configuració, estan contínuament llegint els valors dels sensors de temperatura i del nivell de bateria, per a emetre, en cas necessari, l'alarma corresponent i visualitzar-la a través dels Leds que té cada node.

3.2. Software aplicació central PC.

L'aplicació central que s'ha dissenyat per al sistema presenta l'aspecte de la figura 11:

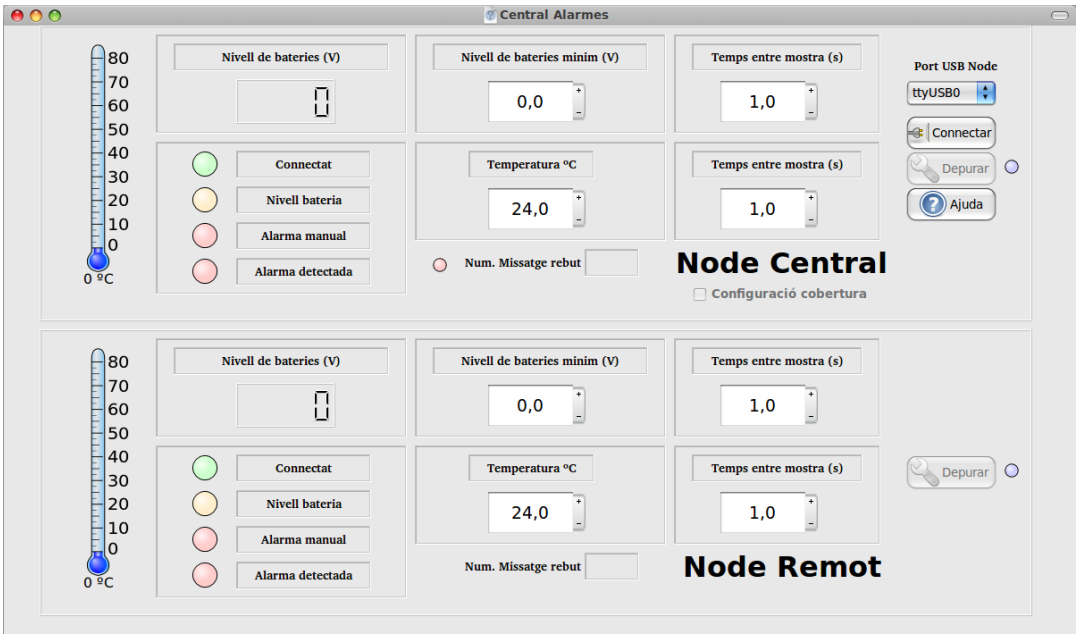


Figura 11. Aplicació central (PC)

A la interfície gràfica creada hi han tots els camps necessaris per a realitzar una configuració adient per a cada una de les sales que es vulguin controlar, també disposa d'una visualització senzilla per a veure els diferents valors que ens proporcionen els nodes central i remot.

L' aplicació central es nodrirà de missatges de diferent tipus rebuts pel port sèrie, els classificarà en funció de la tipologia i anirà actualitzant la interfície gràfica.

A la interfície gràfica disposem de les següents funcions:

- Visualització de les dades proporcionades per cada node.
- Elecció del port USB per on està connectat el node central.
- Visualització de la comunicació sèrie mitjançant un led que parpelleja cada cop que arriba un missatge, sigui del tipus que sigui.
- Visualització del número de seqüència de cada missatge per a poder observar la correcta comunicació sèrie.
- Comprovació de la cobertura del node remot.
- Ajuda per al funcionament de l'aplicació.

3.3. Firmware Node Central.

El firmware implementat al node central s'ha d'encarregar de:

- Llegir l'estat del sensor d'efecte Hall per a fer l'activació/desactivació del node.
- Enviar/Rebre missatges via port USB.
- Enviar/Rebre missatges via ràdio (protocol 802.15.4 - 2.4GHz).
- Realitzar el reconeixement (ACK) de cada missatge enviat via ràdio.
- Llegir l'estat del sensor de temperatura i el nivell de bateria del node, i comparar-los amb els llindars establerts per l'aplicació central.
- Fer les visualitzacions a traves dels leds dels diferents estats del node:
    - Led verd - Intermitència cada segon: El node pot ser activat.
    - Led verd - Encès de forma continua: El node està activat.
    - Led groc - Encès de forma continua: Activat polsador alarma manual.
    - Led groc - Intermitència cada 0'5 segons: Nivell de bateria per sota del llindar establert.
    - Led vermell - Intermitència 0'5 segons: S'ha superat el nivell de temperatura establert i encara no ha arribat el missatge corresponent a l'aplicació central.
    - Led vermell - Intermitència 0'25 segons: Ha arribat el missatge d'alarma de temperatura a l'aplicació central.
    - Led vermell - Encès de forma continua: Alarma acceptada des de l'aplicació central.
- Control antibloquejos (WatchDog): si el node es troba en una situació anòmala, es fa ell mateix un "reset".

El node genera diferents tipus de missatges (cadascun té una finalitat establerta) per a comunicar-se amb l'aplicació central.
Per a la realització d'aquest sistema s'han creat els següents missatges:

- Activar/Desactivar el node.
- Enviar temperatura actual.
- Enviar nivell de bateria actual.
- Rebre la configuració respecte a la temperatura.
- Rebre la configuració respecte al nivell de bateria.
- Enviar les 3 possibles alarmes que pot detectar el node.
- Enviar missatges de configuració de la correcta comunicació entre els nodes.

Al punt 4.1.1 d'aquesta memòria es desenvolupen, amb més detall, quines estructures tenen els missatges.

3.4. Firmware Node Remot.

El firmware del node remot té gairebé el mateix desenvolupament que el del node central, tret del fet que el node remot no es podrà comunicar de forma sèrie amb un PC, per la qual cosa en el node remot s'ha implementat el mateix funcionament que en el node central excepte la part de la comunicació via port USB.

El node remot es comporta de la mateixa manera que el node central, activa els leds de visualització de la mateixa manera i fa servir el mateix tipus de missatges però la comunicació es fa única i exclusivament via ràdio.
De la mateixa forma que el node central fa un reconeixement (ACK) dels missatges que envia el node central al node remot via radio, el node remot també realitza aquest reconeixement en sentit contrari. D'aquesta forma, la comunicació és segura ja que si qualsevol dels nodes detecta que el node receptor no ha rebut el missatge el tornarà a enviar fins que ho rebi.

4. Descripció detallada del sistema.

A continuació es detallaran tots els aspectes de disseny i desenvolupament utilitzats per a aconseguir que el sistema interaccioni amb tots els elements d'una forma eficient i correcta.

4.1. Software aplicació central PC.

L'aplicació central ha estat desenvolupada, com ja s'ha indicat, amb l'ajuda de les llibreries Qt i amb el llenguatge de programació C++.

L'aplicació s'ha dividit en dos blocs o fils d'execució clarament diferenciats, per una banda es necessita un procés que llegeixi contínuament dades del port sèrie del PC per poder enviar els diferents missatges rebuts cap a la seva funció.
I per altra banda, el fil principal d'execució s'encarrega d'actualitzar les dades a la interfície gràfica, fer els càlculs adients amb els missatges que es reben pel port sèrie (veure annex 10.1) i, quan es necessari, enviar dades pel port sèrie.

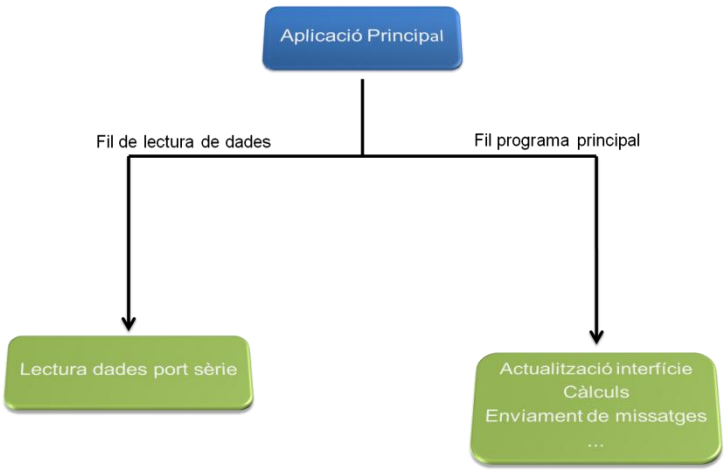


Figura 12. Divisió de l'aplicació principal

La necessitat de crear un fil d'execució paral·lel al fil d'execució principal, ve relacionat perquè s'ha de saber en tot moment si han arribat missatges al port sèrie. I si es realitza una funció que estigui contínuament llegint del port sèrie, no hi ha moments en que la interfície gràfica es pugui actualitzar i que es puguin executar els altres processos del programa. Amb la qual cosa, el programa acaba dedicant-se únicament a llegir dades del port sèrie.

4.1.1. Fil de lectura de dades des del port sèrie.

Per a realitzar aquest fil s'ha creat un QThread (`Thread` en C++ o equivalent a `fork` en C).
Un QThread, segons la documentació que ens proporciona el projecte Qt (veure bibliografia): és un fil de control dins del mateix programa, que pot compartir dades amb altres processos però s'executa independentment del fil principal, com un sistema multitasques.

Aleshores, el programa principal, un cop arrenca la interfície gràfica i s'indica en quin port USB està connectat el node central, passa a la creació d'un fil que serà l'encarregat de llegir les dades que vagin arribant al port sèrie, i cada cop que hagi rebut dades, emetrà un senyal que avisarà a un altre procés del fil d'execució principal.

Per a la lectura de missatges pel port sèrie s'ha fet servir la llibreria codificada en C, `serialsource`, que es proporcionada amb la instal·lació de l'entorn TinyOS i ens permet obrir, tancar , llegir i escriure dades en el port sèrie que s'indiqui.

Aquesta llibreria ens facilita la interacció amb el port sèrie fent servir les següents funcions:

- `open_serial_source`: Crea un descriptor des del qual podrem escriure o llegir dades del port sèrie indicat com a paràmetre en la crida de la funció.
- `close_serial_source`: Tanca el descriptor que s'ha creat amb la funció `open_serial_source`.
- `read_serial_packet`: Llegeix un missatge del descriptor creat amb la funció `open_serial_source`.
- `write_serial_packet`: Escriu un missatge al descriptor creat amb la funció `open_serial_source`.

El diagrama de flux d'aquest fil es mostra a la figura 13:

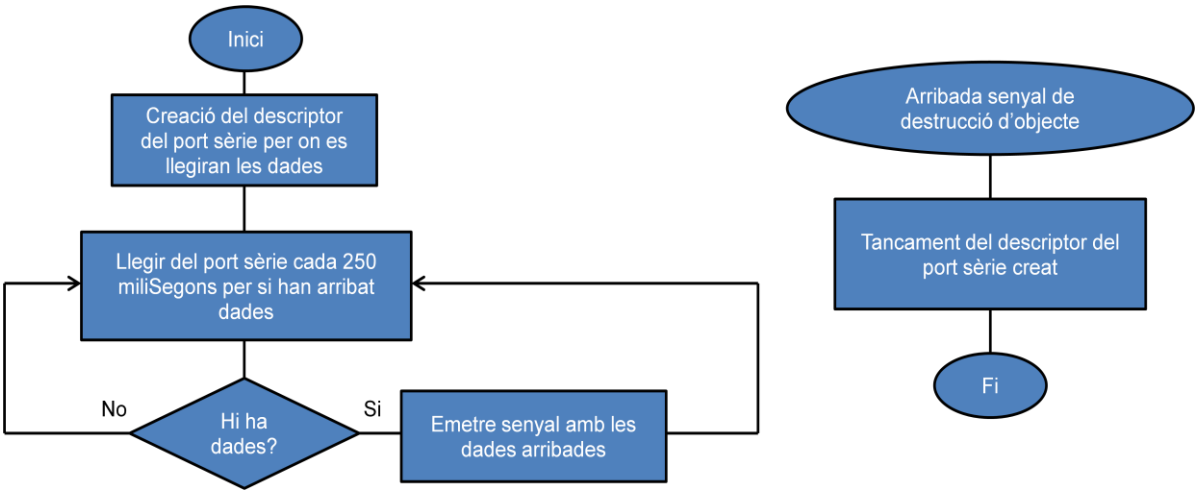


Figura 13. Diagrama de flux del fil d'execució de lectura de dades

Quan des del programa principal es crea un fil de lectura,

- es crea un descriptor que ens servirà per a dirigir-nos al port sèrie on està connectat el node central (funció `open_serial_source`).
- Es llegeix periòdicament el port sèrie (funció `read_serial_packet)`, dirigint-nos-hi amb el descriptor creat en el pas anterior.
- Si hi ha dades s'emet una senyal amb les dades rebudes cap al fil d'execució principal, on s'executarà la funció que s'ha denominat "Classificadora de missatges" en el diagrama de flux de la figura 14.
- Sinó hi ha dades, es torna a llegir amb la periodicitat indicada al diagrama de la figura 13.

S'ha donat un interval entre lectura i lectura de 250 milisegons perquè fer una lectura continua del port sèrie, sense haver-hi dades (com pot ocorre en alguns moments), carregava l'aplicació i no es llegien els missatges sencers.

Quan arriba el tancament de l'aplicació central es crida al tancament del fil, i s'han d'alliberar els recursos que s'han creat dins aquest fil per a no deixar-los adjudicats a la nostra aplicació sense que aquesta estigui en execució.
Per això s'han d'executar les accions següents (indicades al diagrama de la figura 13):

- Tancar el descriptor creat per a llegir les dades del port sèrie .
- Aturar l'execució del fil.

4.1.2. Fil d'execució principal.

El fil d'execució principal a d'encarregar-se de la resta de funcionalitats del sistema:

- Distinció del tipus de missatge arribat i cridar a la funció necessària.
- Actualització de la interfície gràfica amb les dades rebudes i els càlculs adients (veure punt de l'annex 10.1).
- Enregistrament en fitxer de text dels successos ocorreguts durant el funcionament del sistema.
- Enviament de dades necessàries per als nodes.
- Interacció amb l'usuari.

El diagrama de flux que mostra el funcionament de la classificació dels missatges de l'aplicació central, que bàsicament és el control total de l'aplicació central, es mostra a la figura 14.

S'ha d'indicar que, dins de les tasques que es mostren en el diagrama de flux de la figura 14, hi han petites tasques, que indicar-les dins del diagrama el faria molt gran i a la vegada una mica incomprensible, per això, es mostra el funcionament en base a les funcions bàsiques que s'han de realitzar.

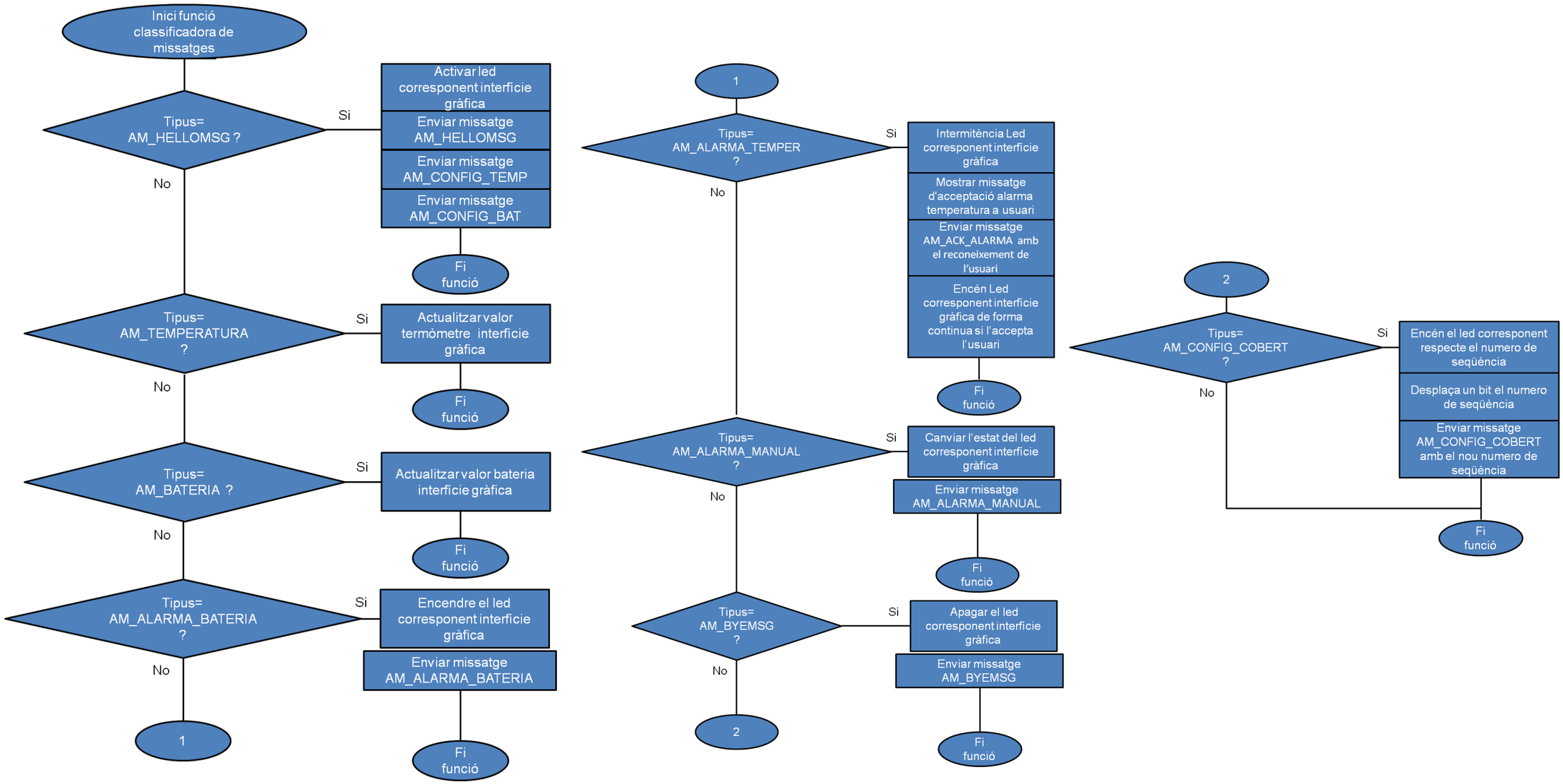


Figura 14. Diagrama de flux de la distinció de missatges

A continuació es mostra les possibles actuacions de l'usuari amb la interfície gràfica, i les operacions que realitzarà l'aplicació principal

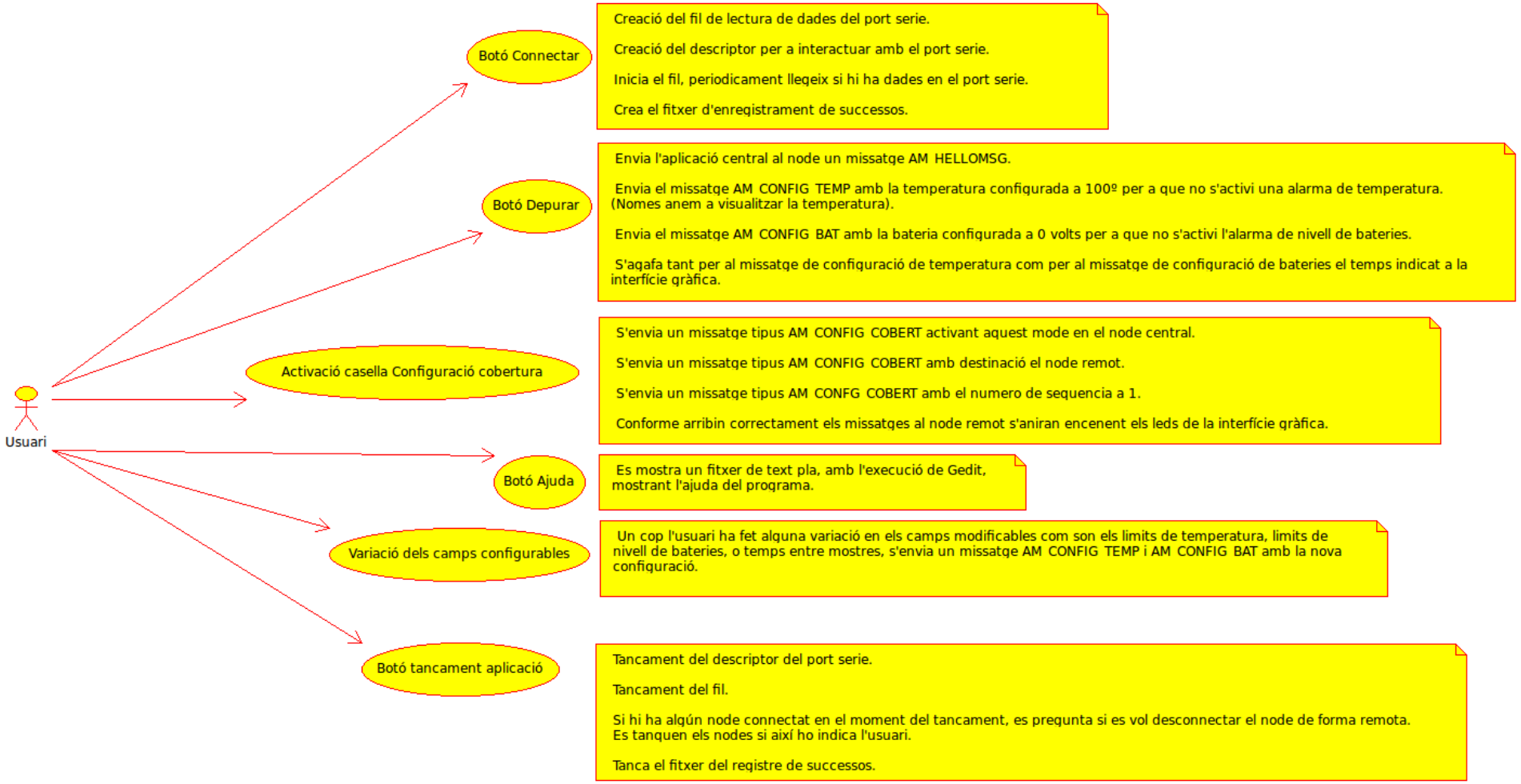


Figura 15. Interacció Usuari - Interfície gràfica

4.1.3. Estructura dels missatges utilitzats.

Com s'ha indicat en el punt 3 d'aquesta memòria, els nodes interaccionen amb l'aplicació central mitjançant missatges.

L'estructura dels missatges utilitzats pels nodes i per l'aplicació central es poden dividir en dues seccions, la capçalera i la càrrega útil.
L'estructura dels missatges en TinyOS té la següent forma:

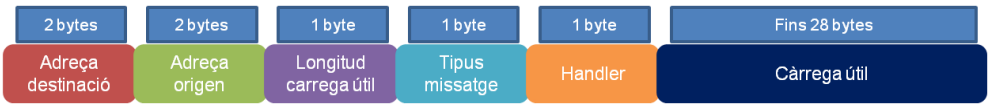


Figura 16. Estructura de missatges en TinyOS

Com s'observa en la figura 16, el missatge està dividit en dues parts, la capçalera (creada amb l'ajuda de TinyOS) i la càrrega útil que serà la zona on aniran les estructures de missatges que s'hagin creat.
Per a la implementació d'aquest sistema s'han definit 10 tipus de missatges:

- Missatge (AM_HELLOMSG, tipus=40): Aquest missatge l'envia el node que es vol connectar a l'aplicació central i aquesta li contesta amb el mateix tipus de missatge per a que la connexió es faci efectiva. L'aplicació central pot enviar aquest tipus de missatge per a indicar a un node que es connecti sense la necessitat de rebre prèviament cap missatge d'aquest node.

- Missatge (AM_CONFIG_TEMP, tipus=41): Aquest missatge l'envia l'aplicació central cap al node que acaba de demanar connexió. Pretén indicar al node la temperatura màxima i la periodicitat amb què ha d'enviar els missatges de temperatura.

- Missatge (AM_CONFIG_BAT, tipus=42): Aquest missatge és igual que l'anterior, però en comptes de referir-se a dades de temperatura, fa referència al nivell de bateria.

- Missatge (AM_TEMPERATURA, tipus=43): Aquest missatge només pot ser enviat pels nodes. Aquest missatge contindrà la temperatura actual mesurada pel node. L'aplicació central calcularà els ºC equivalents al valor enviat pel node tal com s'explica a l'annex 10.1.1.

- Missatge (AM_BATERIA, tipus=44): Aquest missatge, igual que el missatge anterior, l'envia qualsevol dels nodes indicant el nivell de bateria que presenta. L'aplicació central calcularà els volts equivalents al valor enviat pel node tal com s'explica a l'annex 10.1.2.
- Missatge (AM_ALARMA_BATERIA, tipus=45): Aquest missatge només pot ser enviat pels nodes. Indica que el nivell de bateria actual està per sota del llindar configurat.

- Missatge (AM_ALARMA_TEMPER, tipus=46): Aquest missatge només pot ser enviat pels nodes. Indica que la temperatura actual ha sobrepassat el nivell establert a la configuració.

- Missatge (AM_ALARMA_MANUAL, tipus=47): Quan un usuari premi el polsador del node, aquest enviarà a l'aplicació central un missatge d'aquest tipus. L'aplicació li tornarà aquest tipus de missatge indicant-li que ha rebut el missatge correctament.

- Missatge (AM_BYEMSG, tipus=48): Aquest missatge l'envia el node que es vol desconnectar de l'aplicació central i aquesta li contesta amb el mateix tipus de missatge per a que la desconnexió es faci efectiva. L'aplicació central pot enviar aquest tipus de missatge per a indicar a un node que es desconnecti sense la necessitat de rebre prèviament cap missatge d'aquest node.

- Missatge (AM_CONFIG_COBERT, tipus=49): Aquest missatge, enviat per l'aplicació central i també pels nodes, pretén realitzar una senzilla comunicació per a comprovar que la comunicació és correcta entre els 2 nodes.

- Missatge (AM_ACK_ALARMA, tipus=50): Quan arriba a l'aplicació central una alarma de temperatura l'usuari ha d'indicar si l'accepta o no; en funció de la seva elecció s'envia "cert" o "fals" respectivament, amb aquest tipus de missatge.

Amb la funció definida de cada missatge, s'han creat les classes mostrades a la figura 17, així cada missatge serà tractat com un objecte i aquests donaran el suport necessari a l'aplicació central:

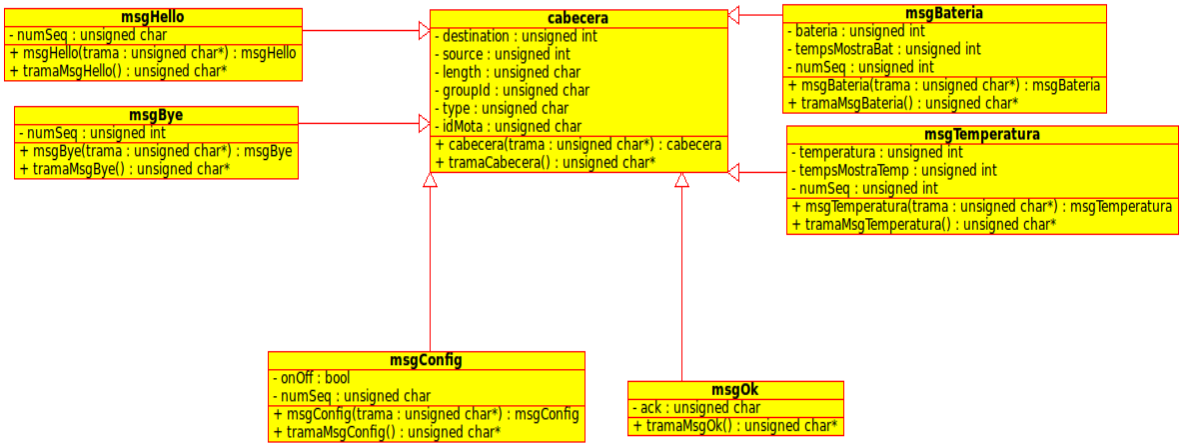


Figura 17. Classes definides per als missatges de l'aplicació central

Com s'observa en el diagrama de classes de la figura 17, tots els missatges hereten de la classe "cabecera", això es així perquè tots els missatges tenen la mateixa estructura de capçalera, l'únic aspecte que varia en cada un d'ells és el tipus de missatge i els continguts de la càrrega útil.
La capçalera que s'ha creat en el diagrama de classes té inclòs el camp "idMota", ja que tots els missatges que s'ha definit per a la implementació del sistema ho tenen, per això, la capçalera que s'ha definit no és exactament igual que la capçalera que genera TinyOS per a enviar un missatge.

Altre aspecte a tenir en compte a cada classe és que cadascuna té una funció "tramaMsg..." que retorna una cadena de caràcters. Quan es tingui un objecte missatge creat i llest per a ser enviat, aquesta funció ens fa la conversió de l'objecte en un array de caràcters que s'enviarà byte a byte(caràcter a caràcter) pel port sèrie.

4.2. Firmware Node Central.

El codi desenvolupat per al node central ha intentat ser el més modular possible, per a poder adaptar el firmware a les possibles variacions d'un futur. Per aquest motiu, per al funcionament complet del sistema s'han realitzat els següents mòduls:

- Msgs.h:

Dins aquest fitxer de capçalera *.h*, s'han generat les estructures pertanyents a tots els missatges que els nodes poden rebre o enviar.

L'estructura dels missatges, tant per als nodes com per a l'aplicació central, és la mateixa, però l'estructura dins el codi dels nodes es defineix com es fa la definició d'una estructura en llenguatge C.

- BaseStation:

Aquest mòdul es proporciona amb la instal·lació de l'entorn TinyOS i resulta molt útil perquè la seva finalitat és enviar els missatges que rep pel port sèrie a través de la interfície ràdio, i al contrari, enviar els missatges que rep via ràdio a traves del port sèrie.

Aquest mòdul crea dues cues de missatges, una per als missatges que arriben pel port sèrie, i s'hauran d'enviar via ràdio, i una altra cua per als missatges que arriben via ràdio, i han d'enviar-se cap al port sèrie.

Un cop rebut un missatge i posat a la cua que li correspon, s'envia una tasca a la pila de tasques per a que quan el node vagi agafant-les per a executar-les, faci l'enviament que correspongui.

Per a adaptar-lo al nostre sistema s'han realitzat una sèrie de modificacions en aquest mòdul:

- S'ha eliminat la interfície per activar/desactivar els leds. Aquest mòdul no actua sobre cap led.
- En rebre un missatge pel port sèrie s'identifica qui és el destinatari (es compara el camp destí del missatge amb la macro `TOS_NODE_ID` que ens indica el id del node que ha rebut el missatge). Si el missatge és per al node remot, aquest es posa en la cua dels missatges que han de sortir via ràdio. Si el missatge és per al node central, mira el tipus de missatge i realitza l'actuació corresponent.
- A la tasca que s'encarrega d'enviar els missatges via ràdio, s'ha incorporat la interfície "`PacketAcknowledgments`" que ens permet saber si el missatge ha arribat al seu destinatari. Si no es rep el reconeixement corresponent, es torna a enviar el mateix missatge, perquè no es treu el missatge de la cua.
- S'han definit també punters a les estructures dels missatges `configTemp`, `configBat`, `reconAlarmaTemp` i `configCobertura`, per a poder agafar la càrrega útil quan ho necessitem, creant missatges dels tipus definits.

Quan arriba un missatge de configuració de cobertura, de configuració de temperatura, de configuració de nivell de bateria o d'un reconeixement d'alarma de temperatura, s'agafa la càrrega útil del missatge i es converteix en l'estructura del tipus de missatge

rebut, i d'aquesta manera s'envien, aprofitant les interfícies `Notify` o `Set`, els valors corresponents cap al mòdul que s'hagi d'actualitzar.

Aquest mòdul és pràcticament el més important del desenvolupament del node central, ja que és aquí on es reben tots els missatges i es deriven els valors corresponents cap als mòduls que estiguin involucrats en cada tipus de missatge.

- Blink:

El mòdul Blink nomes s'encarrega de veure quants cops s'ha activat el sensor d'efecte Hall i de l'estat del led verd, fent intermitències o encenent-lo de forma continua.

La funció d'inici d'aquest mòdul (funció `Boot.booted`), carrega un temporitzador periòdic amb interval de temps 1 segon. Aquest temporitzador, cada cop que s'esgoti, farà la intermitència del led verd.

El sensor d'efecte Hall es configura com una interrupció Hardware, així cada cop que s'activa aquest sensor s'executa una funció, d'aquesta manera es tracta millor aquest comportament en comptes de estar realitzant un Polling cada cert temps, que seria un altre tipus de funcionament en la lectura de perifèrics en un microcontrolador.

El diagrama de funcionament de l'activació del sensor d'efecte Hall del mòdul Blink es mostra en la figura 18:

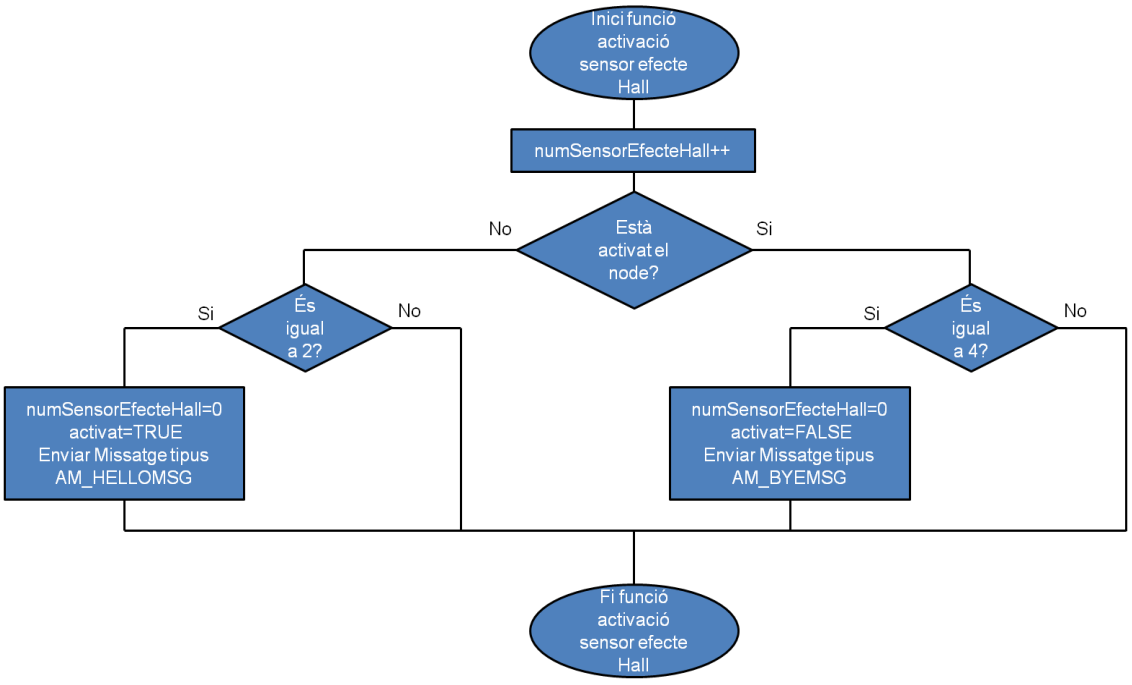


Figura 18. Diagrama de funcionament de l'activació del sensor d'efecte Hall

Encara que l'estat del node s'ha mostrat al diagrama de flux quan està actiu o no, per a clarificar dins del diagrama l'estat del node, la variable real de l'estat del node, així com

l'actuació en el led verd es realitza un cop el node rep des de l'aplicació central, un missatge del tipus AM_HELLOMSG, indicant que s'ha rebut el missatge de connexió, o un missatge del tipus AM_BYEMSG, indicant que es vol desconnectar el node.

Quan arriba un missatge AM_HELLOMSG, s'atura el temporitzador que feia parpellejar el led verd, i el deixa encès de forma continua. Posa l'estat del node a actiu, fent que la resta de mòduls s'assabentin que el node es troba actiu.
Pel contrari, quan arriba un missatge AM_BYEMSG, s'apaguen tots els leds per defecte, i es fa un reinici del sistema per a que el node es quedi amb l'estat inicial.

- Cobertura:

Quan es rep des del mòdul BaseStation el missatge tipus AM_CONFIG_COBERT, el mòdul Cobertura nomes canvia l'estat dels 3 leds cada segon, d'aquesta manera mostra el node central que es troba en estat de configuració de cobertura.

La intermitència dels 3 leds es realitza amb l'ajuda d'un objecte temporitzador, que s'executa de forma periòdica cada segon, així cada cop que s'esgota el temporitzador s'executa la funció Fired, pròpia del temporitzador, que realitza un canvi en l'estat de tots 3 leds.

Quan des de BaseStation es notifica que ja no s'està realitzant la prova de configuració de cobertura, s'atura el temporitzador.

A continuació es mostra el diagrama de funcionament:

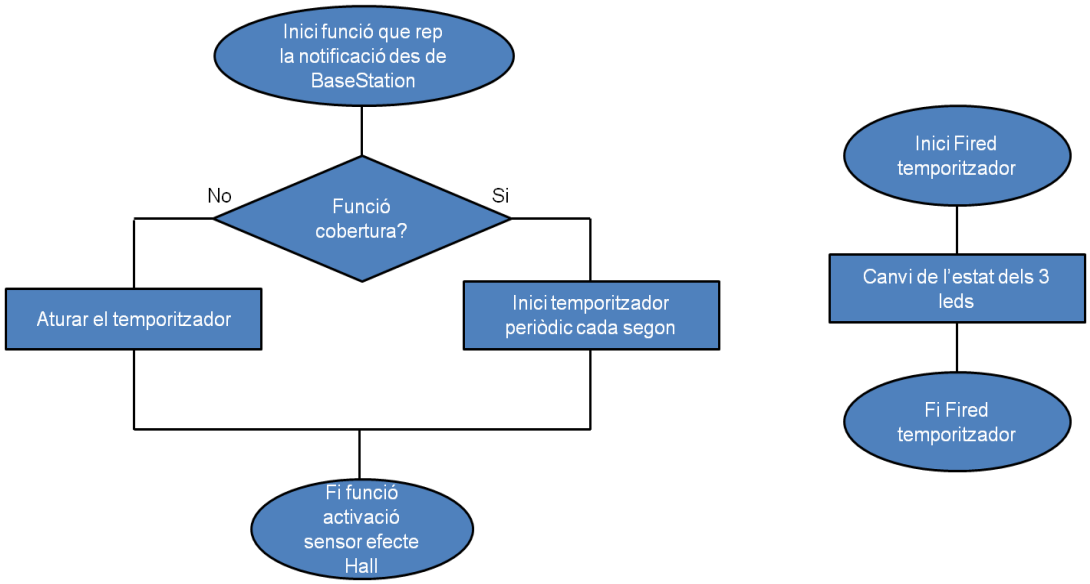


Figura 19. Diagrama de funcionament mòdul cobertura

- lectorBat i lectorTemp:

Aquests 2 mòduls treballen de la mateixa forma, nomes presenten la diferencia en el sensor que llegeix cadascun dels mòduls i el convertidor A/D al que es dirigeixen.

Per a explicar el funcionament dels 2 mòduls, es mostra en la figura 20, el diagrama de flux del mòdul lectorBat:

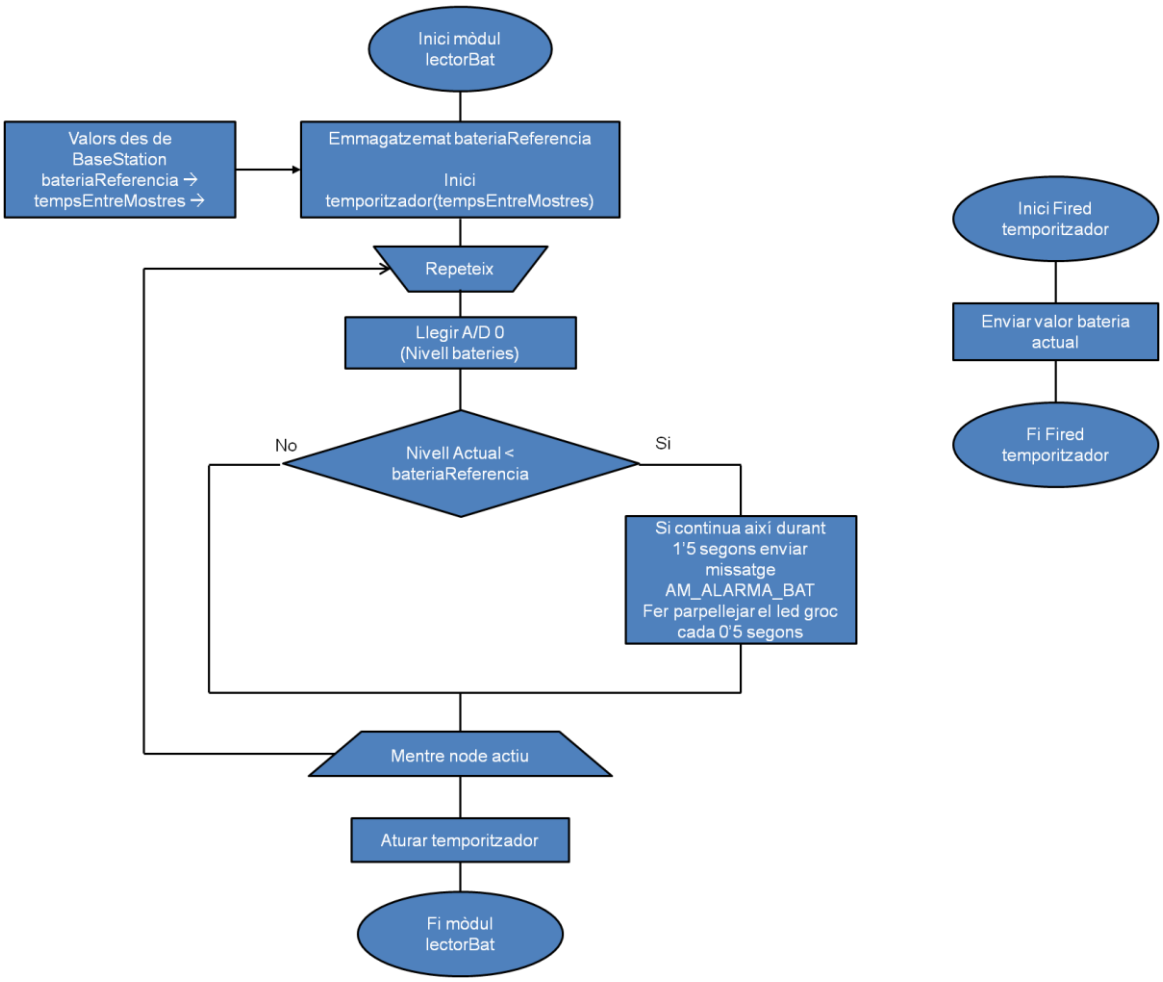


Figura 20. Diagrama de funcionament mòdul lectorBat

Des de BaseStation, s'emmagatzemen els valors de temperatura i bateria de referència, per a tenir els valors amb els quals s'estaran realitzant comparacions de forma continua.

La notificació que s'envia amb el temps entre mostres que s'ha configurat des de l'aplicació central executarà un temporitzador periòdic, amb el valor indicat en la notificació, i cada cop que s'esgoti el temporitzador, s'enviarà un missatge tipus AM_BATERIA o AM_TEMPERATURA amb el valor actual del nivell de bateria o el nivell de temperatura, respectivament.

Si es sobrepassa el nivell de temperatura mesurat pel sensor de temperatura (convertidor A/D numero 2 del microcontrolador) del valor prefixat en la configuració durant 1'5 segons, es fa parpellejar el led vermell amb una latència de 0'5 segons; s'envia un missatge amb aquest fet, i un cop arriba el missatge al destinatari, la latència del led passa a ser 0'25 segons.
Quan l'aplicació central ha rebut l'alarma de temperatura, aquest mòdul espera que l'usuari l'accepti o no per a deixar el led vermell encès de forma continua o parpellejant.
Aquesta alarma desapareixerà si la temperatura baixa 1ºC per sota del límit establert a la configuració.

Per al nivell de bateria, si disminueix el valor del nivell de la bateria (convertidor A/D numero 0 del microcontrolador) del valor prefixat en la configuració, també durant 1'5 segons, es fa parpellejar el led groc amb una latència de 0'5 segons.
Aquesta alarma es pot desactivar si el nivell de bateria és 0'05 volts major del nivell establert a la configuració.

La histeresi de 1'5 segons configurada en els dos mòduls, lector de temperatura i lector de nivell de bateria, s'ha realitzat perquè no hi hagués "oscil·lacions" en el moment que s'acostés la temperatura o el nivell de bateria al límit establert, fent que l'aplicació rebés un munt de missatges cada cop que es sobrepasses el límit.

El cicle continu que està realitzant el node, mesurant i detectant alarmes de temperatura o de nivell de bateria, es realitza mitjançant una estructura de tasques, que es van "apilant" cada cop que es realitza una funció, i modificant l'estat en que es troba en cada moment dins de la iteració del cicle per a saber quina serà la següent tasca que s'haurà d'apilar.

Aquesta funcionalitat de les tasques, fan que el node simuli un sistema multi tasca, en que cada moment s'estan executant diverses tasques, però en realitat, s'executa una tasca, quan s'acaba es comença amb una altra, etc.

Per a la lectura dels convertidors A/D dels quals disposa el microcontrolador, TinyOS proporciona una capa d'abstracció per a realitzar una conversió d'un valor analògic a digital, per això a l'hora de fer una lectura d'un convertidor d'aquest tipus, s'ha de demanar que realitzi la conversió, i amb un esdeveniment, que serà executat de forma automàtica, es retornarà el valor demanat en la precisió que s'hagi indicat (8 bits, 16 bits o 32 bits), quan el valor estigui llest. Aquest tipus d'operacions, en TinyOS se li

denominen Split-Phase (fase dividida), que significa, es demana una certa acció, i el microcontrolador avisa quan la té llesta.

- polsUsuari:

Aquest mòdul, funciona de forma semblant a com es feia amb el sensor d'efecte Hall.
El polsador d'usuari es configura per a generar una interrupció hardware cada cop que es premi el botó.

Un cop es premi el polsador, es canvia l'estat del led groc (si estava encès l'apaga, i si estava apagat l'encén), i envia un missatge tipus AM_ALARMA_MANUAL.
Des de l'aplicació central, es té en consideració que una pulsació activa l'alarma i altre pulsació la desactiva.

S'ha pres aquesta decisió de disseny, per a tenir un funcionament com per exemple el següent:

- Una persona que hi ha a la sala on es troba un dels 2 nodes, percep un fet estrany, (una persona estesa al terra, una finestra trencada,etc.) llavors prem el polsador d'alarma manual.
- A la salsa de control es rep aquesta alarma i es mostrada en pantalla, llavors personal de seguretat s'apropa a la sala on s'ha produït aquest fet, i un cop ha revisat la sala i es troba tot en perfectes condicions, el mateix personal de seguretat prem un altre cop el polsador per a desactivar aquesta alarma, d'aquesta manera s'apaga el led groc, i s'esborra l'alarma de l'aplicació central.

Aquest funcionament fa que la implementació d'aquest mòdul sigui molt senzilla:

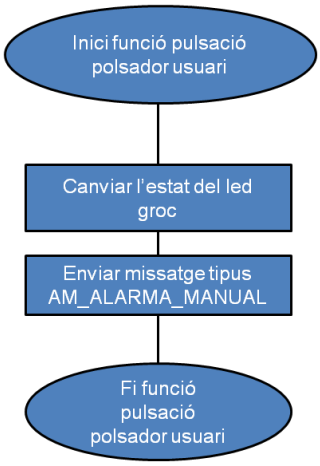


Figura 21. Diagrama de funcionament mòdul polsUsuari

Com l'estat del led groc depèn de les possibles alarmes del nivell de bateria i de l'alarma manual d'usuari, s'ha pres la decisió de fer més prioritària l'alarma del polsador manual, així, si coincideixen a l'hora l'alarma de nivell de bateria amb l'alarma del polsador manual, el led groc es quedarà encès de forma continua. Un cop es premi el boto per a desactivar aquesta alarma, el led continuarà fent les intermitències de l'alarma de nivell de bateria, si aquesta encara es present.
A l'aplicació central es mostren totes dues alarmes.

4.3. Firmware Node Remot.

Degut a que el comportament dels 2 nodes és pràcticament igual, excepte en la manera que té el node remot de comunicar-se, aquest nomes es comunica via ràdio, els següents mòduls,

- Blink
- Cobertura
- lectorBat
- lectorTemp
- polsUsuari

tenen el mateix funcionament que els descrits en el punt anterior.

Respecte aquests mòduls nomes s'hauria de subratllar que, cada mòdul implementa la interfície de recepció del tipus de missatge involucrat a cada mòdul, acció que realitzava el mòdul BaseStation en el node central i que li permetia rebre qualsevol tipus de missatge.
A conseqüència d'això, les accions que es realitzaven en el mòdul BaseStation en rebre un tipus de missatge o un altre, ara es realitzen en el mòdul que tingui implementat l'esdeveniment de recepció del missatge en qüestió.
Tot seguit s'indiquen els tipus de missatges que rep cada mòdul:

- Blink
  - AM_HELLOMSG
  - AM_BYEMSG
- Cobertura
  - AM_CONFIG_COBERT
- lectorBat
  - AM_CONFIG_BAT
  - AM_BATERIA
  - AM_ALARMA_BATERIA

- lectorTemp
  - AM_CONFIG_TEMP
  - AM_TEMPERATURA
  - AM_ALARMA_TEMPER
- polsUsuari
  - AM_ALARMA_MANUAL

Com el node remot nomes es comunica via ràdio amb el node central, s'ha implementat el sistema de reconeixement de missatges, incloent-hi la interfície "PacketAcknowledgements", que ens permet indicar que es vol rebre un reconeixement quan el destinatari ha rebut correctament el missatge.

Aquesta interfície es pot incloure en el codi, un cop s'ha fet la definició d'una variable en el fitxer de compilació:

```
CFLAGS += -DRF230_HARDWARE_ACK
```

En la implementació d'aquest sistema s'ha demanat el reconeixement en tots els missatges que s'envien per ràdio, tant al node central com al node remot, per a tenir, d'aquesta forma, una comunicació fiable entre tots dos, i evitar en tot el possible pèrdues de missatges.

El diagrama de flux que mostra bàsicament el funcionament del reconeixement es mostra a la figura 22:

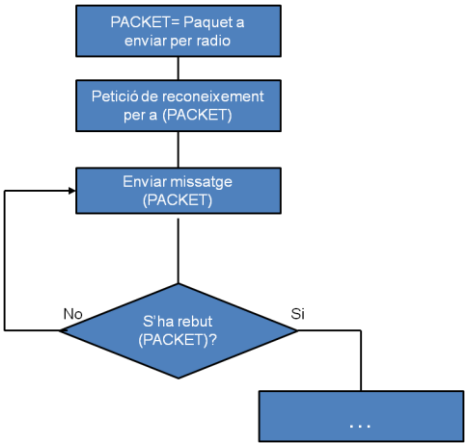


Figura 22. Diagrama de funcionament reconeixement missatges

Per finalitzar el desenvolupament del node remot, quedaria únicament un mòdul, que no el té el node central:

- alarmaTemp:

Aquest mòdul s'encarrega de rebre el missatge de reconeixement de l'alarma de temperatura que s'ha enviat a l'aplicació central.
En funció de la resposta de l'usuari, s'envia un missatge des de l'aplicació central indicant el reconeixement o no de l'alarma de temperatura, i un cop es rep aquest missatge al node remot, s'identifica a l'estructura del missatge si l'usuari l'ha acceptat o no, per a encendre de forma continua el led vermell o deixar-lo fent intermitències.

5. Viabilitat tècnica.

Amb les alternatives que hi ha al mercat, vistes algunes a l'apartat 2.2 d'aquesta memòria, és clar que aquest projecte no pot competir amb solucions professionals com les que existeixen.

Però el que és més important és que aquest projecte ha permès al desenvolupador del sistema desenvolupar un coneixement ampli de les tecnologies WSN (wireless sensors networks) i veure que el camp d'aplicació per aquests nodes és molt ampli i amb moltes possibilitats.

Aprofitant els coneixements adquirits en la realització d'aquest projecte i amb un grup de treball més gran, es podria afrontar, amb aquesta tecnologia, un sistema d'alarmes d'incendis que fos capaç de competir amb les solucions que ja hi han al mercat.

Per altra banda, el sistema que s'ha desenvolupat és més fàcilment adaptable a altres funcions que no pas les solucions comercials trobades, que es dirigeixen a un objectiu clar.
El sistema desenvolupat, a més de fer la detecció d'incendis amb uns dispositius de baix cost i de baix consum, podrien també, per exemple, realitzar estadístiques de la temperatura que hi ha normalment en certs llocs. Això ens permetria tenir dades de quins llocs s'escalfen més o menys, per a proporcionar dades d'estudi.

Per posar un exemple, aquest sistema podria realitzar captures periòdiques de la temperatura en instal·lacions que fossin motiu d'estudi, com ara sales de servidors, oficines, sales de museus, etc. Amb aquestes dades es podria realitzar el control de la temperatura (activar calefaccions o aires condicionats) per a mantenir-les dins uns rangs establerts.

En resum, el nostre sistema no pot competir com un detector d'incendis contra solucions professionals, però és més fàcil d'adaptar a funcionalitats semblants, amb poc cost addicional.

6. Valoració econòmica.

A continuació es mostra un pressupost aproximat que inclou el desenvolupament realitzat i la instal·lació del sistema.
El pressupost només fa la valoració de costos per al control de dues sales, perquè el desenvolupament de l'aplicació central es troba en aquesta fase. Si el client estigués interessat en controlar més sales s'haurien de realitzar algunes modificacions.

Descripció	**Quantitat**	**Preu/unitat**	**Preu total**
Node amb microcontrolador, interfície ràdio, 3 leds, sensors de temperatura, fotosensor i sensor d'efecte Hall. Compartiment per a alimentar amb piles AA de 1'5V. Connexió USB. (Es dona com a exemple el preu del node següent: XM1000 compatible amb TelosB)	**2**	**85€**	**170€**
Desenvolupament dels codis necessaris per al sistema.	**150 Hores**	**30€**	**4500€**
Ordinador personal per a l'execució del programa principal. (Es dona com a exemple el preu del portàtil següent: Acer Aspire 5250,4Gb Ram, 320 Gb disc dur, DVD gravador, pantalla de 15'6 polzades)	**1**	**400€**	**400€**
Instal·lació del software i el hardware	**2 Hores**	**30€**	**60€**
Total			**5130€**

Els preus/hora s'han agafat de forma aproximada comparant-los amb una empresa dedicada a la implantació de sistemes de telecontrol on s'apliquen conceptes de programació i instal·lació de perifèrics.

7. Conclusions.

7.1. Conclusions.

Dels objectius que s'havien definit en un principi per al desenvolupament de tot el sistema, indicats al punt 1.3 d'aquesta memòria,

- Creació d'una xarxa de sensors sense fils
- Monitorar el nivell de temperatura en les sales a controlar
- Monitorar l'estat del nivell de bateria dels sensors
- Control d'activació/desactivació mitjançant un sensor magnètic
- Visualització de senyals
- Comunicació entre els nodes i l'aplicació central
- Proporcionar una interfície gràfica i senzilla per a l'usuari
- Enregistrament periòdic dels successos
- Eines de suport per a la instal·lació dels sensors

S'han aconseguit desenvolupar i provar tots amb un funcionament òptim.

Encara que els objectius s'han aconseguit tots, ha quedat pendent, assolir la implementació d'un WatchDog amb més garanties, que era un dels requisits del sistema.
El que sí s'ha realitzat ha estat un "auto-reset" per a controlar el funcionament del mòdul Blink; si el led deixés de fer intermitències per alguna causa es reiniciaria el sistema.

Per altra banda, a l'aplicació central, si es perd la comunicació durant el doble de temps indicat als paràmetres de configuració de la interfície gràfica amb algun dels nodes, s'avisa i s'intenta reconnectar fins a tres cops amb el node que ha perdut la comunicació.
Aquest era un pas previ per a implementar un sistema WatchDog amb garanties perquè es pretenia verificar mitjançant un missatge periòdic, entre nodes i aplicació central, que si en algun moment no existia aquest missatge es fes un "reset" automàticament.
Aquest fet no s'ha explicat al punt 4 perquè aquesta funcionalitat no esta implementada de forma completa.

7.2. Proposta de millores.

Si aquest projecte es continués desenvolupant, les tasques que s'intentarien dur a terme serien les següents:

- Implementació completa del sistema WatchDog per a un funcionament amb mes garanties.
- Realització d'aturades en la captura de dades dels sensors per a que les bateries s'esgotessin més lentament.
- Implementació al mòdul *configurar la cobertura dels nodes*,el codi necessari per a poder visualitzar les potencies dels nodes.
- Implementació del Web adient per a poder controlar l'aplicació desenvolupada a través d'Internet des de qualsevol lloc del món.
- Enregistrament de les dades de temperatura que es reben periòdicament, per a poder observar l'evolució de la temperatura i així poder estudiar solucions de calefacció o aire condicionat.

7.3. Autoavaluació.

El grau de satisfacció del desenvolupament d'aquest sistema és alt, però no és del 100%.
Per una banda, hi ha hagut situacions en les que el funcionament del sistema m'ha fet perdre una mica la paciència perquè volia que el sistema funcionés d'una forma molt i molt eficient, tenia molt clar el funcionament que s'havia de desenvolupar, però hi ha hagut situacions amb les comunicacions via ràdio que m'han costat moltísim.

Per altra banda, he après que el més important a l'hora d'afrontar un projecte és la planificació del temps, la definició de les tasques de forma molt senzilla, i no quedar-se massa temps aturat en una tasca sense continuar avançant.

8. Glossari.

Ack: (abreviatura de acknowledgement), en telecomunicacions representa una indicació per a saber que un missatge ha estat reconegut pel seu destinatari, es a dir, que s'ha transmès de forma correcta.

Array: Agrupament d'un mateix tipus de variable al qual ens podem referir com una sola variable; o ens podem referir a un element en concret d'aquest grup.

BootLoader: Codi resident en la memòria de programa d'alguns microcontroladors, que permet carregar un firmware nou sense necessitat de programadors especials.

Bit: Unitat mínima d'informació en sistemes electrònics digitals. Es representa amb 0 ó 1.

Byte: Conjunt de 8 bits.

Compilació: Procediment mitjançant el qual es converteix qualsevol llenguatge de programació (C, C++, Pascal,...) en codis que el processador per al qual es realitza aquesta compilació es capaç d'interpretar i executar.

Comunicació sèrie: Comunicació que es basa en transmetre/rebre informació bit a bit per només dos fils. Tots dos han de ser coneixedors de quina forma tindrà la informació, com es comença la transmissió, com acaba i a la velocitat que s'emetrà cada bit.

Convertidor analògic/digital: Sistema hardware que s'encarrega de convertir qualsevol valor de tensió analògic en una paraula de X bits.

Firmware: El codi de més baix nivell que executarà un microcontrolador o microprocessador i que està directament lligat amb el hardware, es a dir, el llenguatge màquina que executarà el processador en qüestió.

Hardware: Conjunt de components electrònics que conformen un dispositiu.

Histeresi: Tendència d'un cert esdeveniment que té en compte l'estat actual i de quin estat prové.

Ide: Sigles d'Entorn de desenvolupament integrat. Software que facilita la codificació de programes als programadors. Existeixen diferents entorns per a diferents llenguatges de programació.

Interrupció: Senyal que aplicat a un processador/microcontrolador el fa entrar a una subrutina especial de forma automàtica, es a dir, deixa el que estava fent per posar-se amb la interrupció. Aquests senyals s'aprofiten en dispositius que necessiten atenció just en el moment que la demanen, es a dir, no poden esperar.

Kernel: Nucli del sistema operatiu.

Leds (Led Emitting Diode): Diode emissor de llum.

Maquina virtual: Software que simula un PC dins del propi PC, i permet instal·lar altre sistema operatiu diferent en aquest PC simulat.

Microcontrolador: Xip que té integrat memòria RAM, ROM, i altres perifèrics, com per exemple UART(comunicació sèrie), convertidors analògics/digitals, comunicació ràdio, etc. Existeixen molts tipus i marques comercials diferents.

Mota: Dispositiu COU24 proporcionat amb els materials per a la realització del projecte de sistemes encastats.

Multi-Hop: Topologia de comunicacions que es basa en el fet que cada element envia els missatges al node que té més proper, i aquest a l'altre més proper seu. Aquesta cadena continua fins a arribar al node central.

nesC: Llenguatge de programació que s'ha fet servir per al desenvolupament del firmware dels nodes. Es basa en components que es comuniquen amb altres components mitjançant interfícies que emeten senyals o reben comandes.

Node: veure Mota.

Notify: Interfície proporcionada per TinyOS que funciona emetent un senyal i enviant en aquest senyal el valor d'una variable definida. Així podem enviar dades a diferents components.

Open Source: També anomenat software lliure, permet compartir el software i el seu codi sense ànim de lucre.

Polling: És l'operació de realitzar contínuament consultes per a conèixer l'estat de certs perifèrics per si volen enviar o proporcionar dades.

Plugin: Complement a una aplicació per a donar altres funcionalitats que l'aplicació no tenia inicialment.

Qt: Llibreries gràfiques desenvolupades per Nokia, que proporcionen interfícies gràfiques molt enriquidores, fàcils d'implementar i gratuïtes.

Reset: Senyal per a que un processador inici la seva execució, normalment significa posar el comptador de programa a l'adreça 0, que és la primera que s'executa en un processador un cop rep un Reset o se li dona alimentació.

Sensor Efecte Hall: Sensor que proporciona un senyal en apropar-li un element magnètic.

Set: Interfície proporcionada per TinyOS que permet canviar el valor d'una variable entre mòduls diferents.

Termistor: Dispositiu que varia el valor de la seva resistència en funció de la temperatura.

Thread: (Fil d'execució), en programació, es tracta d'una part del software que s'executa de forma paral·lela a una altra aplicació.

TinyOS: Sistema operatiu desenvolupat per un consorci liderat per la Universitat de Berkley.

USB (Universal Serial Bus): Bus sèrie que defineix el protocol, els cables i els connectors.

WatchDog: Aplicació que controla el correcte funcionament d'una aplicació. Si detecta un error reinicia el dispositiu.

9. Bibliografia.

Recursos web utilitzats per al desenvolupament del projecte:

- http://cv.uoc.edu/app/mediawiki14/wiki/Inici (Durant el desenvolupament)
- http://docs.tinyos.net/tinywiki/index.php/Main_Page (Durant el desenvolupament)
- http://docs.tinyos.net/tinywiki/index.php/TEPs (Durant el desenvolupament)
- http://doc.qt.nokia.com/ (Durant el desenvolupament)
- http://releases.ubuntu.com/lucid/ (5/3/2012)
- http://www.eclipse.org/ganymede/ (5/3/2012)
- http://tos-ide.ethz.ch/wiki/index.php (6/3/2012)
- http://ww1.microchip.com/downloads/en/DeviceDoc/21942e.pdf (31/5/2012)
- http://www.atmel.com/Images/doc2549.pdf (31/5/2012)
- http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA453978 (1/6/2012)
- http://users.isc.tuc.gr/~ktountas/documents/comparison_tables.pdf (1/6/2012)
- http://www.sase.com.ar/2011/tutoriales/protocolos-para-redes-de-sensores-inalambricos-wsn/ (5/6/2012)
- http://www.hartcomm.org/protocol/about/aboutprotocol_specs.html (6/6/2012)
- http://docs.zigbee.org/zigbee-docs/dcn/09-5180.pdf (6/6/2012)
- http://standards.ieee.org/about/get/802/802.15.html (6/6/2012)
- http://www.boschsecurity.com.mx/productos/lineas_de_productos/Default.asp?nivel1=2&nivel2=1 (6/6/2012)
- http://www.morley-ias.es/index.php?option=com_zoo&task=category&category_id=95&Itemid=3 (6/6/2012)
- http://qt-apps.org/index.php?xcontentmode=4298 (5/3/2012)

Llibres utilitzats per al desenvolupament del projecte:

- **Joyanes Aguilar, Luis ; Zahonero Martínez, Ignacio.** (2005) *Programación en C. Metodología, algoritmos y estructura de datos.* Editorial: McGrawHill.

- **Márquez Garcia, Francisco Manuel.** (2009) *UNIX Programación avanzada.* Editorial: Ra-Ma.

- **Roldán Martínez, David.** (2004). *Comunicaciones inalámbricas.* Editorial: Ra-Ma.

- **Levis , Philip; Gay, David.** (2009). *TinyOS Programming.* Editorial:Cambridge.

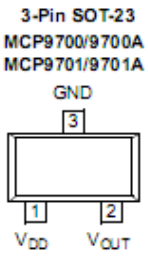
10. Annexos.

10.1. Càlcul dels valors de temperatura i tensió reals.

Quan els nodes envien missatges de temperatura i nivell de bateria, envien un valor codificat en 16 bits (2 bytes), amb el qual ens estan proporcionant un valor equivalent al valor analògic que estan llegint els convertidors A/D, però per a saber a quin valor real pertany s'han de realitzar els càlculs necessaris per a convertir aquest valor proporcionat pels nodes al valor de temperatura en graus o el nivell de bateria en volts equivalent.

10.1.1 Càlcul de la temperatura

El sensor que realitza la mesura de temperatura en els nodes és el MCP9700/9700A, a la figura 23 es mostra el sensor i la taula de característiques:



Sensor Output						
Output Voltage, T_A = 0°C	$V_{0°C}$	—	500	—	mV	**MCP9700/9700A**
Output Voltage, T_A = 0°C	$V_{0°C}$	—	400	—	mV	**MCP9701/9701A**
Temperature Coefficient	T_C	—	10.0	—	mV/°C	**MCP9700/9700A**
	T_C	—	19.5	—	mV/°C	**MCP9701/9701A**

Figura 23. Sensor de temperatura i full de característiques.

Segons s'observa en la figura 23, el valor de temperatura té una relació de 10mV/ºC i el valor de temperatura 0ºC equival a 0'5V.

Per altra banda, el full de característiques del microcontrolador Atmega 1281, que és el microcontrolador del nostre node, observem que la resolució dels convertidors A/D és de 10 bits, o sigui que com a màxim ens donaran $2^{10} = 1024$ valors, es a dir, des de 0 fins a 1023.

- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change

Figura 24. Perifèrics Atmega 1281

Quan demanem una lectura dels convertidors A/D del nostre node, configurem la tensió de referència a 2'56 Volts, això significa que la tensió equivalent que proporcionin els sensors anirà de 0 a 2'56 Volts.

Quan arribi un valor decimal (entre 0 i 1023) a l'aplicació central, i es tracti del valor de temperatura que ens proporciona el sensor de temperatura de qualsevol dels 2 nodes, podrem realitzar el càlcul dels graus amb la següent funció:

$$Temperatura\ en\ ºC = \frac{\left[\left(\frac{Valor\ decimal}{1024}\right) \cdot 2'56V\right] - 0'5V}{10\ mV}$$

La funció que hi ha entre claudàtors ens proporciona el nivell de tensió equivalent al valor que ens ha donat el convertidor A/D, i amb el nivell de tensió, apliquem les característiques del sensor de temperatura.

10.1.2 Càlcul del nivell de bateria

El càlcul per a saber quin nivell de volts té la bateria és més senzill que el càlcul de la temperatura, ja que directament estem mesurant un voltatge, no tenim que fer càlculs respecte a com evoluciona un sensor, com passava amb el sensor de temperatura.

En primer lloc, la tensió que mesurem vindrà donada, de la mateixa manera que el sensor de temperatura, en un valor entre 0 i 2'56 volts, ja que la tensió de referència s'ha configurat de la mateixa manera.

Si tenim en compte que la tensió per al nivell de bateria s'està mesurant en un divisor de tensió, amb 2 resistències del mateix valor, com s'observa a la figura 25:

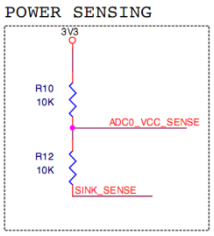


Figura 25. Divisor de tensió per a mesurar el nivell de tensió de la bateria.

El valor de tensió que mesurarà el convertidor A/D, aplicant la llei d'Ohm és:

$$Intensitat = \frac{V_{BATERIES}}{10K + 10K} = \frac{V_{BATERIES}}{20K}$$

$$V_{ADC0} = Intensitat \cdot 10K = \frac{V_{BATERIES}}{20K} \cdot 10K = \boxed{\frac{V_{BATERIES}}{2}}$$

El fet que els valors de les resistències siguin iguals, significa que la tensió mesurada serà la meitat del valor de la tensió general. Així doncs, quan es rep un valor decimal (entre 0 i 1023) des d'un node, tenim un valor de tensió equivalent a la meitat del que presenta la bateria.

Per a conèixer el valor en volts que presenta la bateria s'aplica la següent expressió:

$$Tensió\ de\ les\ bateries = \left(\frac{Valor\ A/D(entre\ 0\ i\ 1023)}{1024} \cdot 2'56Volts\right) \cdot 2$$

10.2. Compilació i execució

10.2.1 Compilació dels nodes

Un cop tenim el fitxer .ZIP descomprimit, tindrem una distribució de carpetes igual que la mostrada a la figura 26:

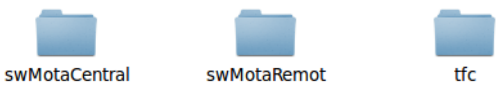


Figura 26. Distribució de carpetes del projecte.

Per a realitzar la compilació del node central s'ha d'obrir un terminal (combinació <CONTROL> + <ALT> + <t>) i navegar amb la comanda `cd` fins a la carpeta `swMotaCentral`. Un cop dins la carpeta, introduir la comanda

```
make cou24 install,1
```

Amb aquesta comanda, compilem el codi del node central i li assignem al node el id=1. El terminal mostrarà, després d'una estona, el següent aspecte:

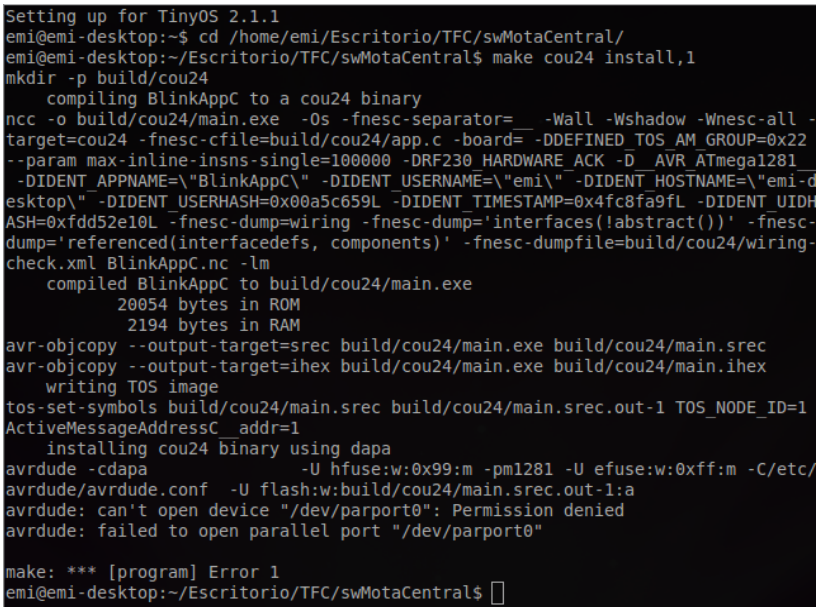


Figura 27. Compilació del codi del node central.

Encara que al final surt un error, no es referent a la creació del fitxer executable que s'emmagatzemarà en el node.

Per a carregar en el node el codi que s'acaba de compilar es fa servir la comanda següent:

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

On X serà el número de port USB on estigui connectat el node. Aquest número el podem saber mitjançant la comanda `motelist`, que ens mostra el llistat dels nodes que tenim connectats al PC i quins ports USB estan ocupats amb els nodes.

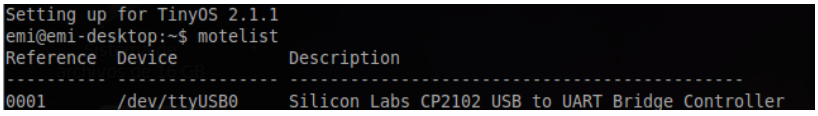


Figura 28. Comanda `motelist` que mostra un dispositiu connectat

Un cop introduïda la comanda `meshprog`, s'ha de polsar el polsador de "reset" del node que es vol programar i després d'una estona el codi ja estarà instal·lat al node.
Per a programar ara el node remot, hem de fer les següents comandes:

```
cd ..
cd swMotaRemot
make cou24 install,3
meshprog -t /dev/ttyUSBX -f ./build/cou24/main.srec.out-3
```

Amb la instal·lació dels nodes amb id=1 i id=3 , s'identifiquen els missatges que envia el node central i els que enviar el node remot, perquè l'origen de cada missatge vindrà especificat per aquest id. Aquest id dins el codi realitzat en els nodes, es pot veure amb la macro TOS_NODE_ID.

10.2.2 Compilació de l'aplicació central

Per a fer la compilació de l'aplicació central pot ser interessant tenir instal·lat també el entorn Qt per si es vol fer algun canvi en la interfície gràfica.
Amb l'entorn instal·lat, realitzar la compilació de l'aplicació central és més fàcil.

Per a instal·lar l'entorn Qt Creator dins el sistema operatiu Ubuntu 10.04 anem a "Centro de Software de Ubuntu":

Figura 29. Menú de sistema "Opció Centro de software de Ubuntu"

Dins de la finestra que s'ha obert, a la cantonada superior dreta, on hi ha el quadre per a realitzar cerques posem "Qt" i de seguida surt a l'esquerra *Qt Creator*, aleshores es fa una pulsació en *instal·lar* per a realitzar la instal·lació:

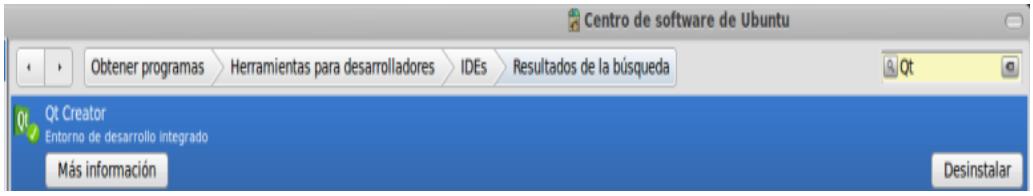


Figura 30. Finestra per a instal·lar Qt Creator

Un cop instal·lada l'aplicació *Qt Creator* i executada, anem al menú "`Archivo > Open File or Project`", un cop escollida aquesta opció ens sortirà un navegador per a seleccionar la carpeta on tenim el projecte creat.

En el nostre cas anem fins a la carpeta `/tfc` i dins de la carpeta seleccionem el fitxer que té extensió `.pro` .

Un cop carregat el projecte fent *<CONTROL>+ <r>* el projecte es compila i s'executa. Quan ja s'ha compilat, dins la carpeta `/tfc` tindrem el fitxer executable que es podrà executar des d'un terminal, introduint `./tfc`.

Una ultima indicació, per a que funcioni l'aplicació central, aquesta fa servir la llibreria C "`serialsource`", que es troba a `/opt/tinyos-2.1.1/support/sdk/c/sf/`.

Per a compilar-la, si no s'ha fet amb la instal·lació del TinyOS, s'han de seguir els següents passos:

```
sudo apt-get install autoconf
cd /opt/tinyos-2.1.1/support/sdk/c/sf
./bootstrap
./configure
make
```

Ara si, l'aplicació central ha de funcionar correctament.

10.3. Registre de successos

L'aplicació central des de que es iniciada fins que es tanca, va enregistrant cada succés que ocorre en el transcurs de l'execució de l'aplicació. Aquest registre es realitza creant en el inici del programa un fitxer, anomenat amb el dia i l'hora en que s'executa el programa, i té l'aspecte que mostra la figura 31:

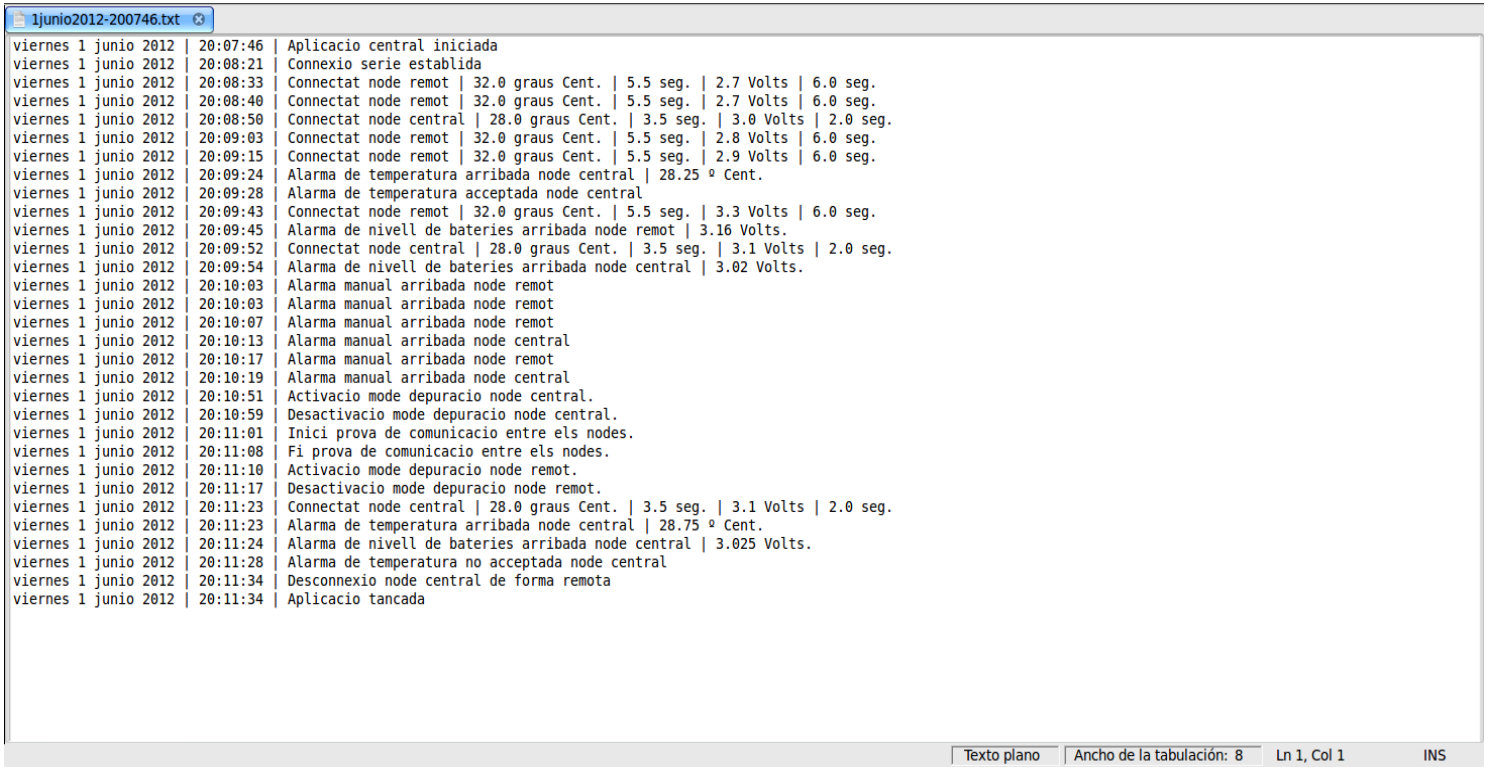


Figura 31. Fitxer de text amb el registre dels successos

La figura 31 mostra el registre que s'ha emmagatzemat de l'aplicació central i els 2 nodes, el dia 1 de Juny de 2012, iniciant l'aplicació central a les 20:07:46.En aquesta figura tenim el registre de:

- quan ha iniciat l'aplicació
- quan s'ha connectat un node
- quina configuració s'ha enviat al node en connectar-se
- quines alarmes han arribat
- quin valor presentava la temperatura o el nivell de bateria en arribar les alarmes
- les alarmes manuals
- l'activació/desactivació de la configuració de cobertura
- les proves de depuració
- les desconnexions de forma remota
- el tancament de l'aplicació

Aquests missatges sempre van acompanyats de la data i l'hora en que han ocorregut.

Els registres de successos (anomenats normalment logs), tenen la seva utilitat per a mantenir un històric d'incidències, i poder estudiar en alguns moments quins han estat els successos que han ocorregut en certs moments.