MEMORIA DEL TFC

Sistemes encastats

Alarma d'incendis

Autor: Carlos Salinas Naval
Enginyeria tècnica en Informàtica de Sistemes

Consultor: Jordi Bécares Ferrés

12 de Juny de 2012

Al meu futur i estimat fill Eric

i

a la meva dona i futura mare Sandra.

Agraïments

Voldria agrair el temps dedicat als consultors, tutors i companys, que m'han acompanyat al llarg de la meva aventura a la UOC, els vostres consells i comentaris han fet sense cap mena de dubte més lleugera la carrega.

No és fàcil però si necessari, en aquesta època tant desmotivadora que estem vivint, mantenir-se ferm en la creença de que l'educació és el pilar bàsic del nostre futur, per tant a part de donar les gràcies també voldria animar-vos a continuar amb la bona feina que esteu fent.

No menys important ha estat el necessari suport i els milers d'ànims que he rebut per part de la meva família i amics.

Resum

El següent document intentarà presentar d'una manera ordenada les motivacions, processos i resultats d'un projecte basat en l'àrea dels sistemes encastats i sensors remots.

El projecte en concret tracta de la realització d'una centraleta d'alarmes d'incendis. Per a aconseguir el seu objectiu, aquest sistema fa ús de la tecnologia distribuïda i de la comunicació sense fils. La idea principal es bassa en realitzar un sistema de detecció d'incendis a través d'un seguit de petits dispositius independents i autònoms, que treballaran en conjunció per a realitzar el control de seguretat d'un area més o menys gran.

La programació d'aquests petits dispositius anomenats en l'argot tècnic *motes*[1], s'ha realitzat sota la plataforma *TinyOs*[2] , aquest sistema operatiu ha estat dissenyat expressament per fer-se servir en dispositius remots. Per fer la programació s'ha fet servir el llenguatge de programació *nesC*[3].

Amb la programació de cada mota arribem a gestionar el control de la temperatura així com el nivell propi de bateria, així com una interactuació amb l'usuari local.

D'altre banda s'ha realitzat una aplicació en entorn gràfic per a facilitar el control de totes les motes que composen el sistema total. Per a realitzar aquesta aplicació s'ha fet servir programació java sota la plataforma Linux.

1. ***motes:*** *Sistema encastat destinat al seu ús distribuït bàsicament compost de un µC i els seus perifèrics on podem trobar com a component destacat un sistema de comunicacions sense fils, així com una sèrie de sensors.*

2. ***TinyOs:*** *Sistema Operatiu amb filosofia de programari lliure, basat en l'ús de components i la seva interconnexió, destinat al govern de sensors remots.*

3. ***nesC:*** *(Network embedded System C) llenguatge de programació bassat en C, dissenyat especialment per a treballar amb xarxes de sensors remots.*

Índex de continguts

Índex de figures

Índex de Taules

1. Introducció

L'automatització dels edificis o *domòtica*[1] s'ha estès àmpliament en el transcurs de les ultimes dècades. Edificis corporatius, hotels, centres d'art o d'altre edificis de gran volum i ús públic han trobat en aquesta tècnica, la domòtica, un sistema per a reduir costos energètics, afavorir el confort dels ocupants i gestionar els sistemes de seguretat.

Treballant en aquest terreny el projecte en qüestió proporciona al possible usuari, una aplicació completa que resol la necessitat del control d'incendis en un edifici, per aconseguir això basarem l'estudi en un sistema de sensors distribuïts intercomunicats.

El resultat final del projecte és un producte, que a falta de petites modificacions que comentarem més endavant, pot ser comercialitzat d'una manera pràctica sense cap mena de dubte, amb un alt grau de satisfacció pel que fa al compliment dels seus objectius.

L'abast per tant del projecte engloba només la part de programació dels sistemes, ja que l'estudi de necessitats així com el disseny del hardware ha estat realitzat en projectes anteriors. No entra tampoc en l'abast del projecte que tractem la part de la comercialització i el marketing, que seria necessari per a poder parlar d'un producte final.

1. ***domòtica:*** *Tecnologia bassada en sistemes electrònics dissenyats per al seu ús en edificis ja siguin industrials o de vivendes que al ser instal·lats aporten un augment del confort, estalvi energètic i la seguretat dels inquilins o usuaris.*

1.1 Justificació

Malgrat els seus beneficis els sistemes domàtics clàssics incorporen un cost afegit, gran part del qual és degut a la seva instal·lació. Aquest cost d'instal·lació fàcilment pot ser computable des de un punt de vista econòmic, ja que cadascun dels dispositius que la formen el sistema ja siguin sensors o actuadors havien d'anar connectats mitjançant un cablejat el preu del qual per certes distancies podia ser bastant elevat.

Per afegit i des de un punt de vista estètic això implicava un gran esforç per a amagar tot aquest cablejat o penalitzar el valor estètic de l'edifici amb la intenció de minimitzar costos.

Així dons podríem reduir aquest cost econòmic i estètic si eliminéssim el cablejat , i és per això que els sistemes autònoms i amb comunicacions sense fils estan guanyat cada dia més terreny dins de la domòtica. Per tant és lògic el fet de voler fer el disseny complert d'un sistema de detecció d'incendis basat en aquestes tecnologies.

1.2 Descripció

Com ja s'ha dit el sistema basat en sistemes encastats, petits sistemes autònoms que treballen coordinats per un sistema central que s'executa en una plataforma PC. Les comunicacions entre aquesta aplicació i els diferents sistemes remots és farà per sistemes sense fils.

Els requisits bàsics del projecte dons, serà a partir de un hardware ja dissenyat i seleccionat, programar aquests dispositius per a aconseguir el compliment dels objectius establerts per uns clients amb la intenció de cobrir les necessitats en la gestió de seguretat en front els incendis en edificis.

1.3 Objectius

Els objectius principals imposats pels usuaris, és resumeixen aquí en cinc punts per clarificar els resultat global desitjat, el conjunt complet de objectius és pot trobar en la part d'annexos del document.

- Dotar al sistema o conjunt de sistemes de les eines necessàries per a realitzar el control de temperatura en els diferents punts de la instal·lació, edifici del qual és vulgui gestionar la seguretat.

- Controlar els nivells de bateria dels diferents circuits autònoms.

- Desenvolupar tot el protocol de comunicació per tal d'assegurar les correctes comunicacions entre els diferents aparells del sistema.

- Proporcionar al sistema global una certa robustesa en front a fallades, que li permeti complir els anteriors.

- Dissenyar un sistema *HMI*[1] que permeti el control de tot el sistema des de un punt centralitzat.

Com s'ha dit hem vist un resum dels objectius detallats del projecte, que podem trobar en la secció d'annexos.

*1. **HMI:** Sigles que provenen de l'anglès "human machine interface" i signifiquen sistema de comunicació entre el usuari i la màquina o aplicació.*

1.4 Enfocament i mètode seguit

Podem dividir el projecte en tres grans parts o seccions, cadascuna d'aquestes podria haver estat desenvolupada d'una manera separada però per tema de temps i practicitat s'ha decidit desenvolupar d'una manera paral·lela.

PROGRAMACIÓ DE LES MOTES

Sense cap mena de dubte aquesta és la part més característica del projecte i també és podria dir la més important. És tracte del disseny del programa que s'executarà al sistemes distribuïts.

Com s'ha dit en aparts anteriors aquesta part del projecte ha estat desenvolupada amb el llenguatge de programació nesC, el motiu de aquesta elecció ve donat per la pròpia naturalesa d'aquest sistema de programació especialment dissenyat per tal de treballar amb dispositius encastats. Permeten així l'ús de llibreries pròpies del llenguatge ja desenvolupades i funcionals que han facilitat la realització del programa.

Primer de tot s'ha destinat un cert temps en la preparació de l'entorn de treball, cosa que facilità el correcte desenvolupament de les diverses probes realitzades i l'obtenció del codi destinat a les motes.

Inicialment i com que és disposava de dues motes, és va decidir que una d'elles treballes només com enllaç amb el PC i l'altre fes la funció de sensor remot.

Un cop finalitzades les probes i el codi de control de les motes, estava previst fer la unió dels dos codis el de control i el de comunicació per tal que la mota central també fes les tasques de sensor.

Aquest plantejament inicial que semblava correcte ha devingut a la fi un problema ja que no s'ha pogut fer la unió d'una manera correcta i la mota que fa de pont entre la plataforma PC i la resta de motes només fa aquesta feina.

Per a realitzar el codi final i degut als coneixements inicials dels sistemes encastats és va optar per un estudi inicial seguit de la realització de petits programes destinats a cobrir cadascun dels objectius per separat.

Cap al final del projecte s'han cohesionat i ajustat els diversos fragments obtenint un tot que ha donat com a resultat el codi final destinat a la programació del sistema.

Aquesta manera de treballar ha permès obtenir resultats desitjats inicialment a excepció del ja comentat pel que fa la mota connectada al PC. D'altre banda ha permès realitzar un sistema de probes bastant senzill ja que moltes de les funcionalitats estaven ja provades abans de ser associades al codi final i per tant només calia comprovar el correcte funcionament un cop unides.

APLICACIÓ PC

Al principi és va decidir realitzar aquesta aplicació amb plataforma java, i també que havia de ser gràfica. El motiu de fer-la gràfica ha estat principalment un tema de nivell estètic, però també de nivell funcional ja que permetia una millor i més fluida comunicació entre l'usuari i el sistema.

Per a desenvolupar correctament l'aplicació s'ha anat fent en paral·lel al codi de les motes, donant suport a les funcionalitats un cop desenvolupades. Per tant podríem dir que aquesta part a anat a remolc de la programació de les motes.

Aquesta manera de treballar ha permès tenir, en tot moment, un gran control del que s'estava desenvolupant, però ha donat un resultat molt irregular ja que de vegades al no haver-hi un plantejament global des de un principi ha fet que l'evolució hagi estat per trams.

MEMORIA O DOCUMENTACIÓ

En aquest cas inicialment és plantejar realitzar la composició de la memòria d'una manera paral·lela a les altres dues tasques, però degut a la dedicació de temps i recursos a aquestes es modificar l'idea inicial per la que al final s'ha dut a terme.

Des del inici, s'ha anat enregistrant tots els avanços, proves, cerques de informació i modificacions en un registre que ha permès fer la composició de la documentació del projecte d'una manera ràpida i senzilla.

1.5 Planificació

Per presentar la planificació farem servir dos *diagrames de Gantt*[1], un amb la planificació temporal inicial i l'altre amb la temporització final, també s'indicaran els motius que han dut a les no correspondències de la realitat amb el pressupostat en un inici.

Per facilitat la comprensió dels dos diagrames, s'ha dividit per colors les tres parts diferenciades, i comentades en l'apartat anterior, en que s'ha desenvolupat el projecte.

Amb un estel vermell s'han volgut destacar i indicar les dates finals dels diferents períodes de lliurament del projecte.

A la figura 1.5.a podem veure el diagrama de Gantt realitzat en les primeres setmanes del projecte. D'altre banda a la figura 1.5.b podem veure l'evolució real del projecte.

*1. **Diagrama de Gantt**: és tracta d'una eina per a representar de forma visual i clara la temporització de les diferents tasques que composen una activitat i la relació que hi ha entre elles.*

PLANIFICACIÓ INICIAL

	Nombre de tarea	Comienzo	Fin
1	**Recerca d'informació, sobre les motes.**	**lun 27/02/12**	**jue 15/03/12**
2	Sistema operatiu TinyOs.	lun 27/02/12	jue 15/03/12
3	Anàlisis dels components incorporats.alanilisi incorporat.	lun 27/02/12	mar 06/03/12
4	Mètodes de programació.	lun 27/02/12	jue 15/03/12
5	Planificació del TFC.	mar 13/03/12	lun 19/03/12
6	Instal·lació de software i preparació de l'entorn de treball.	jue 01/03/12	mié 21/03/12
7	Realitzar un esbós del projecte.	mar 28/02/12	mié 07/03/12
8	Realitzar l'aplicació per a PC.	jue 29/03/12	mié 09/05/12
9	Primera connexió amb les motes des del PC.	lun 26/03/12	jue 29/03/12
10	Realització d'un petit programa de control dels polsadors i indicadors.	vie 30/03/12	jue 05/04/12
11	**Proves dels sensors i adquisició de dades.**	**vie 06/04/12**	**mié 11/04/12**
12	Programa de control sensor de llum.	vie 06/04/12	mié 11/04/12
13	Programa de control sensor de temperatura.	vie 06/04/12	mié 11/04/12
14	Programa de control del sensor de magnetisme.	vie 06/04/12	mié 11/04/12
15	Realitzar un programa que controli el nivell d'energia.	jue 12/04/12	mar 17/04/12
16	PAC 2 Lliurament inicial del codi	mar 24/04/12	mar 24/04/12
17	Realitzar el programa final que cohesioni totes les parts.	mié 18/04/12	mar 08/05/12
18	Realitzar proves de funcionament.	jue 10/05/12	mar 15/05/12
19	Des. Sist. d'instal·lació de l'aplicació, així com un manual d'usuari.	mié 16/05/12	lun 21/05/12
20	PAC 3 Lliurament del codi	mar 22/05/12	mar 22/05/12
21	Confecció de la memòria del TFC.	mar 10/04/12	mar 12/06/12
22	Presentació del TFC.	jue 07/06/12	jue 14/06/12

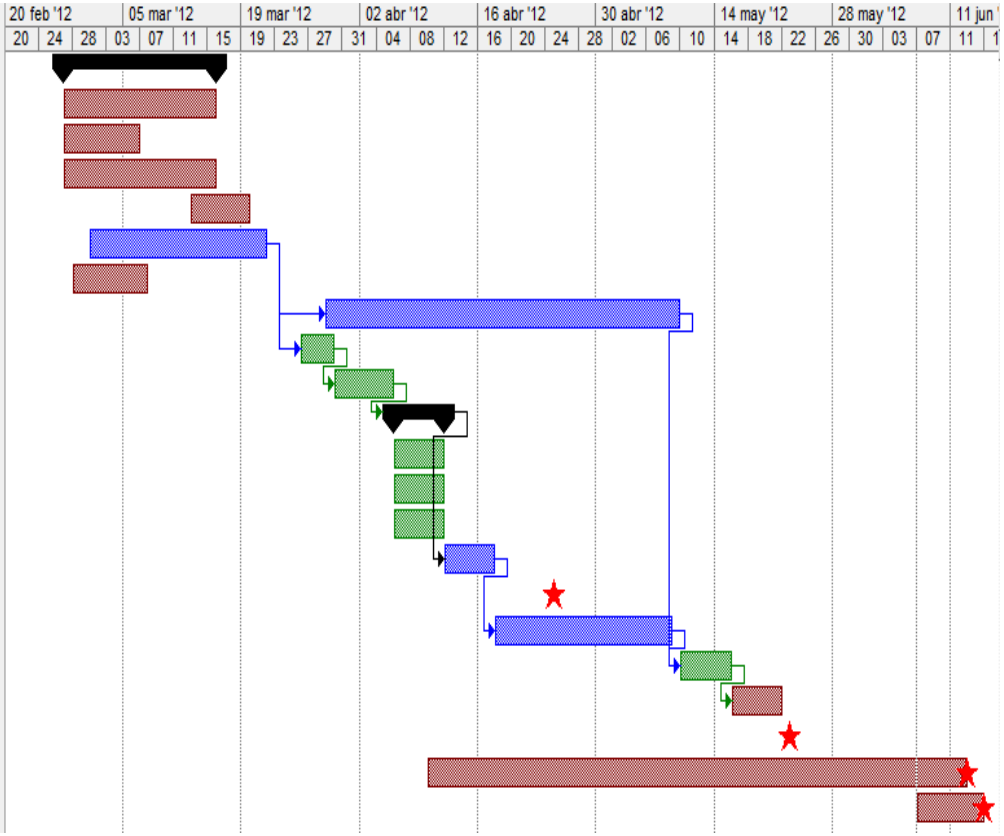

Figura 1.5.a: Planificació temporal inicial

TEMPORITZACIÓ REAL

	Nombre de tarea	Comienzo	Fin
1	**Recerca d'informació, sobre les motes.**	**lun 27/02/12**	**jue 15/03/12**
2	Sistema operatiu TinyOs.	lun 27/02/12	jue 15/03/12
3	Anàlisis dels components incorporats.alanilisi incorporat.	lun 27/02/12	mar 06/03/12
4	Mètodes de programació.	lun 27/02/12	jue 15/03/12
5	Planificació del TFC.	mar 13/03/12	lun 19/03/12
6	Instal•lació de software i preparació de l'entorn de treball.	jue 01/03/12	mar 27/03/12
7	Realitzar un esbós del projecte.	mar 28/02/12	mié 07/03/12
8	Realitzar l'aplicació per a PC.	jue 29/03/12	mié 23/05/12
9	Primera connexió amb les motes des del PC.	mié 28/03/12	jue 05/04/12
10	Realització d'un petit programa de control dels polsadors i indicadors.	vie 06/04/12	mié 11/04/12
11	TEMPS MORT	jue 12/04/12	mié 18/04/12
12	**Proves dels sensors i adquisició de dades.**	**jue 19/04/12**	**jue 26/04/12**
13	Programa de control sensor de temperatura.	jue 19/04/12	jue 26/04/12
14	Programa de control del sensor de magnetisme.	jue 19/04/12	jue 26/04/12
15	Realitzar un programa que controli el nivell d'energia.	vie 27/04/12	mié 02/05/12
16	**PAC 2 Lliurament inicial del codi**	**jue 26/04/12**	**jue 26/04/12**
17	Realitzar el programa final que cohesioni totes les parts.	jue 03/05/12	mar 22/05/12
18	Realitzar proves de funcionament.	jue 24/05/12	jue 24/05/12
19	Des. Sist. d'instal•lació de l'aplicació, així com un manual d'usuari.	vie 25/05/12	vie 25/05/12
20	**PAC 3 Lliurament del codi**	**vie 25/05/12**	**vie 25/05/12**
21	Captació de informació per a realitzar la memoria	jue 01/03/12	vie 25/05/12
22	Confecció de la memòria del TFC.	lun 14/05/12	lun 28/05/12
23	TEMPS MORT	mar 29/05/12	vie 01/06/12
24	Confecció de la memòria del TFC.	lun 04/06/12	mar 12/06/12
25	Presentació del TFC.	lun 11/06/12	jue 14/06/12

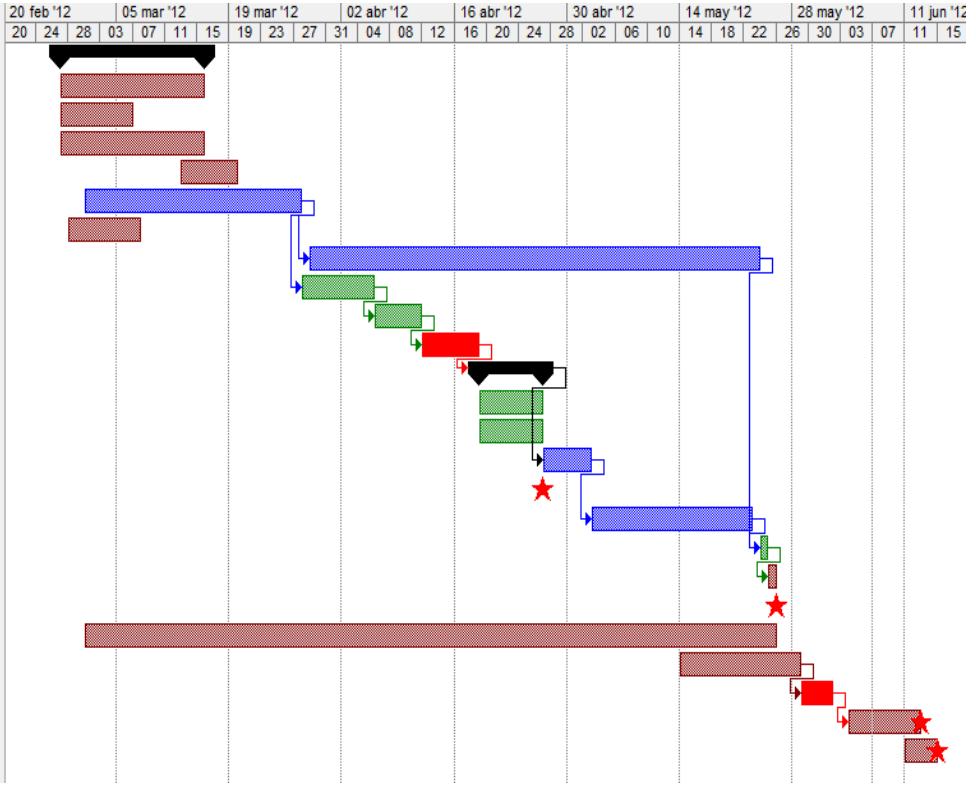

Figura 1.5.b: Temporització real

Com és pot veure a les dues imatges, les variacions més significatives han estat en dos moments bastant localitzats, el primer en ple període de producció del programari tant de l'aplicació de la mota com de la del PC. S'ha indicat a la planificació com a una tasca anomenada TEMPS MORT.

El motiu d'aquest retràs va ser una reducció dràstica del temps disponible per diferents motius gairebé tots relacionats amb d'altres projectes no previstos en el moment de la planificació inicial. Aquesta reducció tot i que va afectar a quasi tot el període del projecte, va ser crítica en la setmana indicada, ja que va aturar per complert el procés del projecte.

La segona gran modificació, semblant a la primera per aquest cop per un viatge d'indole laboral que si no ha aturat la redacció de la documentació si que l'ha enrederit considerablement. També el veiem indicat com a TEMPS MORT.

Totes les altres modificacions han estat degudes a la falta de coneixements en el moment de la planificació inicial. S'ha minimitzat l'efecte en la temporització amb la dedicació d'hores extres no previstes en molts moments del període de realització del projecte.

Hi ha hagut d'altres modificacions de menor entitat, provocats en la seva totalitat per petits problemes en la generació de codi, si ens fixem veurem un allargament del temps de les tasques inicials i un escurçament de les finals això reflexa l'adquisició pausada dels coneixements referents a la programació dels sistemes encastats.

El resultat dels petits desajustos, ha esdevingut una retallada sobre tot en el període de proves, tot i com ja s'ha comentat la pròpia estratègia de disseny ha fet que el temps dedicat a proves no hagi hagut de ser tant com al principi és va preveure.

Recursos emprats

MAQUINARI

Ha estat necessari per a la realització del projecte l'ús de dues motes model cou24, com les de la imatge següent:

Figura 1.6.a: Detall de la mota feta servir en el projecte.

Es pot observar a la mota una petita modificació respecte a les originals aquesta modificació ha consistit en incorporar un petit interruptor per a desconnectar la mota de les bateries que l'alimenten i per tant a part d'estalviar energia quan no estigui en ús, permetre realitzar simulacions de talls d'energia fàcilment.

A part de les motes ha estat necessari un ordinador domèstic per tal de connectar-hi una de les motes i executar l'aplicació de control.

PROGRAMARI

El projecte s'ha realitzat completament fent servir programari lliure, a excepció de la documentació que s'ha realitzat amb el paquet office de Microsoft.

Els sistema operatiu de les motes ve establert pel client i es tracta, com ja s'ha comentat de TinyOs, un sistema operatiu de codi obert i lliure distribució dissenyat per un consorci d'entitats entre les que destaquen la Universitat de California a Berkley i Intel Research. TinyOs ha estat desenvolupat amb l'objectiu de treballar mitjançant components per al control de sensors inalambrics.

Per a realitzar l'edició dels programes de les motes, s'ha fet servir l'IDE Eclipse Galileu, al qual se li han afegit les llibreries de TinyOS.

Alhora de fer la descarrega del programa a les motes, s'ha utilitzat l'aplicació *meshprog*[1], fent servir per tal propòsit el port USB de l'ordinador. La crida d'aquesta aplicació s'ha realitzat des de la shell de Linux, tot i que també s'hagués pogut realitzar directament des de Eclipse.

També s'ha fet servir d'altres aplicacions que proporciona TinyOs, per tal de fer la comprovació i gestió de les comunicacions, una de molt important és la funcionalitat serialForwarder que fa la gestió dels enviaments i recepcions de paquets entre les aplicacions de PC i el port USB.

Per a realitzar el programa de l'aplicació gràfica del PC, on es farà el control general, s'ha fet servir el IDE de programacació NETBEAMS IDE 6.8. Degut a la facilitat que dona per treballar amb entorns de finestres.

---

*1. **meshprog**: programa de transferència de dades a través del port USB que pot fer servir diferents tipus de protocols de transmissió de dades via sèrie, com poden ser (JTAG, ISP,...)*

1.7 Productes obtinguts

AM_Basic: Com a resultat del projecte s'han obtingut dos elements, per una part tenim un codi per a programar les motes que faran de sensors distribuïts. El sistema ha estat dissenyat per un volum màxim de 20 components.

BaseStation: Una dels 20 motes, la que anirà connectada al PC, serà especial i serà programada amb un codi proporcionat per TinyOs anomenat BaseStation, aquest codi fa de pont entre els altres sensors i el port USB del PC. La seva funció és transformar les comunicacions de radio freqüència a dades series i a l'inversa.

Alarmes: Per últim també es fa entrega de l'aplicació que serà executada al ordinador així com un petit manual d'instruccions.

Amb els productes obtinguts podríem realitzar fàcilment una instal·lació real, amb poques modificacions.

1.8 Descripció d'altres capítols

En els capítols següents es farà un anàlisis més detallat de les tecnologies, processos, proves i funcionalitats resultants.

Antecedents

2.1 Estat de l'art

Quan parlem de tecnologia encastada estem parlant de petits dispositius que combinen tecnologies electròniques i informàtiques. És basen en la utilització de processadors de mida petita amb una perifèria bàsica, tot i que no estan dissenyats a mida, la seva configuració sol ser molt ajustada a les necessitats.

HISTORIA

Si s'hagués de buscar un inici en els sistemes encastats, aquest aniria sense dubte lligat als micro-Controladors (µC) dispositius que englobaven un processador, la memòria, ports de entrades, ports de sortides i normalment encara que no sempre sistemes de comunicacions.

Potser els primers de tots, si més no els primers que és van arribar a estendre àmpliament van ser el 8031 d'Intel i el 6800 de Motorola en la dècada dels anys setanta, l'aparició d'aquests dispositius va ser ja a les hores una petita revolució tecnològica a nivell industrial.

Aquests sistemes normalment treballaven individualment, ja que el seu disseny estava pensat per al control de petites màquines o electrodomèstics. No va ser fins a entrats els anys noranta que amb la implantació generalitzada de les busos i xarxes de comunicacions industrial , quan és va començar a plantejar-me l'ús d'aquests elements d'una manera distribuïda.

Un exemple seria el ús de sistemes encastats en l'automoció, amb l'aparició de sistemes distribuïts connectats amb bus CAN desenvolupat per la marca alamana BOSH i aplicat en un principi a Mercedes Benz. Van minimitzar el cablejat intern dels automòbils, amb les avantatges de pes i cost que això implica.

El salt definitiu arriba amb l'aparició de les comunicacions sense fils, gracies a aquest fet, els sistemes poden ser autònoms i mantenir-se en contacte amb d'altres d'una manera eficient.

ESTAT ACTUAL

Actualment les motes o sistemes encastats poden treballar en xarxes de sensors inalambrics, que poden ser distribuïts arreu del món gracies a la tecnologia TCP/IP.

D'altre banda podem trobar sistemes encastats que treballen amb sistemes operatius a temps real, que faciliten un control més fiable d'algunes aplicacions.

Les seves avantatges principals respecte a altres tecnologies, són varies:

- La seva mida que cada cop es redueix més, permet instal·lar-les en multitud d'equips.
- Cost d'instal·lació, al ser autònoms no se'ls ha de connectar físicament, i per tant ens estalviem el cost del cable i la mà d'obra de la instal·lació.
- També a conseqüència dels dos anteriors podem minimitzar el consum energètic tant important en l'actualitat.
- Versatibilitat de projectes gracies al seu potencial i el seu disseny molts cops dedicats.
- Establiment, podem fer la instal·lació en llocs on no hi ha un fàcil accés com pot ser entorns rurals o paratges naturals per fer qualsevol control.

2.2 Estudi de mercat.

SISTEMES ACTUALS PER A LA PRODUCCIÓ DE DISPOSITIUS

Al mercat, a part del dispositiu fet servir en el projecte, podem trobar multitud de fabricants de plataformes similars on queda obert el disseny per complementar segons les necessitats de cada projecte.

Un exemple de gran renom en l'actualitat és Arduino veurem a la següent imatge una placa d'aquest fabricant en concret el model Duemilanove, veiem que el format de la placa recorda el esposat en el capítol anterior. Lògicament amb diferents perifèrics.

Figura 2.2.a: Sistema encastat Arduino

Hi ha motes d'altres fabricants com poden ser Crossbow, Ember, Intel, Sun , Sentlla o Mircrosystems també han desenvolupat una gran varietat de plataformes.

Gracies a aquesta gran varietat i la competència ha fet que el cost s'hagi reduït i les prestacions hagin augmentat, amb el que la seva utilització s'ha estès molt. En l'actualitat aquests sistemes són realment tant potents com ordinadors personals i poden donar gairebé totes les seves funcionalitats.

UTILITZACIÓ D'AQUESTA TECNOLOGIA AL MERCAT.

Gracies a les avantatges, ja comentades, d'aquests sistemes. La seva implantació tant a nivell domèstic com industrial, s'està estenen ràpidament en els últims temps i la seva utilització arriba a sectors que en un principi podien no semblar destinats a rebre aquesta tecnologia.

Com a exemples de sectors on s'ha implantat la seva utilització podríem tenir:

- Sistema d'alarmes d'intrusió o seguretat d'edificis.
- Domòtica
- Meteorologia, realitzant petites centraletes que poden ser fàcilment instal·lables en llocs de difícil accés.
- Automoció.
- Robòtica.
- Industria.
- Electrònica de consum.

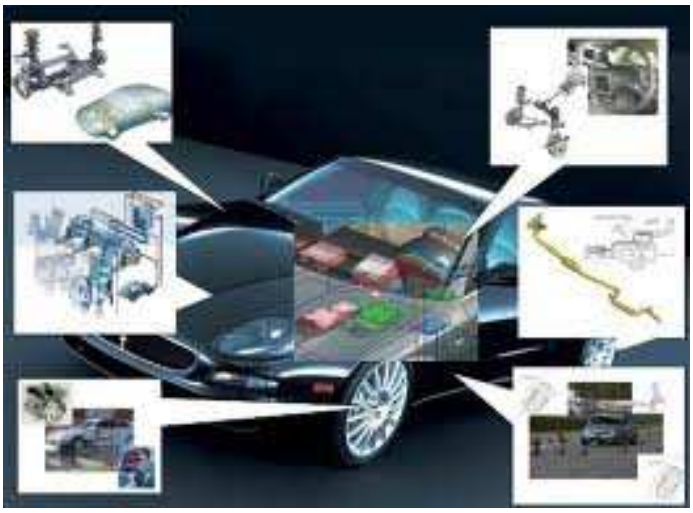

Figura 2.2.b: Utilització dels sistemes encastats a l'automoció.

Descripció Funcional

Per explicar la funcionalitat del sistema, ho farem seguint el fil de tot el document, és a dir dividirem el sistema total en dues parts diferenciades, però primer de tot farem un anàlisis del conjunt.

3.1 Sistema complet

Com ja s'ha comentat el sistema funcionarà d'una manera distribuïda tot i que una de les motes ha d'estar unida al PC, que estarà programada amb el programa **BaseStation** i ha de tenir identificador numèric igual a 1, les altres treballaran independentment i aniran enviant i reben informacions via comunicacions sense fils.

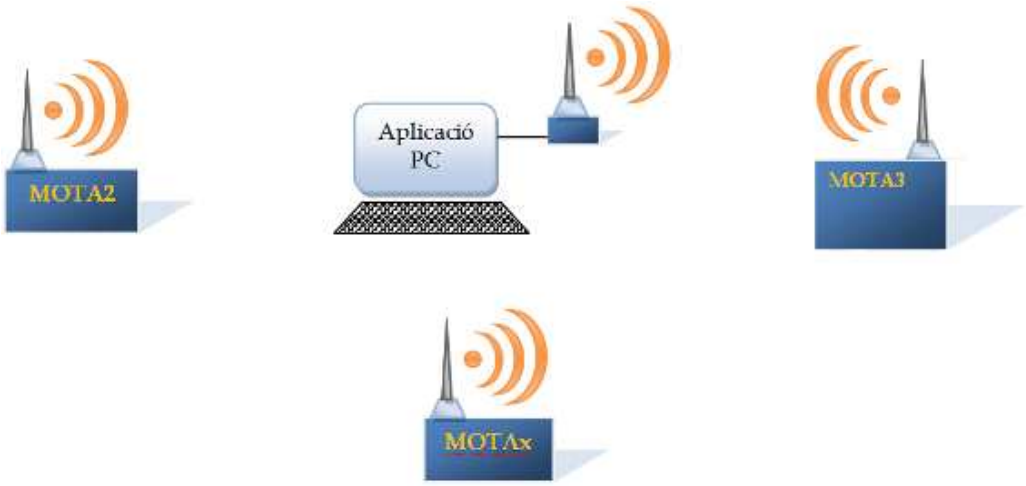

Figura 3.1.a: Distribució de les motes en el sistema global

Totes les motes tindran un identificador numèric, que haurà d'anar en aquest sistema de la mota 2 a la mota 20, ja que ha estat dissenyat per un nombre màxim de sistemes distribuïts.

COM ÉS COMUNIQUEN ENTRE ELLS

Ja sabem que la tecnologia emprada en la comunicació és sense fils, però quan i perquè és comuniquen. Cada component és independent i envia les comunicacions quan ho necessita, en un principi no hi ha un sistema central com pot semblar a simple vista.

Hi ha dos tipus de comunicacions, en la primera de caràcter periòdic, els sensors envien les dades cada cert temps, aquest temps és configurable. Aquestes comunicacions no disposen de comprovació d'entrega, ja que al ser periòdica no és crític que és perdi algun missatge.

El segon format de comunicació és fa servir per les comunicacions no periòdiques, en aquest cas el component ja sigui el PC o la mota que vol comunicar algun fet, envia un missatge i espera la resposta de confirmació i de vegades, fins i tot envia un tercer missatge de finalització de la comunicació, el motiu és que aquests tipus de missatges engloben tant els de control com els de seguretat (Alarmes) i per tant en aquest cas si que hem d'assegurar la correcta recepció.

JUSTIFICACIONS DEL DISSENY

Sobre tot s'han d'aclarir dos conceptes, el primer és el motiu de perquè una de les motes ha d'estar programada amb l'aplicació **BaseStation** en comptes de fer també de sensor i així poder obtindre les dades de la posició central com seria desitjable, la raó és senzilla hi ha estat un motiu de no poder unificar les dues funcionalitats, encara que és del tot possible. Aquí dons tenim un punt de millora per un futur.

L'altre aspecte seria el fet de que el numero de sensors sigui limitat, en un principi seria millor que fos il·limitat, però això no és va tindre en compte al inici del projecte per una mala previsió, això a derivat en un error que d'altre banda no és senzill de modificar per l'estructura interna de l'aplicació PC, per tant si és vol modificar aquest factor, cosa interessant d'altre banda, el més lògic és torna a fer gran part de l'aplicació.

3.2 Descripció de l'aplicació de PC

L'aplicació que s'executarà al PC ha estat dissenyada en la seva integritat amb llenguatge *java* fent servir la filosofia de la programació orientada a objectes.

El fet de treballar amb objectes fa que el codi final sigui molt més entenedor, aquesta a estat una de les raons per treballar en aquest tipus de llenguatge, d'altre banda també es volia aprofitar el potencial respecte a la gestió de gràfics. Aquí tenim una figura on és mostra d'una manera molt bàsica les relacions entre aquestes classes.

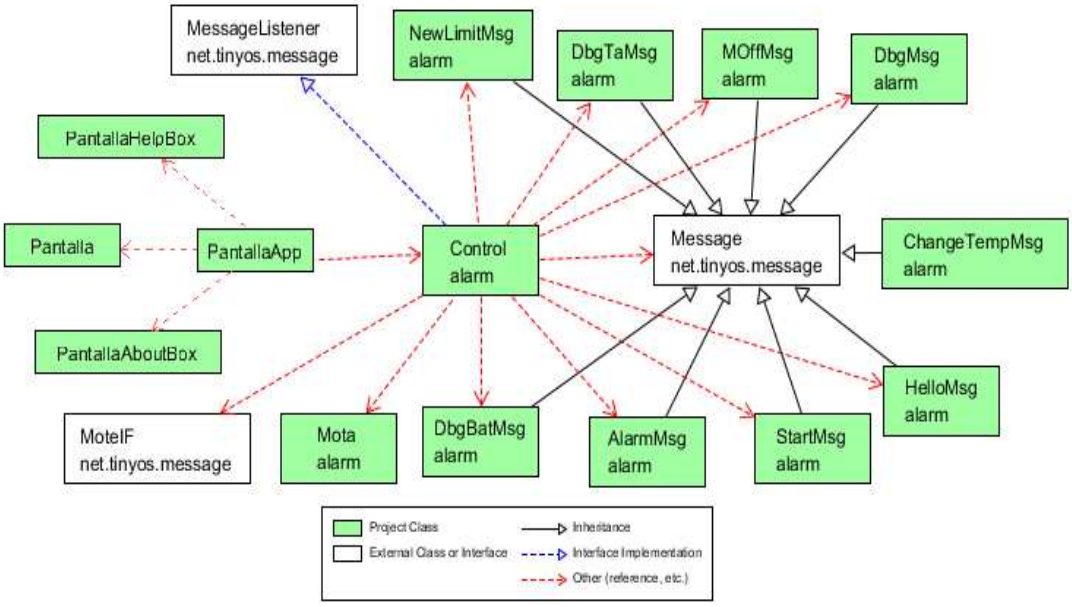

Figura 3.2.a: Diagrama de classes de l'aplicació per PC

Com és pot veure s'ha generat una classe per cada tipus de missatges que podem comunicar o rebre de les motes. La motivació ve donada perquè d'aquesta manera podem realitzar un control més acurat de cadascun dels diferents missatges, ajustant en cada cas els camps necessaris. S'hagués pogut crear un únic tipus de missatge que englobes tota la informació, això simplificaria la programació però faria més ineficient la comunicació ja que els paquets serien més llargs.

D'altre banda també s'ha generat una classe especifica per a mantenir la informació de cada mota, en un principi això és necessari ja que no totes les motes poden trobar-se en el mateix estat, una per exemple pot estar activada i l'altre connectada al sistema però sense activar.

Gracies a aquesta informació que serà guardada podrem en cas de que la mota passi per una seqüència de reactivació, per exemple per un reset o una caiguda de tensió, enviar-li l'estat en que es trobava abans de reiniciar-se.

D'altre banda tenim tot el control separat de la part gràfica ja que d'aquesta manera el control com veurem més endavant pot treballar d'una manera més eficient.

La part de l'aplicació que controla la iteració amb l'usuari *Pantalla* té dues parts diferenciades, tot el que s'escriu com informació visual, ja sigui mitjançant el Log o els camps més específics i d'altre banda tot el que l'usuari introdueix com a consignes del sistema.

Aquestes consignes poden modificar la configuració de una o varies motes a l'hora, la lectura de les comandes és realitza per un sistema d'esdeveniments, mentre que per tal de mantenir la informació actualitzada és realitza un refrescament periòdic.

3.3 Descripció de la mota

La descripció és només de les motes remotes, les que fan de sensors, ja que l'altre la que fa de pont de comunicació està programada amb un codi proporcionat per TinyOS anomenat *BaseStation* com ja s'ha comentat, més endavant analitzarem el seu funcionament.

La programació de les motes si be es cert que s'ha realitzat d'una manera modular, aquest factor no s'ha realitzat d'una manera optima.

Per realitzar el disseny les funcionalitats s'han separat en quatre mòduls en funció de la tasca que desenvolupen dins del disseny global. El diagrama de blocs quedaria així.

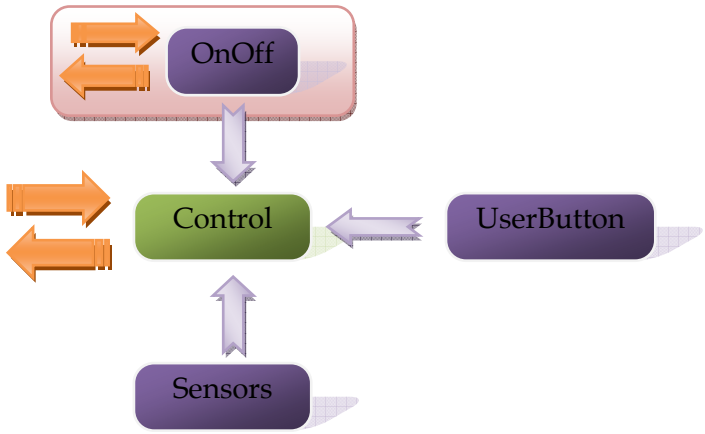

Figura 3.3.a: Diagrama de mòduls de la mota

A la figura anterior veiem que tenim 3 mòduls, OnOff, Sensors i UserButton. Aquests serien els mòduls de captació de informació la manera de treballar seria amb una filosofia mestre/esclau on el que faria la funció de mestre seria el mòdul central de control.

OnOff: En aquest mòdul controlem tot el referent a l'activació i desactivació magnètica de la mota, cada cert temps serà consultat pel mòdul de Control per veure el si hi ha hagut algun canvi en l'estat de la mota. Té també la missió de comunicar a l'aplicació PC els canvis d'estat.

Sensors: Aquest altre mòdul és l'encarregat de la lectura dels sensors de temperatura i bateria. La manera de comunicar aquesta informació al mòdul central de control, serà periòdica i com en el cas anterior serà el mòdul central que faci la demanda d'informació.

UserButton: És l'encarregat de fer la gestió del botó d'usuari de que disposa la placa de la mota, amb les seves pulsacions farem diferents accions en funció de l'estat en que es trobi la mota.

Control: Mòdul central i que porta gairebé tota la carrega de l'aplicació. És el mòdul on és prenen les decisions d'activació de les alarmes, així com on és fa el tractament de les comunicacions, excepte la ja comentada del mòdul OnOff, amb altres elements dels sistema global, i on és gestiona el control de flux del programa de la mota.

És aquest mòdul el que fa la gestió també dels indicadors lluminósos el significat dels quals s'explicarà més endavant.

JUSTIFICACIONS DEL DISSENY

Quan parlem de que la comunicació o lectura de dades entre mòduls es fa d'una manera periòdica, hagués estat molt més eficient unes comunicacions sota canvis simulant un sistema d'interrupcions però això complicava molt la programació i al no ser gaire el moviment de dades s'ha decidit fer-ho així.

En aquest cas s'ha desenvolupat el sistema en mòduls degut a que clarifiquen l'enteniment del sistema global. De totes maneres seguint aquest raonament s'hagués pogut modularitzar molt més, per exemple amb un mòdul de comunicacions un altre de gestió d'alarmes i deixant el control només per la gestió del flux de programa. El motiu de no fer-ho, ha estat que el mòdul de control necessitaria per la presa de decisions molta informació dels altres suposats mòduls i per tant hi hauria una gran utilització del processador.

Descripció Detallada

4.1 Breu explicació de l'aplicació BaseStation

Primer de tot analitzarem les funcionalitats que aporta, el codi emprat per a realitzar el pont de comunicació entre el port USB i les motes que fan la funció de sensors remots.

La mota programada amb aquest codi, haurà d'anar configurada amb el identificador numèric 1 ja que serà a aquesta adreça on la resta de motes faran els enviaments dirigits a l'aplicació de control.

Recordar també que per tal de realitzar la comunicació és imprescindible que l'aplicació *SerialForwarder* inclosa també a les llibreries de TinyOS, estigui funcionant per tal que l'aplicació PC rebi i enviï els missatges. En el manual d'usuari s'explica com fer l'execució d'aquesta aplicació.

Un cop programada la mota i connectada un port sèrie de l'ordinador, aquesta mitjançant els seus indicadors lluminosos que són tres mostrar el funcionament correcte o no de les transmissions, aquest indicadors seguiran la següent taula.

LED	Estat	Significat
Vermell	Commutació	Enviament d'un missatge des del USB a RF
Groc	Commutació	Enviament d'un missatge des de RF al USB
Verd	Commutació	Missatge perdut

Taula 4.1.a: Indicacions de l'aplicació BaseStation

4.2 Explicació de AM_Basic, aplicació de sensors remots

El llenguatge de programació nesC en el que s'ha desenvolupat el codi de les motes, a part de estar orientat a la gestió d'esdeveniments, també com ja s'ha comentat permet per treballar, en un sistema de components o modular per tant anem a explicar detalladament cadascun dels mòduls realitzat.

Cada mòdul consta de dues parts, la primera normalment diferenciada perquè el nom acaba amb C, és la part de la configuració i s'indica en ella els components (mòduls) que farem servir i com és connectaran, nom que els hi donem, en aquest mòdul. La segona part seria la implementació en si.

MÒDUL ONOFF

La funció d'aquest mòdul és el control de l'activació i desactivació de la mota. De fet la mota és pot trobar en tres estats diferenciats.

ESTAT DE CONSULTA O RESTABLIMENT

El primer seria un estat de consulta, quan la mota rep energia, és a dir, fa el que en anglès s'anomenaria un *start up*, la mota no sap si aquest s'ha produït una iniciació controlada o no.

Potser que el motiu de perdre l'alimentació hagi estat perquè és la primera instal·lació després de ser programada o per una caiguda de tensió o per un apagat forçat per un mal funcionament.

Així dons s'aprofita aquest primer estat per a preguntar a l'aplicació PC l'estat d'aquella mota, aquesta comunicació però la farà el mòdul de control que veurem més endavant. D'aquest estat se'n surt en rebre des de el PC l'estat previ a la pèrdua de contacte amb la mota.

MOTA ACTIVADA O DESACTIVADA

Els altres dos estat en que podem trobar la mota són reconeguda i activada o reconeguda però no activada. Per tal de realitzar aquest control és fa servir el sensor d'efecte Hall, que treballa com un polsador, dons mitjançant un circuit d'adaptació activa o no un port d'entrada digital de la placa. Mitjançant aquest sensor i un sistema de control de repeticions modifiquem l'estat de la mota.

Passarem d'un estat desactivat a un activat si activem dos cops seguits el sensor i el canvi invers si és produeixen quatre repeticions, aquestes repeticions han de ser bastant seguides en el temps, si de fet és deixa passar un temps entre les repeticions haurem de tornar a començar. Veiem un petit diagrama per aclarir el sistema emprat.

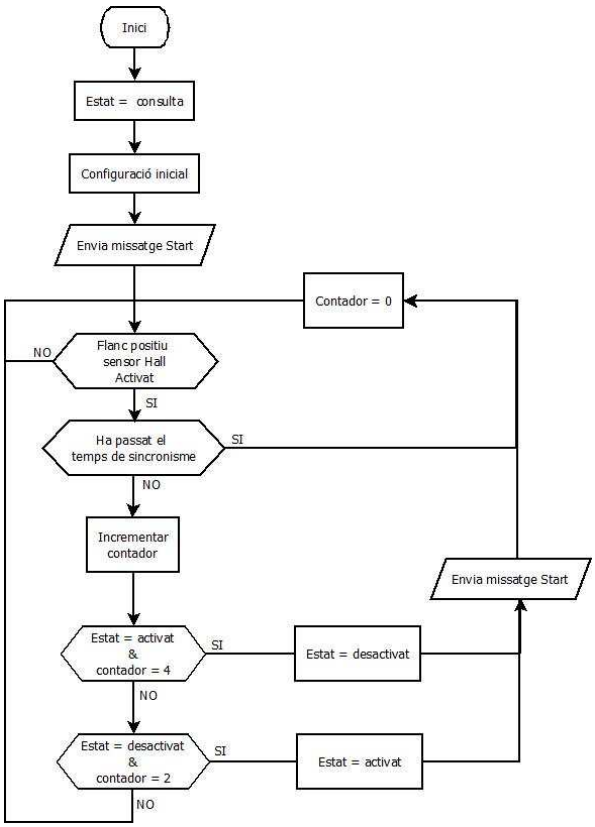

Figura 4.2.a: Algoritme del mòdul OnOff

Aquest algoritme gràfic que és pot veure en la figura anterior, indica d'una manera molt esquematitzada el procés de control de les activacions i desactivacions. En realitat el sistema és bastant més complex ja que a part de treballar en mòduls la dupla *nesC-TinyOs* també proporciona eines per treballar en multi-tasca i gestió d'esdeveniments.

Farem servir aquest mòdul que és bastant senzill, en comparació per exemple al de control, per explicar les metodologies utilitzades.

Quan volem realitzar una tasca paral·lela a l'actual, hem de fer servir la comana *post* com per exemple en aquest mòdul a l'hora d'enviar un missatge de Start, farem:

```
post enviaStart();
```

Degut a treballar en multi-tasca, el flux de programa pot arribar a canviar d'una part del codi a un altre, sense control del programador, per tant és important assegurar que aquells llocs que són crítics, com poden ser la gestió de variables compartides per més d'una funció, aquestes variables no siguin modificades erròniament, per tal d'evitar aquest problema existeix la paraula reservada *atomic* que fa que un tros de codi s'executi des de principi a fi sense alteracions. Hi ha que anar en compte amb utilitzar aquesta instrucció en trams llargs de programa ja que això fa que l'avantatge de treballar en multi-tasca es perdi.

Això farà que el flux del programa llaci la petició i executi el contingut de la tasca enviaStart, on simplement es generarà el paquet amb el camps propicis i s'enviarà fent servir comanes de les interfícies que proporciona *TinyOs*. Per cridar aquestes *comanes*, farem servir la instrucció *call*. En la següent línea es veu el codi necessari per tal de activar una temporització única de 2s que és el temps màxim perquè les activacions del sensor Hall s'incrementin.

```
call TimerControl.startOneShot(2000);
```

També es treballa com ja s'ha dit mitjançant esdeveniments, és a dir quan succeeix algun canvi al sistema es llancen ordres que es poden aprofitar o no, en el cas d'aquest mòdul fem servir els esdeveniments per exemple quan la mota rep un missatge, s'executa una subrutina que a part de fer altres accions retorna el missatge rebut. Veiem com seria l'encapçalament de la funció que tracta l'esdeveniment de la rebuda d'un missatge de tipus Start.

```
event message_t * ReceiveStart.receive(message_t *msg, void *payload, uint8_t len){
        .
        .
        .
        return msg;
}
```

El fet de que aquest mòdul sigui senzill a permès analitzar una mica més la filosofia de programació aquesta filosofia s'aplica a tots els altres mòduls.

Aquest mòdul també farà una petita gestió dels indicadors lluminosos, de fet mentre estem en estat de consulta, els leds verd i vermell estan activats i el groc commuta cada missatge enviat, el temps de consulta serà el que necessiti la mota per posar-se en contacte amb el PC la primera vegada que rep tensió d'alimentació. Per tant i si es tracta de la primera vegada que és connecta pot servir per tindre una idea de la cobertura que hi ha en aquella ubicació respecte a les comunicacions RF.

L'aplicació també te la activat un sistema de *watchdog* que forçarà un reinici de l'aplicació si questa es queda bloquejada més de quatre segons, el sistema de *watchdog* és torna a iniciar dins del bucle principal del mòdul de control.

MÒDUL UserBUTTON

Aquest mòdul encara és més senzill de fet no té més que una missió que és controlar les pulsacions del boto d'usuari. Cada cop que polsem mantindrà a 1 una variable durant una fracció de temps de 150ms suficient perquè el mòdul de control pugui fer la seva lectura. Aquesta fracció de temps serà sempre de la mateixa durada independentment del temps que és mantingui polsat el boto.

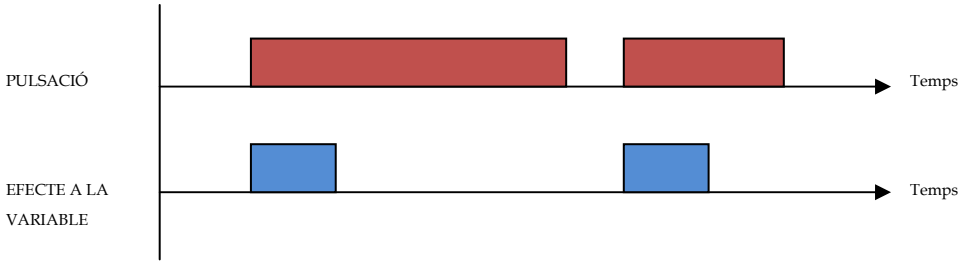

Figura 4.2.b: Funcionament del polsador d'usuari.

MÒDUL SENSORS

En realitat dels quatre sensors que disposa la plataforma només s'utilitzen tres. El primer dels tres que fem servir, és un sensor d'efecte Hall per detectar magnetisme que ja s'ha vist en l'apartat on s'ha descrit el mòdul OnOff.

Els tres sensors restants són molt semblant en la manera de treballar ja que si be la seva lectura depèn de magnituds físiques diferents temperatura, lluminositat i nivell de bateria, totes entreguen un valor analògic de la lectura. Dels quals només fem servir el de temperatura i el de bateria.

LA plataforma només disposa d'un *ADC*[1] el qual és comparteix mitjançant un sistema de multiplexació pels les quatre entrades analògiques que té la plataforma.

1. ***Maquina d'estats:*** *Tècnica de programació seqüencial basada en estats i transicions que permet un control molt robust de sistemes amb naturalesa seqüèncial. Molt utilitzada en processos industrials.*

Per tal de realitzar un correcte i robust control dels dos sensors analògics, farem servir una *maquina d'estats*[1] on anirem adquirint un a un els quatres sensors, encara que només volem la informació de dos d'ells farem la lectura de tots tres per tal de poder aprofitar en un futur aquest mòdul en properes aplicacions. El període de captura dels tres sensors l'ajustarem a 500ms això vol dir que cada mig segon tindrem una actualització de les mesures dels sensors.

Veiem un esbós de la màquina d'estat emprada, normalment l'esquematització d'una maquina d'estat és realitza gràficament ja que hi solen haver transicions múltiples i salts de retorn entre els diferents estats. En aquest cas al ser una màquina totalment lineal i per simplicitat és presentarà mitjançant una taula.

Estat	Significat
READY	Estat inicial que serveix per activar els sistemes i esperar que s'estabilitzin, passat un temps de 10ms passem al següent estat.
WARMED_UP	Sensors activats correctament passem a llegir el primer sensor.
READING_BAT	Estat de procés de lectura del nivell de bateria.
BAT_READ	Estat de lectura correcta del nivell de bateria.
READING_PHOTO	Estat de procés de lectura del nivell de lluminositat
PHOTO_READ	Estat de lectura correcta del nivell de lluminositat.
READING_TEMP	Estat de procés de lectura de la temperatura.
TEMP_READ	Estat de lectura correcta de la temperatura.
SENSORS_READ	En aquest estat els sensors ja han estat llegits, podem fer amb aquesta informació el que és vulgui, en l'aplicació en aquest estat no és fa res, ja que serà el control qui vindrà a buscar la informació quan ho necessiti. S'ha deixat per poder aplicar aquest mòdul a futures aplicacions.
ERROR	Estat en que va la màquina si hi ha algun descontrol, fa que la màquina torni a l'estat inicial.

Taula 4.2.b: Taula d'estat de la màquina per la lectura dels sensors

Com ja s'ha dit serà el mòdul de control el que quan ho necessiti preguntarà els valors dels sensors a aquest mòdul.

MÒDUL DE CONTROL

Aquest és possiblement el mòdul més important de tota l'aplicació és l'encarregat de totes les comunicacions, excepte la del missatge d'inici que és fa directament al mòdul OnOff, i d'altre banda és on es prenen les decisions pel que fa al control de tota l'aplicació.

L'estructura principal del mòdul és un bucle infinit que s'executa cada 100ms per tant totes les decisions temporals que fem les hem de calcular en base a aquest temps. Per realitzar aquest control anirem incrementant un comptador i per tant si volem comprovar un temps d'un segon, haurem de comptar fins a 10. També s'ha de tindre en compte perquè no podrem controlar situacions que puguin variar més ràpidament de 100ms.

Al inici de cada cicle, el sistema recull la informació dels altres mòduls, aquí hi ha una incongruència ja que si els cicles és repeteixen cada 100ms i els sensors com s'ha dit s'actualitzen cada 500ms no faria falta revisar el seu valor com a mínim fins cada 500ms, aquest error no afecta al correcte funcionament de l'aplicació però podria millorar-se en futures versions de cara a l'eficiència.

A part d'aquest bucle general, i gracies al que ja s'ha explicat sobre que estem treballant amb un sistema multi-tasca, cada cop que el control vulgui enviar un missatge a l'aplicació PC ho farà mitjançant el llançament d'una tasca nova.

De la mateixa manera, els esdeveniments de recepció de missatges treballaran paral·lelament a aquest cicle i modificaran la configuració o el mètode de treball del cicle principal.

Les funcionalitats que estan contemplades són les següents:

Funcionalitat	Explicació
Control estat de la mota	Determina si la mota és activa, en cas de no estar activa totes les funcionalitats següents queden inhabilitades.
Alarma automàtica	Configurable des de l'aplicació PC, podem ajustar el nivell en que s'activarà i el temps que ha d'estar en aquest nivell per a activar l'alarma.
Alarma manual	En qualsevol moment en que l'alarma estigui inactiva podem polsar el botó d'usuari i activar una alarma manual.
Confirmació d'alarma	Si l'alarma ha estat activada d'alguna de les dues maneres anteriors, des de el PC podrem confirmar-la, des del mòdul farem els ajustos necessaris per tal de deixar constància d'aquest fet.
Restablir alarma	Un cop confirmada l'alarma podrem polsant el botó d'usuari restablir-la al seu nivell de inactiva.
Monitorització de temperatura	També és configurable des de el PC, per tant amb un esdeveniment de recepció de missatge, podrem dir si volem o no fer una monitorització de la lectura de la temperatura, també podrem configurar la periodicitat.
Monitorització de la bateria	En aquest cas només es pot ajustar el temps de periodicitat ja que sempre haurà d'estar activa.
Control del nivell de bateria	Si en algun moment la lectura del nivell de bateria baixa per un llindar preestablert el sistema ho indicarà per tal de que les bateries siguin reemplaçades.
Resposta a missatges de Hello	Quan la mota rep un missatge de Hello, és llançarà un esdeveniment i aquest activarà una tasca per tal de respondre a emissor del primer missatge.

Taula 4.2.c: Taula de funcions controlades pel mòdul de Control

La manera de gestionar tot el mòdul és molt senzilla dins del bucle principal és comproven totes les condicions de si s'ha d'activar o tindré en compte alguna de les funcionalitats anteriors, i en funció de les respostes d'aquestes és realitzen les accions corresponents com pot ser per exemple el següent exemple, del control de la monitorització de la temperatura.

```
if (debugTa==1){
        if (call TmEnviaDbgTa.isRunning()==0)
                call TmEnviaDbgTa.startPeriodic( tempsDbgTa * 1000 );
        }
else call TmEnviaDbgTa.stop();
```

GESTIÓ DELS MISSATGES

Fins ara no s'ha parlat de la gestió que fa la mota dels missatges i quins tipus és poden trobar. Recordem que en aquesta aplicació tots els missatges són entre una de les motes i el PC, o entre el PC i una de les motes, encara que també hi ha missatges en format *broadcast*[1].

És presenta a continuació una taula resum amb tots el missatges, el codi, necessari per tal de que els sistemes receptors sàpiguen quin esdeveniment han de llançar.

Missatge	ID	Estructura		Funció
		Tipus	Nom	
AM_MOFFMSG	40	Byte	moteId	Dona informació a l'aplicació PC de quin és l'estat de la mota.
		Byte	Off	
AM_HELLOMSG	41	Byte	moteId	Simplement és un missatge de comprovació que podem fer servir per veure la cobertura de les motes.
		Byte	seqNum	
		Byte	Ack	
AM_ALARMMSG	43	Byte	moteId	Gestiona l'enviament de totes les alarmes, tant les automàtiques com les manuals
		Byte	alarm	
		Byte	Ack	
AM_NEWLIMITMSG	44	Byte	moteId	Estableix uns nous nivells d'activació de l'alarma automàtica.
		Enter	counts	
		Enter	time	
		Byte	Ack	
AM_DBGMSG	45	Byte	moteId	Actualitza les monitoritzacions tant de la temperatura com del nivell de bateria
		Enter	Dbg	
		Enter	temps	
		Enter	Ack	
AM_DBGTAMSG	46	Byte	moteId	Envia el valor de la temperatura instantània
		Enter	counts	
		Enter	vref	
AM_DBGBATMSG	47	Byte	moteId	Envia el valor actual del nivell de la bateria.
		Enter	counts	
		Enter	vref	
AM_STARTMSG	48	Byte	moteId	Gestiona tot el referent a l'encesa de la mota. Controlat des de OnOff.
		Byte	code	
		Byte	Ack	

Taula 4.2.c: Taula de funcions controlades pel mòdul de Control

Com és pot apreciar a la taula anterior hi ha missatges que no contenen el camp Ack aquests missatges no necessiten ser persistents ja que la informació que s'envia si be si que és important es transmesa d'una manera periòdica, per tant si un missatge no arriba no és crític.

A continuació es presenta un diagrama representatiu d'un missatge persistent en concret la gestió d'una alarma manual.

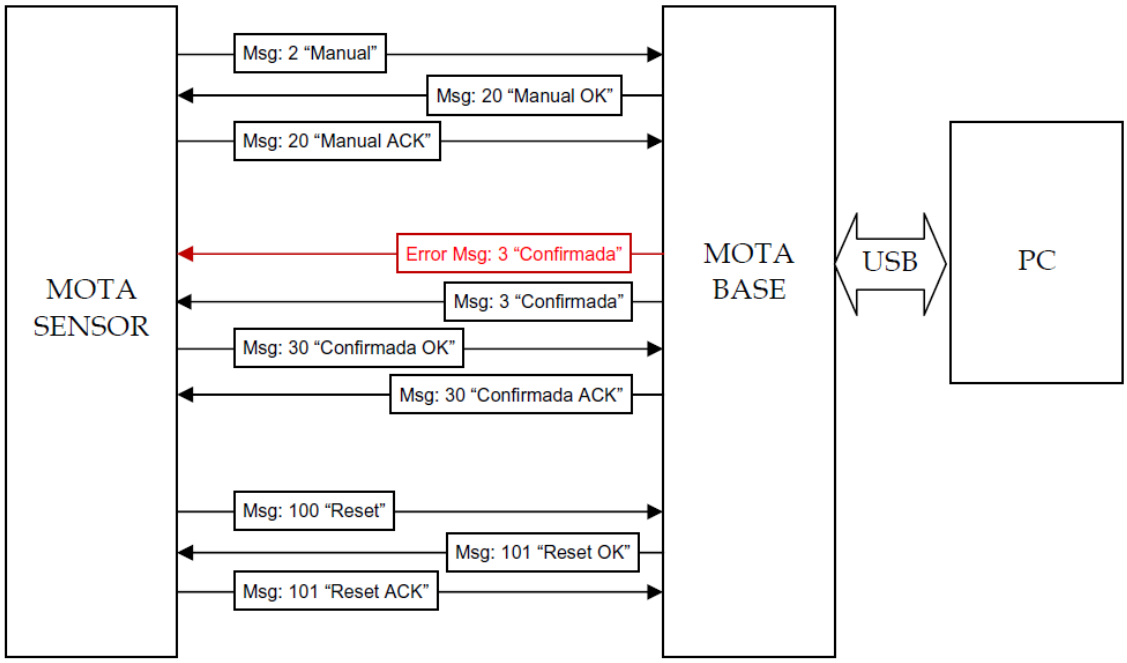

Figura 4.2.c: Detall de comunicació per una alarma manual.

En la imatge anterior, és pot veure que si algun dels enviaments falla, és a dir no arriba al destí, és repeteix l'enviament, això és així excepte amb els últims de cada missatge, els que estan marcats amb ACk, aquest ja no són importants, ja que a l'hora de ser enviats les dues parts ja tenen actualitzats els diversos estat.

Els altres missatges tindrien una estructura similar.

GESTIÓ D'ALARMES

La gestió dels missatges referents a les alarmes és una mica complexa per tan s'ha cregut convenient el fet d'explicar-la. És pot veure a la següent taula que hi ha codis repetits això és degut a que no és el mateix que una alarma hagi sigut activada a que també aquesta activació hagi sigut rebuda per l'aplicació del PC.

Codi Alarma	Significat
0	Alarma no activa o en estat normal, també té aquest estat un cop restablerta
1	Alarma activada de forma automàtica
10	Alarma activada de forma automàtica i rebuda a l'aplicació central
2	Alarma activada de forma manual
20	Alarma activada de forma manual i rebuda a l'aplicació central
3	Alarma confirmada per l'aplicació central.
30	Alarma confirmada per l'aplicació central i rebuda a la mota.
100	S'ha realitzat un reset de l'alarma.

Taula 4.2.d: Codis dels estat de l'alarma.

INDICADORS LLUMINOSOS

Mitjançant els indicadors lluminosos la mota pot comunicar el seu estat als possibles usuaris, a la següent taula podem veure les diferents indicacions i els seus significats. Primer de tot dir que si tots els leds verd i vermell estan encesos a l'hora sense intermitències i el groc fa intermitències, vol dir que la mota esta buscant la primera connexió amb el sistema.

LED	Estat	Significat
Vermell	Actiu	Alarma activada
	Intermitent ràpid	Alarma rebuda a l'estació central
	Intermitent ràpid	Alarma confirmada
Groc	Intermitent	Bateria baixa
Verd	Intermitent lent	Mota inactiva
	Intermitent ràpid	Mota activa

Taula 4.2.e: Indicacions de les motes.

4.3 Explicació de l'aplicació del PC

L'aplicació del PC és pot separar en dos grans parts si s'observa la figura 3.2.a, es pot veure clarament que hi ha un grup de classes destinades al tractament de la pantalla i la iteració amb l'usuari i un altre que clarament està destinada al propi control de l'aplicació.

CLASSES DE CONTROL

De fet es pot dir que només hi ha una classe important que és *control.java*, ja que les altres classes que intervenen són totes les referents als missatges que s'han vist anteriorment.

La classe control per la seva banda, la es pot estudiar de la mateixa manera que el mòdul de control de les motes, de fet la filosofia ve a ser la mateixa. Tenim un bucle principal que s'executa periòdicament cada 250ms, i unes accions laterals per a la gestió d'esdeveniments que en aquest cas són únicament les recepcions de missatges.

Un cop és rep un missatge, s'identifica el tipus de missatge i segons sigui és modifiquen les variables o prenen decisions degudes.

Per un altre banda un cada cop que entrem en el bucle principal, el primer que fem és revisar els indicadors (flags) que ens indiquen les accions que hem de fer. Aquests flags, poden ser modificats per els missatges rebuts, per alguna acció de la classe de control o directament per l'usuari, des de l'aplicació gràfica.

S'ha de recordar que poden haver-hi fins a 20 motes treballant a la vegada, i que per cada mota tindrem els diferents flags, per tant cada cicle de treball s'han de revisar tots ells això ho fem sempre amb un bucle que recorre la llista de motes.

Un cop realitzada aquesta comprovació passarem a gestionar els flags, la majoria dels quals indiquen la necessitat d'enviar un missatge, per tant mirarem cadascun dels flags i anirem enviant els missatges un a un.

Aquí tenim un tros de codi que gestiona l'enviament del missatge de Hello.

```
/*********************** ENVIO HELLO  ************************************/
      if (motes[i].enviaHello){
                //S'envia un missatge a la mota
                println("-> Missatge Hello a mota "+i+", intent: "+motes[i].HelloTry);
                HelloMsg msg = new HelloMsg();
                msg.set_moteId((short) 1);
                msg.set_seqNum((short) motes[i].HelloTry);
                msg.set_Ack((short) 0);
                motes[i].HelloTry++;
                try {
                        mif2.send(i,msg);               // send brodcast
               } catch (IOException e) {
                        System.err.println("No s'ha pogut enviar el missatge a les motes");
                }
      }
```

Com es pot veure al codi hi haurà un flag de *enviaHello* en cadascuna de les motes de la llista i si aquest està actiu passarem a generar el missatge i a enviar-lo.

Per últim també s'ha de comentar com és comunica aquesta classe amb la classe de pantalla, existeixen dues necessitats una seria que des de la pantalla és pogués modificar els flags abans comentats i l'altre que des de control és pogués mostrar informació a l'usuari a través de la classe pantalla, per això utilitzarem un sistema de logs..

Treballar en un entorn orientat a objectes, la opció lògica seria generar uns atributs privats dins de la classe control i mitjançant uns rutines d'accés obtenir o modificar aquests atributs. A la realitat i per un tema de practicitat s'ha decidit fer servir atributs protegits que ja que són directament accessibles des de una classe del mateix paquet. Lògicament perdem la independència entre classes una de les bases de l'orientació a objectes, per tant també aquest podria ser un punt de millora.

CLASSES DE PANTALLA O PRENSENTACIÓ

En principi hi ha quatre classe, dues d'elles *pantallaAboutBox.java* i *pantallaHelpBox.java* però només són pantalles purament gràfiques amb missatges d'informació a l'usuari.

De les dues restants *pantallaApp.java* s'ha inclòs en aquesta secció de classes gràfiques però també hagués estat possible donar-li una categoria superior, ja que és la classe on és generen totes les classes restant. De fet es en aquesta classe on s'instancia control, és generen les escoltes de missatges i s'inicia el rellotge general de la classe control.

Només queda comentar la classe *pantalla.java* on és fa tot el referent a la comunicació amb l'usuari. A la següent figura és mostra una imatge de la pantalla de control, on s'han indicat amb números les diferents parts que després seran comentades.

Figura 4.3.a: Pantalla de l'aplicació amb indicacions.

EXPLICACIÓ DE LES DIFERENTS PARTS DE LA PANTALLA:

1 Configuració d'alarma

En aquesta part tenim dos camps i un polsador, el primer dels dos camps s'indicarà el nivell de temperatura i al segon el temps que ha d'estar la temperatura per sobre d'aquest nivell per tal d'activar l'alarma.

En el moment de polsar el botó, agafarà aquest valors i els enviarà a totes les motes a l'hora aquestes després contestaran si han rebut correctament el missatge.

Aquest canvi només és possible si l'alarma no esta en estat d'activació, és a dir que només es pot modificar si està desactivada o confirmada.

2 Comprovació de l'estat de les alarmes

L'aplicació disposa d'un selector que permet anar recorrent totes les motes del sistema, al costat té dos indicadors en el primer dels quals és pot veure l'estat de la l'alarma en aquella mota que com ja s'ha comentat, pot ser:

- NORMAL
- AUTOMATICA
- MANUAL
- CONFIRMADA

Al segon indicador és mostrarà la hora del sistema en el moment d'activació en cas de que l'estat sigui automàtica o manual. El botó *OK* serveix per confirmar les alarmes un cop activades, lògicament només confirma l'alarma que estigui seleccionada en el selector.

3 Monitorització de la temperatura

Quan s'activa la monitorització de la temperatura es fa per totes les motes del sistema, per fer-ho s'ha d'activar la icona de selecció i fixar un temps de refresc, un cop fet això polsarem el botó.

Si al polsar el botó, la icona de selecció no està activada o el temps està fixat a cero, el sistema deixarà de fer la monitorització.

A la part inferior d'aquesta zona, tenim el camp que ens mostra la temperatura quan la monitorització està activada i un missatge de *STOP* en cas que no estigui activada o sigui la mota seleccionada la que no estigui activa.

4 Monitorització del nivell de bateria

És una funcionalitat molt semblant a l'anterior, però amb dues diferencies la primera és que aquesta monitorització no pot ser aturada. I la segona que a part del nivell i *STOP* també podem un missatge *NO MSG*, aquest és mostrarà quan la mota suposadament estigui activa però haguí passat un temps sense comunicar aquesta informació per tant és suposa que hi ha un error de cobertura.

5 Gestió dels missatges de *Hello* i l'estat de les motes

En aquesta part tenim dues funcionalitats, la primera és mitjançant el sistema ja explicat és podrà visualitzar l'estat d'una mota.

La segona funcionalitat és enviar un missatge de *hello* a la mota seleccionada o be si la icona de selecció està activada fer una transmissió *broadCast*, aquest segon tipus de missatge pot servir per visualitzar totes les motes actives.

6 Hora del sistema

Podem veure l'hora del sistema en aquest lloc de la pantalla.

7 Sub-pantalla de registre general

En aquesta pantalla anirà surtin totes les informacions que afectin al sistema tant de missatges rebuts com de accions i decisions preses pel sistema.

8 Botó de Reset del registre

Mitjançant aquest botó podrem esborrar la sub-pantalla del registre.

Seguidament és mostra una imatge de l'aplicació funcionant.

Figura 4.3.b: Detall de l'aplicació funcionat.

5. Comercialització

5.1 Viabilitat tècnica

Com tot producte la seva finalitat és ser comercialitzat, el producte del projecte no és una excepció. Però lògicament el resultat d'aquest estudi i el temps dedicat a la seva creació no garanteix un èxit en la seva comercialització.

Primer de tot dir que per a poder comercialitzar aquest producte primer l'hauríem de certificar això comportaria al tractar-se d'un sistema de seguretat , aquest tipus de sofware han treballar amb redundancia i existeixen diversos nivells, s'hauria de mirar la normativa vigent referent a aquest sector per a establir un pla d'acció però això no està contemplat en l'abast del projecte.

De fet en el transcurs del desenvolupament i en les proves realitzades posteriorment, s'han detectat errors que s'haurien de resoldre abans de pensar en poder fer una comercialització.

ERRORS DETECTATS

Podríem dir que els problemes, errors que actualment té l'aplicació són un total de tres, dos d'ells a l'aplicació de la mota i un altre a l'aplicació del PC.

1. De totes maneres només hi ha un que pot ser considerat exactament un error, que és el ja comentat, la mota que fa la connexió amb el PC ha de treballar amb l'aplicació BaseStation, el motiu és que no he pogut fer una unió amb aquest components i els que s'han dissenyat per solucionar el sistema d'alarmes.

   La raó no la acabo de tenir clara hi he dedicat bastant de temps però no trobo el problema. He provat varies estratègies, com no fer les comunicacions broadcast, també he eliminat tot el que feia referència a l'activació de les motes, ja que aquesta sempre estaria activa. El resultat no ha estat satisfactori, he adjuntat l'aplicació anomenada AM_BASE_ID1, que és el últim resultat del meu estudi, encara que no funciona.

2. És un problema que ja ha estat comentat al llarg del document. Si estan tot el sistema activat tanquem l'aplicació del PC, en tornar a activar-la no fa cap comprovació de l'estat de les motes i aquestes per a tornar a connectar-les s'els hi ha de fer un reset.

   Hi ha varies formes de solucionar aquest problema, per exemple crear un missatge de *kill* que al encendre l'aplicació de PC s'envia a totes motes i aquestes és fan un reset. Podria ser forçant un desbordament del watchdog. Un altre opció podria ser que si l'aplicació rep un missatge de nivell de bateria d'una mota que suposa que és troba desactivada canvi el seu estat a actiu.

   Les solucions són en principi de fàcil aplicació, el motiu de no haver aplicat aquestes modificacions ha estat la dedicació quasi completa del temps al primer punt.

3. D'una manera semblant un problema que té fàcil solució i milloraria el conjunt del sistema, és que per guardar les diferents dades de les motes, s'ha fet servir un Array de 20 posicions, que són bastants però poden ser no suficients.

   Lògicament java permet fer el control dinàmic amb un ArryList, la millora seria dons molt senzilla de realitzar, però al mateix temps seria laboriosa degut a l'estructura actual de l'aplicació. Com al punt 2, s'ha preferit dedicar temps a intentar solucionar el problema 1.

POTENCIAL TÈCNIC

De totes maneres i malgrat els errors comentats, aquestes són de fàcil solució en una propera versió, deixant el sistema preparat per a la correcta comercialització pel que fa al programari, faltaria però fer algunes modificacions en el disseny, per exemple una carcassa per les motes que fos més funcional i complís amb les normatives de instal·lació a edifici.

5.2 Valoració econòmica

VALORACIÓ DEL DISSENY

Lògicament per a la comercialització d'un producte s'ha de tindre en compte el cost del disseny i no sols el cost de la producció un cop llançat al mercat. De fet normalment al cost de producció se li afegeix un tant per cent del cost del disseny, la quantitat del qual depèn del numero de ventes previstes.

No s'ha tingut en compte el cost d'una possible certificació a la normativa vigent cosa que com ja hem explicat seria del tot necessària.

El motiu de realitzar aquest càrrec al producte final és per tal de recuperar la inversió inicial o cost del disseny.

En aquest producte el cost del disseny que és presenta a continuació només està inclòs el cost del disseny del programari, ja que la part del maquinari, és a dir les motes sobre les que s'ha treballat s'han adquirit de un tercer i per tant només valorarem el seu cost.

Per establir un preu de cost de la mà d'obra a un preu raonable segons el mercat actual per la categoria d'enginyer tècnic. S'han computat totes les hores de Disseny, Formació i Documentació a un preu diferent.

Motiu	Cost unitat (€)	Quantitat	Cost (€)
Ordinador personal	600	1	600
Monitor	150	1	150
Plataformes (Motes)	40	2	80
Hora de Formació	25	48	1200
Hora de Disseny	40	138	5520
Hora de Documentació	30	52	1560
		Total	**9110 €**

Taula 5.2.a: Pressupost del disseny.

VALORACIÓ DE LA COMERCIALITZACIÓ

Tot seguit és realitzarà la valoració del cost i preu de venda del producte final, per fer aquest estudi no s'ha tingut en compte el cost de la segona part del disseny de que s'ha parlat en l'apartat de viabilitat tècnica.

Encara que potser alguna de les activitats és pugui realitzar en menys temps la fracció mínima de temps serà de 0,25 hores per tal d'establir un criteri clar. El cost d'aquestes hores s'ha computat un grau més baix que les de disseny ja que la qualificació dels tècnics que les realitzaran serà més baixa.

El següent s'ha realitzat l'estudi del cost per una instal·lació d'un sistema de 10 motes a un possible client situat a una localitat de 100km de distancia.

Motiu	Cost unitat (€)	Quantitat	Cost (€)
Ordinador personal	600	1	600.00
Monitor	150	1	150.00
Plataformes (Motes)	40	10	400.00
Hora de Programació per mota	20	2.5	50.00
		Cost de la preparació del dispositiu	**1200.00**
Kilometratge del tècnic	0.04	100	4.00
Hora d'instal·lació	20	5	100.00
		Cost en brut	**1304.00**
		retorn de inversió..... 0,1%	1.30
		Benefici comercial..... 60%	782.40
		Total	**2087.70**
		x 18% d'IVA	375.79
		Total	**2463.68€**

Taula 5.2.a: Pressupost de la instal·lació.

Com és pot observar el producte final no té un cost excessiu, si el comparem amb les alarmes antiincendis del mercat el seu preu està bastant per sota. També val a dir que les prestacions de les alarmes que podem trobar al mercat són bastant més elevades.

6. Conclusions

En aquesta part del document l'autor que ha mantingut un llenguatge formal en tot el document, canvia a un llenguatge de to més personal, ja que el que vindrà a partir d'ara són sobre tot opinions pròpies.

6.1 Conclusions

Dels objectius inicials, que podem trobar en la primera secció dels annexos, podríem dir que s'han satisfet un alt percentatge, de fet s'han assolit totes les funcionalitats, excepte les que ja s'han comentat en l'apartat de comercialització, d'altre banda s'han realitzat algunes millores sota el meu parer com pot ser que les alarmes és pugin restablir al seu valor normal després de ser activades.

El resultat del projecte, encara que no està del tot acabat, sempre es podrà millorar, de fet aquest és la filosofia de la millora continua una de les bases de la qualitat industrial tant de moda en l'actualitat, si que podem parlar d'un producte quasi acabar amb poques modificacions podria ser llençat a un sector de mercat bastant ampli.

El fet de programar en un llenguatge nou, encara que basat en el llenguatge C amb el que he programat molt, ha fet que aprengués eines noves, que suposo hem seran d'utilitat en un futur.

També he aprofitat la necessitat de realitzar una aplicació que interactués amb un usuari per tal de dissenyar una aplicació de finestres gràfiques cosa que encara que sembli mentida no havia fet fins ara tot i cursar completament una carrera tècnica de sistemes.

6.2 Proposta de millores i línees de futur.

Ni ha moltes sobre tot solucionar els problemes ja comentats. El primer que s'hauria de fer és solucionar aquest problemes, i no estaria malament fer una revisió de tot el codi per tal de fer-lo més eficient.

D'altre banda seria interessant satisfer tots els requisits extres proposats amb l'enunciat del TFC, que no s'han dut a terme, com d'encriptació de dades o una interfície web.

Una de les coses interessant que és podria fer és que les motes és poguessin enviar, ja poden però no està considerat, missatges entre les motes o que fins i tot hi poguessin haver més d'una estació central.

També és podria desenvolupar una aplicació per poder realitzar el control del sistema des de un terminal mòbil, ja que a dia d'avui és bastant comú aquest tipus d'aplicacions i han passat a ser més que un valor afegit una necessitat per entrar en el mercat.

Altres línees de millora podria ser i de cara a la comercialització estaria be plantejar-se l'aspecte de la certificació dins de les normatives actuals, s'ha de recordar que un producte ha d'estar marcat amb el certificat CE per tal de poder-lo comercialitzar dins de la comunitat europea.

6.3 Autoavaluació.

Tot el procés del TFC ha estat bastant interessant per a mi, en realitat el punt de partida no era des de cero pròpiament dit, partia d'una base molt forta en la programació de µC però mai havia programat sota el sistema operatiu TinyOs encara que si ho havia fet en sistemes de temps real com per exemple VxWorks, però d'això fa molt de temps i la veritat és que m'ha costat més del previst fer-me amb aquesta manera de programar.

Considero que el procés de construcció del software no ha estat del tot malament, tot i que si hagués de tornar a començar com sol ser habitual ho faria d'un altre manera. Estic segur que els objectius que no s'han assolit podrien ser satisfets amb una mica més de temps, amb això no vull dir que no sigui un problema el fet de no haver-los desenvolupat ja que l'organització del temps també és una cosa avaluable.

Al llarg del procés de programació m'han anat apareixent problemes a mesura que anava fent probes sobre l'aplicació. Un dels que vaig haver de solucionar per exemple, va ser que és perdien massa missatges d'activació d'alarma, degut a que no gestionava del tot bé els missatges, vaig decidir dons fer el sistema de comunicacions que s'explica en aquest document.

Referent al temps, sense cap mena de dubte el temps dedicat ha estat superior al que inicialment tenia previst, encara que això és normal en processos de llarga durada com ara aquest, he de dir que el si ens fixem el numero de crèdits d'aquesta assignatura dins del currículum de la carrera, les hores dedicades són molt superior potser fins i tot tres o quatre cops més que d'altres assignatures del mateix suposat volum.

Si be és cert el comentat en elparagraf anterior, també val a dir que el temps dedicat és bastant més agraït.

Cal dir que també és cert que no he pogut per temes externs al projecte, mantenir un ritme de treball constant i això encara a provocat més diferencies amb el planificat en un inici. De totes maneres després d'hores robades a les nits i al lleure, crec que el resultat és satisfactori.

D'altre banda veig que el projecte te molts punts millorables per no dir fluixos, soc conscient però que no hem de perdre de vista que és tracte d'una primera fase del disseny, encara que possiblement no n'hi haurà de futures, si és tractes d'un producte destinat al mercat, hi haurien moltes coses a polir en fases futures.

En definitiva crec que en un alt tant per cent s'han assolit els objectius inicials. En un inici vaig preveure que faria algun dels objectius extres però per circumstancies això no ha estat possible, el que si que he fet per una motivació personal és la realització de l'aplicació gràfica.

Tot i que sempre és podria haver treballat una mica més, una mica més constant i d'una manera potser més ordenada, sincerament estic content dels resultats obtinguts.

Malgrat la gran taca de no poder realitzar la unió de la meva aplicació amb la BaseStation. ¿Qui sap si en un futur? Però ja fora de concurs.

7. Glossari

***D*iagrama de Gantt**: és tracta d'una eina per a representar de forma visual i clara la temporització de les diferents tasques que composen una activitat i la relació que hi ha entre elles.

domòtica: Tecnologia bassada en sistemes electrònics dissenyats per al seu ús en edificis ja siguin industrials o de vivendes que al ser instal·lats aporten un augment del confort, estalvi energètic i la seguretat dels inquilins o usuaris.

HMI: Sigles que provenen de l'anglès "human machine interface" i signifiquen sistema de comunicació entre el usuari i la màquina o aplicació.

Maquina d'estats: Tècnica de programació seqüencial basada en estats i transicions que permet un control molt robust de sistemes amb naturalesa seqüèncial. Molt utilitzada en processos industrials.

meshprog: programa de transferència de dades a través del port USB que pot fer servir diferents tipus de protocols de transmissió de dades via sèrie, com poden ser (JTAG, ISP,...)

motes: Sistema encastat destinat al seu ús distribuït bàsicament compost de un µC i els seus perifèrics on podem trobar com a component destacat un sistema de comunicacions sense fils, així com una sèrie de sensors.

nesC: (Network embedded System C) llenguatge de programació bassat en C, dissenyat especialment per a treballar amb xarxes de sensors remots.

TinyOs: Sistema Operatiu amb filosofia de programari lliure, basat en l'ús de components i la seva interconnexió, destinat al govern de sensors remots.

8. Bibliografia

Wiki de l'assignatura de sistemes encastats de la UOC
http://cv.uoc.es/app/mediawiki14/wiki/Inici24#Learning_TinyOS
**En aquesta pagina s'ha anat navegat a d'altres pagines de la mateixa temàtica*

Wiki de documents de tinyOs
http://docs.tinyos.net/tinywiki/index.php/Main_Page

Philip Levis
(2006, Juny), "TinyOS Programming"
http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf
**Document PDF molt interessant sobre programació en TinyOs (anglès)*

David Gay, Philip Levis, David Culler, Eric Brewer
(May 2003), "nesC 1.1 Language Reference Manual"
http://nescc.sourceforge.net/papers/nesc-ref.pdf
**Document PDF molt interessant sobre el llenguatge nesC (anglès)*

Jose Maria Alcaraz Calero
http://ants.dif.um.es/rm/apuntes/Tutorial-Slides.pdf
**Document PDF molt interessant sobre el llenguatge nesC (castellà)*

Jose Maria Alcaraz Calero
http://ants.dif.um.es/rm/apuntes/Tutorial-TinyOS.pdf
**Document PDF molt interessant sobre programació en TinyOs (castellà)*

Datasheet microcontrolador atmega1281
http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf

Esquema mota COU
http://eimtcollab.uoc.edu/softcou24_12/mota_cou_1_2_24A2_2011.pdf

Cataleg d'alarmes antiincendi
http://www.elecam.com/imagenes/Catalogo_panasonic.pdf

7. Annexos

7.1 Llista detallada de objectius

A continuació podem veure una llista detallada dels objectius i requisits del projecte:

1. Detectar incendis
   (a) Detecció per temperatura. Quan la temperatura supera un llindar (TEMP_ALARM) el node ha d'enviar l'alarma.
   (b) El nivell d'alarma s'ha de poder configurar des de l'aplicació de l'usuari.
   (c) El sistema agafarà una dada del sensor cada N segons per tal de comprovar el llindar.
   (d) N ha de ser configurable per l'usuari

2. Notificar automàticament l'alarma de forma remota.
   (a) Un cop es detectada l'alarma aquesta s'ha d'enviar de forma automàtica.
   (b) La transmissió de dades serà a través d'una comunicació sense fils.

3. Notificar automàticament l'alarma de forma local.
   (a) S'ha d'informar de l'alarma de forma local, mitjançant un senyal visual (led/s)

4. Sistema manual per activar l'alarma
   (a) El node ha de proveir d'un botó d'emergència per tal d'activar l'alarma de forma manual

5. Transmissió d'alarmes sense pèrdues
   (a) No es pot perdre cap alarma d'incendi
   (b) La comunicació s'ha de fer amb ACK

6. Reconeixement d'alarmes des de l'aplicació d'usuari
   (a) L'aplicació ha de mostrar per pantalla quan hi ha hagut una alarma.
   (b) Ha de mostrar:
      i. Tipus d'alarma (manual o automàtica)
      ii. Quin sensor ha detectat l'alarma i quina temperatura detecta.
      iii. Hora en que s'ha produït.
   (c) L'usuari ha de poder confirmar la recepció d'alarma.

7. Monitoritzar la bateria dels nodes de la xarxa.
   (a) Cada node enviarà la seva bateria de forma periòdica cada L segons
   (b) L ha de ser configurable des de l'aplicació
   (c) Quan el node detecti que té la bateria critica, és a dir per sota de un llindar (BAT_LVL_ALARM), enviarà una alarma notificant del seu estat
   (d) Aquest missatge es farà servir per controlar que el node funciona correctament. Si el PC fa més de L segons que no rep un missatge del node, mostrarà una alarma per pantalla.

8. Mostrar els diferents estats del sistema en el node:
   (a) Sensor apagat. De forma periòdica pot encendre un led, conforme el node pot funcionar
   (e) correctament.
   (b) No alarma. De forma periòdica pot encendre un led, conforme el node esta funcionant
   (f) correctament.
   (c) Alarma detectada
   (d) Alarma rebuda en estació central
   (e) Alarma reconeguda
   (f) Nivell bateria baix
   (g) Fora de cobertura

9. Sistema de protecció de caigudes. Els nodes no es poden quedar "penjats" (WatchDog)
   (a) El node ha de fer servir el WatchDog per evitar quedar-se penjat.
   (b) Ha de recuperar la configuració de forma automàtica

10. Sistema de debug de l'aplicació.
   (a) El usuari ha de poder veure les temperatures dels diferents sensors.
   (b) El usuari pot demanar rebre les dades cada M segons. M ha de ser configurable.

11. Prova de cobertura
   (a) Proporcionar un sistema per comprovar si on s'instal·la el node hi ha cobertura amb el node base.

12. Sistema d'activació
   (a) El sensor no estarà sempre encès. Per tal d'iniciar el sensor, es farà ús del sensor d'efecte Hall. Per tal d'encendre el sensor s'haurà de passar un imant 2 vegades seguides per sobre d'aquest. Per apagar-lo s'haurà de passar 4 vegades seguides per sobre.

13. Interfície d'usuari
    (a) Ha de ser simple
    (b) Autoinstal·lable i autoexecutable
        i. script d'instal·lació, ...
        ii. Contingui totes les llibreries necessàries
        iii. Manual d'instal·lació
    (c) Contenir un menú d'ajuda (help amb les comandes)

7.2 Execució i compilació

Per a la poder executar les aplicacions , serà necessari un ordinador amb linux i TinyOs instal·lats correctament. La instal·lació consta de quatre etapes, que són:

1. Instal·lació de les aplicacions que són tres:

    - Com ja s'ha dit la mota que haurà d'estar connectada al PC ha de dur l'aplicació BaseStation que inclou tinyOS en les seves llibreries. Aquesta mota, serà sempre la mota amb identificador 1, ja que totes les demés envien a aquesta adreça per a comunicar-se amb el PC.

    - Les altres motes, les motes sensores portaran l'aplicació anomenada AM_BASIC, que s'adjunta al l'informe. Haurem de fer la configuració de cada identificador de mota a l'hora de la instal·lació, els nombres que podrem agafar seràn superiors a 1 i en aquest cas inferiors a 20. El motiu s'explicarà en l'apartat de millores. Per fer això necessitarem dues linees de comanes a un terminal de linux, ho farem així.

        ```
        Make cou24 install,X

        Meshprog –t/dev/ttyUSB0 –f./build/cou24/main.srec.out-X
        ```

        Tot seguit haurem de polsar el boto reset de la mota, el simbol X indica el numero d'identificador que donem a la mota, ha de ser el mateix en les dues comanes.

    - Per últim l'aplicació del PC, ha estat dissenyada en java, per a la plataforma linux. Només cal copiar la carpeta adjunta "ALARMES" i ja estarà instal·lada.

2. Abans de l'execució haurem de activar el SerialForWarder de tinyOS, ja que l'aplicació del PC el necessita per a funcionar correctament, això ho farem de la següent manera, des de un terminal linux:

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

3. Executar l'aplicació de PC "ALARMES", per això executarem l'script Alarmes. Amb això obrirem l'aplicació gràfica de la central d'alarmes.
4. Per últim amb les motes programades, anirem encenent-les d'una en una.

7.3 Manual d'usuari

Més o menys el funcionament del sistema ja ha estat explicat en els punts anteriors d'aquest document, de totes maneres l'usuari disposa d'una pantalla d'ajuda que pot trobar al menú superior de l'aplicació, aquest menú conté la següent informació.

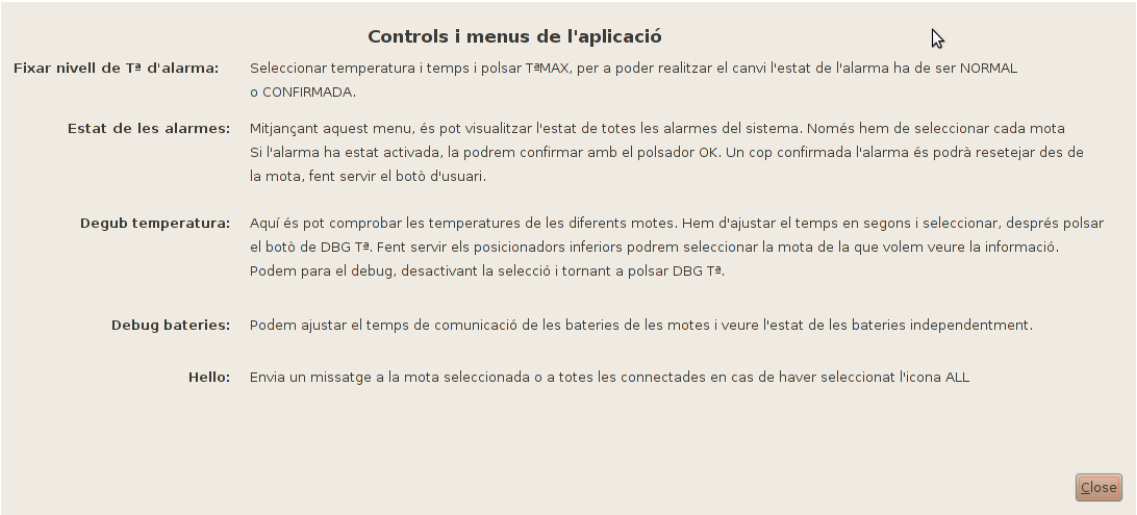

Figura 7.3.a: Pantalla d'ajuda de l'aplicació