Monitorització ambiental d'una sala de servidors mitjançant una xarxa de sensors sense fils

Enric Salvador García
“Enginyeria tècnica en informàtica de sistemes ”

Consultor: Jordi Bécares Ferrés

10 de juny del 2011

Agraïments

Al consultor Jordi Bécares per tota la seva ajuda i els bons consells donats durant l'elaboració d'aquest projecte.

Resum

L'objectiu d'aquest projecte és el treball, la experimentació i implementació de dos aplicacions bàsiques que exploten les possibilitats d'un sistema bàsic de dos sensors sense fils. Com el nom del projecte indica,, aquest sistema de sensors s'aplicarà a una sala de servidors, on el valor de la temperatura de la sala és una variable física crítica, pel correcte funcionament del maquinari present. No controlar el valor d'aquesta temperatura i en el cas de tenir una anomalia en el seu valor, podria suposar la averia del maquinari, dels servidors i el col·lapse de qualsevol empresa, on el seu funcionament depengués del sistema informàtic. Avui en dia la gran part de les mitjanes i grans empreses tenen sales de servidors, on la temperatura ha de ser controlada i aquest projecte utilitza una tecnologia de sensors sense fils, que es pot afegir com un sistema complementari més, del control ambiental de qualsevol sala de servidors.

Índex de continguts

Índex de figures

1.Introducció

1.1 Justificació

Normalment les empreses es gasten una gran part del seu pressupost en condicionar la sala de servidors amb diferents sistemes, tan de seguretat per a la restricció del accés malintencionat, com ambiental per assegurar una temperatura constant per l'òptim funcionament del maquinari, amb costosos aparells de refrigeració. Tan mateix, també s'assegura amb sistemes antiincendis, com extintors, armaris de material ignífug, alarmes antiincendis, etc.

El sistema que es presenta en aquest projecte és una aportació més per a la seguretat i control que es requereix en la sala de servidors d'una empresa. Aquest projecte parteix de la necessitat de presentar un sistema viable de programari i maquinari , per el control ambiental, principalment de dos variables físiques de qualsevol sala de servidors, que son la temperatura i la lluminositat. Encara que aquesta tasca sembli molt simple, les avantatges son clares doncs el sistema és sense fils, no requereix instal·lació i és molt senzill d'ampliar el nombre de motes, adaptant-se a les dimensions de la sala i a la quantitat de punts estratègics , on fa falta controlar aquestes variables.

1.2 Descripció

S'ha implementat una xarxa bàsica d'un parell de sensors sense fils amb els següents requisits bàsics:

- La principal funcionalitat és la monitorització de la temperatura.
- Altres funcionalitats son la monitorització de la lluminositat i el nivell de les bateries de la mota remota.
- Funcionalitat de poder configurar la freqüència de mostratge de les tres variables.
- Funcionalitats de configuració d'alarmes de temperatura i nivell de les bateries. Depèn del valor de les bateries el correcte funcionament de les lectures de temperatura i lluminositat.
- Funcionalitat d'aturada dels sensors, per l'estalvi de les bateries.
- Desenvolupament d'una interfície adient per a la comunicació amb les motes.

1.3 Objectius

Els diferents objectius s'han assolit gradualment a mida que s'ha anat avançant en el coneixement del llenguatge 'nesC' necessari per a la implementació del codi en la mota remota. S'esmenten a continuació els objectius assolits:

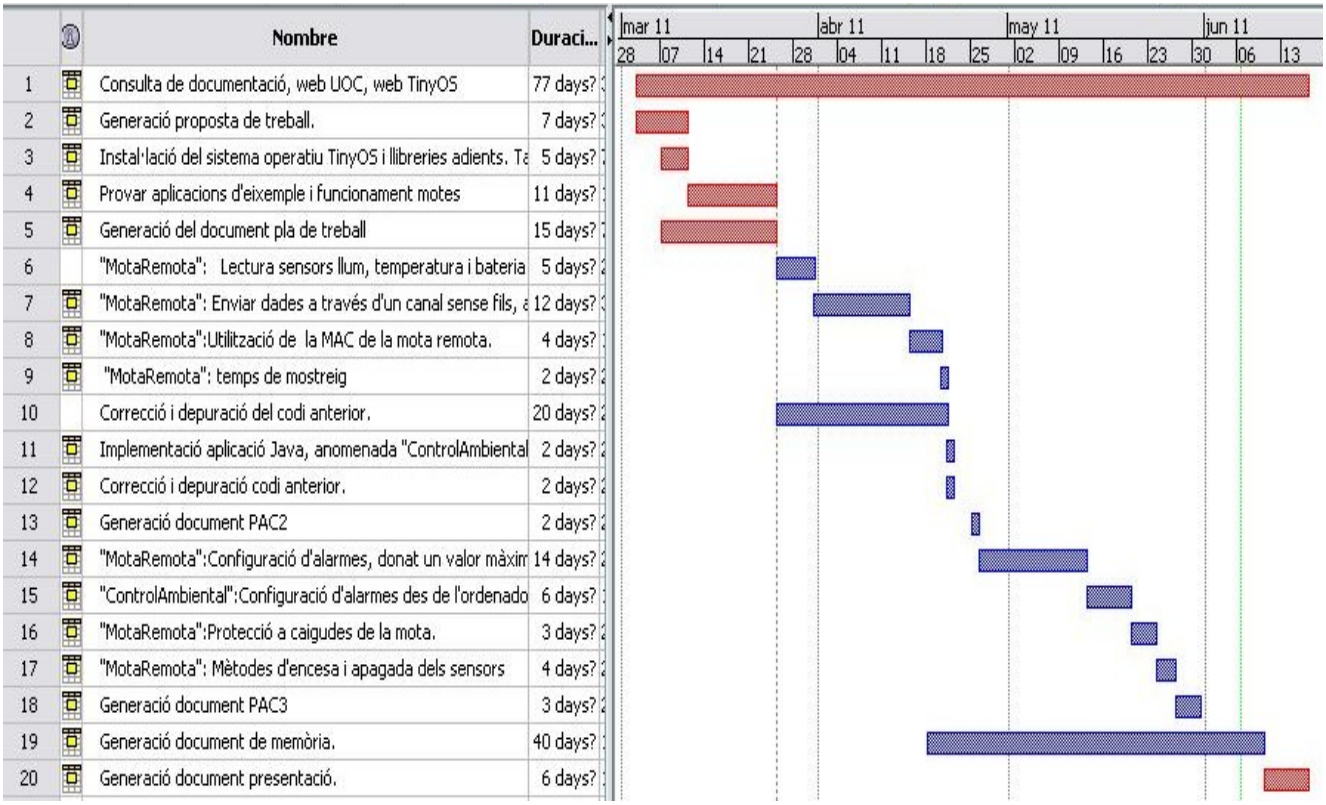
- Comunicació sense fils.
- Obtenció direcció mac de la mota remota.
- Monitorització de la temperatura ambiental.
- Monitorització del nivell de les bateries de la mota remota.
- Monitorització de la lluminositat ambiental.
- Configuració de l'alarma de temperatura.
- Configuració de l'alarma del nivell de les bateries.
- Notificació d'alarmes generades per sobrepassar un llindar.
- Aturada sensors.
- Interfície d'usuari basada en l'enviament de comandes a la mota remota.
- Implementació d'un sistema de persistència de dades amb un sistema gestor de base de dades, concretament amb 'MySql', per poder fer un anàlisi d'una quantitat significativa de lectures.
- Integració, dintre de la interfície de consola, d'una petita aplicació gràfica, que faciliti al usuari la interpretació de les dades recollides.

1.4 Enfocament

En la realització de les diferents fases del projecte s'ha seguit un mètode progressiu de resolució, primer de tot de les funcionalitats bàsiques proposades i després s'ha anat afegint funcionalitats complementaries. En tot moment s'ha consultat la bibliografia disponible en la web de Tinyos (apartat bibliografia), per implementar les diferents funcionalitats bàsiques necessàries en la mota remota com l'enviament de lectures dels sensors, rebre els paràmetres de activació i configuració, freqüència de mostreig, alarmes i aturada de sensors. El mateix mètode s'ha seguit per a la interfície per consola, enviament a la mota remota de paràmetres de configuració, definint noves classes amb nous tipus de missatges i rebuda de lectures dels sensors. Després d'assolides les funcionalitats bàsiques, aquestes s'han complementat amb altres funcionalitats com la persistència de dades.

1.5 Planificació

La planificació inicial proposta, amb el cronograma i els diagrames de Gant és la següent:



	Nombre	Duraci...
1	Consulta de documentació, web UOC, web TinyOS	77 days?
2	Generació proposta de treball.	7 days?
3	Instal·lació del sistema operatiu TinyOS i llibreries adients. Ta	5 days?
4	Provar aplicacions d'eixemple i funcionament motes	11 days?
5	Generació del document pla de treball	15 days?
6	"MotaRemota": Lectura sensors llum, temperatura i bateria	5 days?
7	"MotaRemota": Enviar dades a través d'un canal sense fils, a	12 days?
8	"MotaRemota":Utilització de la MAC de la mota remota.	4 days?
9	"MotaRemota": temps de mostreig	2 days?
10	Correcció i depuració del codi anterior.	20 days?
11	Implementació aplicació Java, anomenada "ControlAmbiental	2 days?
12	Correcció i depuració codi anterior.	2 days?
13	Generació document PAC2	2 days?
14	"MotaRemota":Configuració d'alarmes, donat un valor màxim	14 days?
15	"ControlAmbiental":Configuració d'alarmes des de l'ordenado	6 days?
16	"MotaRemota":Protecció a caigudes de la mota.	3 days?
17	"MotaRemota": Mètodes d'encesa i apagada dels sensors	4 days?
18	Generació document PAC3	3 days?
19	Generació document de memòria.	40 days?
20	Generació document presentació.	6 days?

Fig. nº 1: Cronograma i diagrama de Gant inicial.

Aquesta planificació correspon a la primera versió desenvolupada del projecte. Per a la versió final del mateix, es fa una reestructuració del programari resident a la mota remota, es a dir, es desenvolupen més components i també pel programari que correspon a la interfície d'usuari, implementant el seu funcionament mitjançant comandes i afegint funcionalitats de persistència de dades i gràfiques, que en un principi no s'havien planificat. La fase més problemàtica del projecte ha sigut la inicial, de instal·lació i configuració de tot l'entorn necessari pel correcte funcionament del sistema i l'aprenentatge del llenguatge de programació "**nesC**", per a la programació de les motes.

Finalment les tasques han quedat definides de la manera que a continuació es detallen:

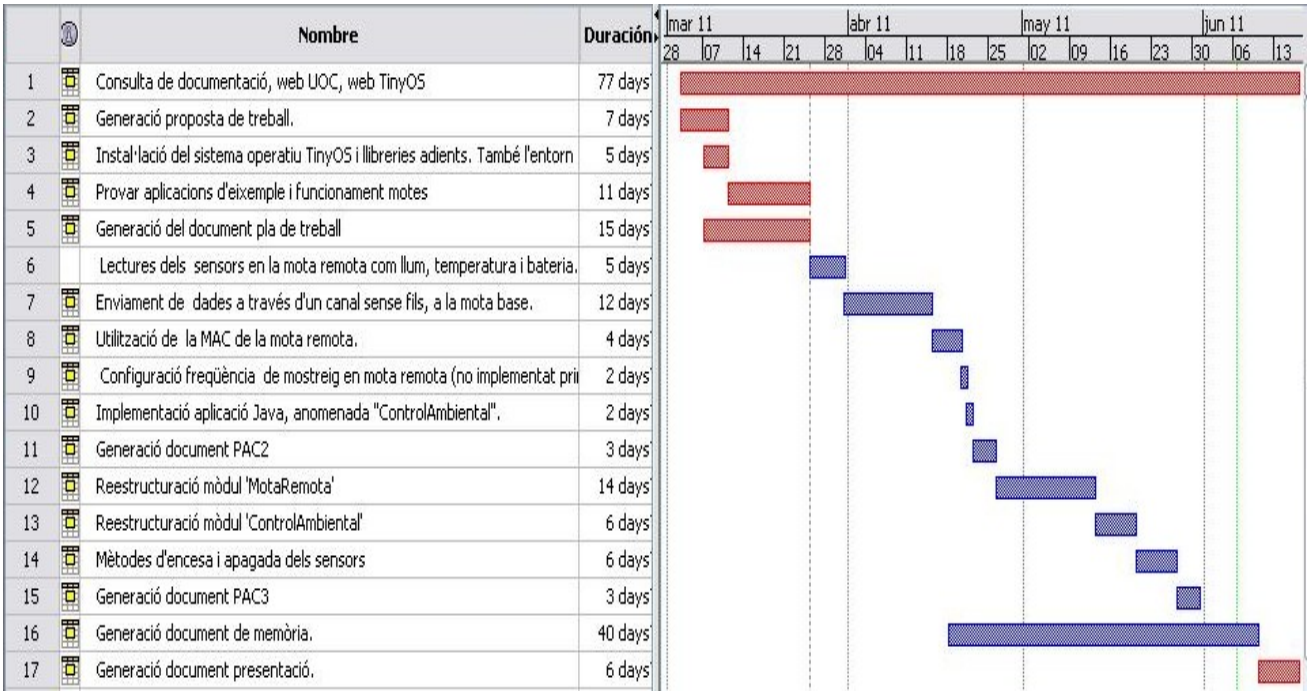
- **Consulta de la documentació de la UOC i de la web de TinyOS (03/03/11 a 17/06/11):** En el transcurs de tot el projecte les consultes de tota la documentació disponible ha sigut constant, per tant aquesta tasca ha estat sempre present.
- **Generació proposta de treball (03/03/11 a 11/03/11):** Es realitza un primer document amb la proposta de treball inicial, aquesta proposta contempla funcionalitats bàsiques ha implementar principalment.
- **Instal·lació del sistema operatiu Tinyos amb llibreries adients i entorn de desenvolupament 'Eclipse'** (**07/03/11 a 11/03/11**): Realització de les instal·lacions del sistema operatiu 'TinyOS', amb les seves llibreries i el entorn de desenvolupament 'Eclipse', amb l'entorn 'Yeti', necessari pel desenvolupament en llenguatge 'nesC' . Instal·lació de 'meshprog' com aplicació externa en l'Eclipse per a la gravació de les motes.
- **Provar aplicacions d'exemple i funcionament de les motes (11/03/11 a 25/03/11):** Entrada amb contacte amb l'entorn de desenvolupament i llenguatge 'nesC', aplicació externa, compilació motes, proves amb el funcionament de les motes i aplicacions complementàries pel correcte funcionament de tot l'entorn.
- **Generació del document Pla de Treball (07/03/11 a 25/03/11):** Es genera la primera aproximació de les tasques necessàries per implementar les funcionalitats bàsiques, per això es realitza el Pla de Treball.
- **Lectures dels sensors en la mota remota (25/03/11 a 31/03/11):** Estudi del mòdul 'MotaRemota', on implementem els components i interfícies encarregades de les lectures dels sensors de temperatura, lluminositat i bateries.
- **Enviament de dades a la mota base (31/03/11 a 15/04/11):** Estudi del mòdul 'MotaRemota' , on implementem els components i interfícies encarregades del enviament de les lectures dels sensors.
- **Utilització de la mac de la mota remota (15/04/11 a 20/04/11):** Estudi del mòdul 'MotaRemota', on implementem els components i interfícies encarregades d'enviar la mac a la mota base.
- **Configuració de la freqüència de mostreig en la mota remota (20/04/11 a 21/04/11):** No es va implementar en una primera versió del codi (PAC 2).
- **Implementació aplicació en Java 'ControlAmbiental' (21/04/11 a 22/04/11):** Implementació de la primera versió de la interfície mitjançant la consola, que ens permet la comunicació amb les motes.
- **Generació document PAC2 (22/04/11 a 26/04/11):** Realització i entrega de la primera versió del codi amb les funcionalitats implementades fins aquesta data.
- **Reestructuració mòdul 'MotaRemota' (26/04/11 a 13/05/11):** Reestructuració i implementació del mòdul, afegint la configuració de les freqüències de mostreig i alarmes a la mota remota.

- **Reestructuració mòdul 'ControlAmbiental' (13/05/11 a 20/05/11):** Reestructuració i implementació de tota la interfície per consola, afegint noves classes, noves comendes, persistència de dades i interfície gràfica.
- **Mètodes d'encesa i apagada dels sensors (20/05/11 a 27/05/11):** Implementació d'aturada de sensors, modificant mòduls 'ControlAmbiental' i 'MotaRemota', amb l'enviament de les comandes necessàries a la mota remota.
- **Generació PAC 3 (27/05/11 a 31/05/11): Depuració** de la versió definitiva del codi i generació del document adient.
- **Generació document de memòria (17/04/11 a 10/06/11):** Generació del document de memòria, amb les notes que s'ha anat prenent en el transcurs del projecte.
- **Generació document de presentació (10/06/11 a 17/06/11):** Realització del document de presentació del projecte.

Les versions finals del cronograma i el diagrama de Gant, queden de la següent manera:

	Nombre	Duración	Inicio	Terminado
1	Consulta de documentació, web UOC, web TinyOS	77 days?	3/03/11 ...	17/06/11 ...
2	Generació proposta de treball.	7 days?	3/03/11 ...	11/03/11 ...
3	Instal·lació del sistema operatiu TinyOS i llibreries adients. També l'entorn de de	5 days?	7/03/11 ...	11/03/11 ...
4	Provar aplicacions d'eixemple i funcionament motes	11 days?	11/03/11...	25/03/11 ...
5	Generació del document pla de treball	15 days?	7/03/11 ...	25/03/11 ...
6	Lectures dels sensors en la mota remota com llum, temperatura i bateria.	5 days?	25/03/11...	31/03/11 ...
7	Enviament de dades a través d'un canal sense fils, a la mota base.	12 days?	31/03/11...	15/04/11 ...
8	Utilització de la MAC de la mota remota.	4 days?	15/04/11...	20/04/11 ...
9	Configuració freqüència de mostreig en mota remota (no implementat primera	2 days?	20/04/11...	21/04/11 ...
10	Implementació aplicació Java, anomenada "ControlAmbiental".	2 days?	21/04/11...	22/04/11 ...
11	Generació document PAC2	3 days?	22/04/11...	26/04/11 ...
12	Reestructuració mòdul 'MotaRemota'	14 days?	26/04/11...	13/05/11 ...
13	Reestructuració mòdul 'ControlAmbiental'	6 days?	13/05/11...	20/05/11 ...
14	Mètodes d'encesa i apagada dels sensors	6 days?	20/05/11...	27/05/11 ...
15	Generació document PAC3	3 days?	27/05/11...	31/05/11 ...
16	Generació document de memòria.	40 days?	17/04/11...	10/06/11 ...
17	Generació document presentació.	6 days?	10/06/11...	17/06/11 ...

Fig. nº 2: Cronograma final.



	Nombre	Duración
1	Consulta de documentació, web UOC, web TinyOS	77 days
2	Generació proposta de treball.	7 days
3	Instal·lació del sistema operatiu TinyOS i llibreries adients. També l'entorn	5 days
4	Provar aplicacions d'eixemple i funcionament motes	11 days
5	Generació del document pla de treball	15 days
6	Lectures dels sensors en la mota remota com llum, temperatura i bateria.	5 days
7	Enviament de dades a través d'un canal sense fils, a la mota base.	12 days
8	Utilització de la MAC de la mota remota.	4 days
9	Configuració freqüència de mostreig en mota remota (no implementat pri	2 days
10	Implementació aplicació Java, anomenada "ControlAmbiental".	2 days
11	Generació document PAC2	3 days
12	Reestructuració mòdul 'MotaRemota'	14 days
13	Reestructuració mòdul 'ControlAmbiental'	6 days
14	Mètodes d'encesa i apagada dels sensors	6 days
15	Generació document PAC3	3 days
16	Generació document de memòria.	40 days
17	Generació document presentació.	6 days

Fig. nº 3: Cronograma de Gant final.

1.6 Recursos

Disposem principalment de tres tipus de recursos, de programari, de maquinari i web.

Maquinari:

Ordinador amb processador AMD Athlon 64 bits, i 2GB de memòria RAM i dues motes model ATZB-24-A2, amb microcontrolador Atmega 1281. En apartats posteriors es detallen les seves característiques tècniques.

Programari:

Tot el programari de desenvolupament resideix en l'ordinador, tenim l'entorn de desenvolupament en Java anomenat Eclipse versió Galileo, amb l'afegit Yeti, per desenvolupar amb nesC. Per poder compilar el codi Java hem instal·lat el jdk 1.6 i per compilar el codi en nesC, tenim el programari i les llibreries de TinyOS.

Per últim tenim la utilitat meshprog, per passar el codi compilat en nesC a les motes. Lògicament en les motes resideixen les fonts compilades en nesC.

Web:

Els recursos Web estan detallats a la bibliografia.

1.7 Productes obtinguts

D'aquest projecte han sortit dos productes. El mòdul anomenat 'MotaRemota' implementat en nesC, que és el codi font que compilat es grava a la mota remota i li dona les funcionalitats que es detallaran en el capitul d'anàlisi funcional de l'aplicació. Altre mòdul anomenat 'ControlAmbiental' i implementat en Java, que és la interfície per consola que ens permet la comunicació amb les motes, la persistència de les lectures rebudes i que també porta integrat un petit mòdul gràfic per la visualització de les dades. Existeixen dos utilitats necessàries pel correcte funcionament de l'aplicació que pertanyen a les llibreries i utilitats que ens ofereix TinyOS de forma gratuita, aquestes son el 'SerialForwarder' i el 'BaseStation'. El primer serveix de pont entre l'aplicació d'interfície i el port serie amb la mota base i el segon es el programari, que compilat, resideix en la mota base i ens permet enviar i rebre la comunicació amb la mota remota.

1.8 Breu descripció dels capítols posteriors

En els següents apartats de la memòria es fa una breu descripció del estat de la tecnologia de xarxes de sensors sense fils, per passar posteriorment a fer una descripció de les tecnologies utilitzades per a realització del projecte, sistema operatiu 'TinyOS' amb el llenguatge de programació 'nesC', necessari per a la programació de les motes. Protocol de comunicació utilitzat 'ZigBee' en les comunicacions entre les motes. També es farà una breu descripció del les models de motes utilitzats i les seves característiques tècniques. Posteriorment passarem al anàlisis funcional de l'aplicació desenvolupada en el projecte.

2.Antecedents

2.1 Estat de l'art

Les xarxes de sensors sense fils constitueixen actualment un dels camps d'investigació mes actius en el mon dels sistemes distribuïts. Els nodes son les unitats que composen les xarxes de sensors sense fils, on cada node compren un petit dispositiu anomenat mota que esta compost d'un microcontrolador, uns sensors i un transmissor per a la comunicació de les dades recollides pels sensors. Les motes tenen la funció de interactuar amb el medi físic mitjançant els sensors i comunicar les seves lectures. Son dispositius electrònics autònoms, de baix cost i pocs recursos, on l'estalvi d'energia es vital pel seu rendiment i baix manteniment.

Els orígens de les WSN (Wireless Sensor Network), son unes determinades aplicacions militars desenvolupades en els anys cinquanta per la USNavy amb el acrònim SOSUS (Sound Survelliance System), es a dir sistema de vigilància sònica que comprenia una serie de sensors submergits en l'Atlàntic per a la detecció de submarins d'origen rus en la època de la guerra freda entre EEUU i la URS.

Les principals elements que constitueixen una xarxa de sensors sense fils son:

- **Sensors**: Existeixen de molts tipus i tecnologies i la seva funció és detectar la informació del medi on es troben i convertir-la en una senyal elèctrica.
- **Nodes de Sensor**: Prenen les dades del sensor, mitjançant les seves portes, que converteixen la senyal analògica a digital i la envien a l'estació base.
- **Gateway**: Elements per a la interconnexió entre la xarxa de sensors i una xarxa tcp/ip.
- **Estació base**: Node recollidor de dades.
- **Xarxa sense fils**: Normalment basada en el protocol de comunicació 802.15.4.ZigBee és una norma establerta per un conjunt de varies empreses fabricants de semiconductors amb l'objectiu de desenvolupar i evolucionar les tecnologies sense fils de baix cost, com les xarxes de sensors sense fils.

Les principals característiques de les xarxes sense fils son:

- Son xarxes 'ad-hoc', es a dir sense infraestructura de xarxa.
- Xarxes formades per milers de petits dispositius de baix cost.
- S'instal·len en llocs amb difícil accés.
- Els nodes son autònoms, pensats per tenir poc manteniment.
- Els nodes son dispositius molt restringits en potència de comput, emmagatzematge i comunicació.
- Nodes amb baix consum.
- Els nodes poden ser mòbils amb topologies de xarxa dinàmiques.

Les aplicacions de les xarxes de sensors sense fils son nombroses, algunes de les quals comentem a continuació:

- Aplicacions mediambientals, amb una monitorització de variables mediambientals no intrusiva, sense necessitat de infraestructures.
- Aplicacions en domòtica, automatització de tasques pel confort i seguretat en habitatges.
- Aplicacions en monitorització d'infraestructures per controlar el desgast del material.
- Aplicacions sanitàries pel diagnòstic de pacients .
- Aplicacions logístiques pel control d'equipatges i contenidors.

Existeix una llarga llista de nodes de sensors sense fils, comercials i també d'investigació, com exemple tenim la plataforma comercial de la empresa 'Crossbow' , que ha tret molts models al llarg dels anys, esmentem alguns:

- WeC (1998) : Incorporava un microcontolador 'AT90LS8535' de Atmega amb 8Kb.
- René (2000) i Dot(2001): Incorporaven un microcontrolador 'Atmega163' amb 16 Kb.
- Mica(2001) i Mica 2 (2002): Incorporaven un micro 'Atmega128' amb 128 Kb.
- TelosB(2004-2006): Incorpora micro TIMSP430 de 'Texas Instrument' amb 10 Kb de memòria RAM. Utilitza sistema operatiu de programari lliure com és 'TinyOS' i te unes característiques tècniques molt semblants als models utilitzats en aquest projecte.

2.1.1. Motes utilitzades

Les motes utilitzades en aquest projecte son unes petites plaques de circuit imprès per les dues cares, on s'han soldat diversos tipus de sensors, temperatura, lluminositat, efecte Hall, un compartiment per dos piles AA de 1,5 volts, un connector USB i el nucli principal esta format per un mòdul ATZB-24-A2, que conté al microcontrolador 'Atmega 1281' més el transmissor 'AT86RF230'.
Presentem a continuació una fotografia de la mota anomenada COU24, amb les seves parts indicades:

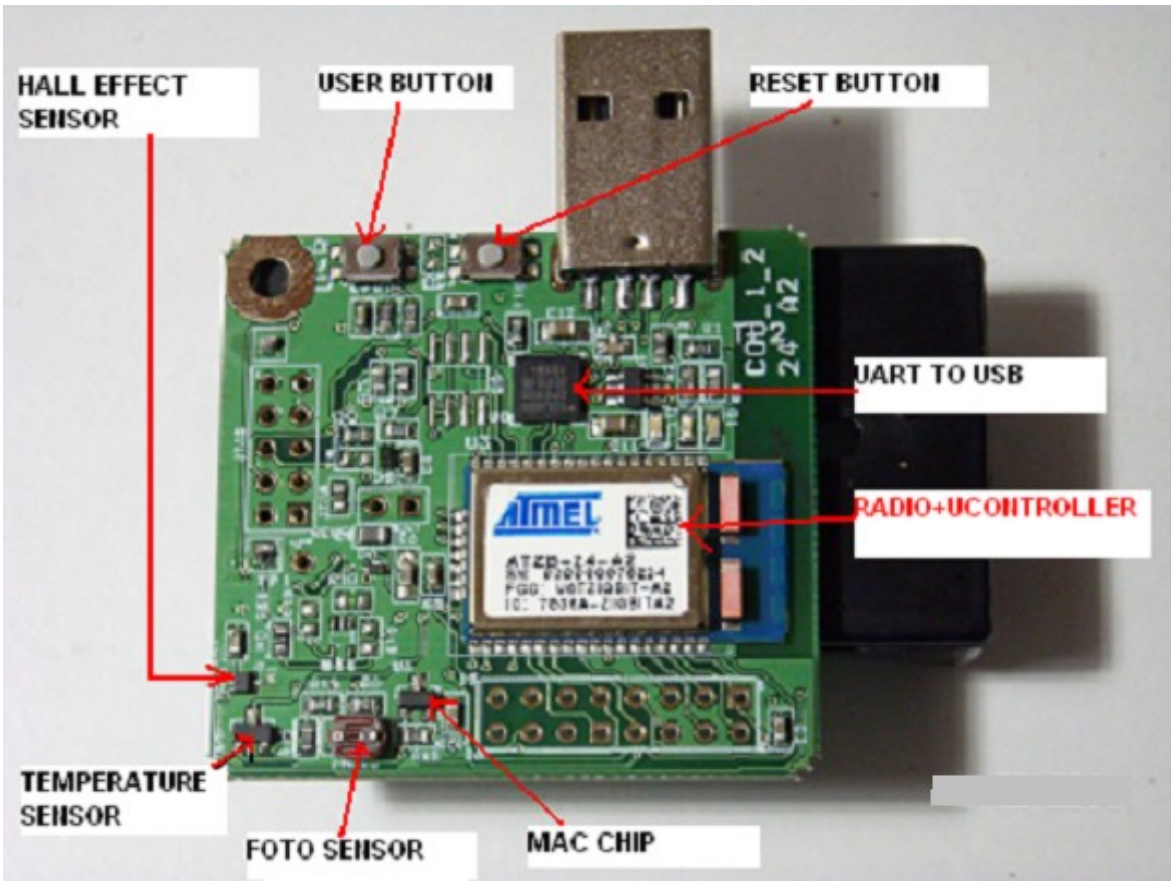


Fig. nº 4: Motes utilitzades en el projecte (COU24).

Com hem esmentat, el mòdul principal de la mota conté el microcontrolador i el transmissor i té unes característiques tècniques molt adients per aquests tipus de dispositius. A continuació passem a mostrar el diagrama de blocs d'aquest mòdul principal i les seves característiques, que s'ha consultat en el 'Datasheet' del fabricant:

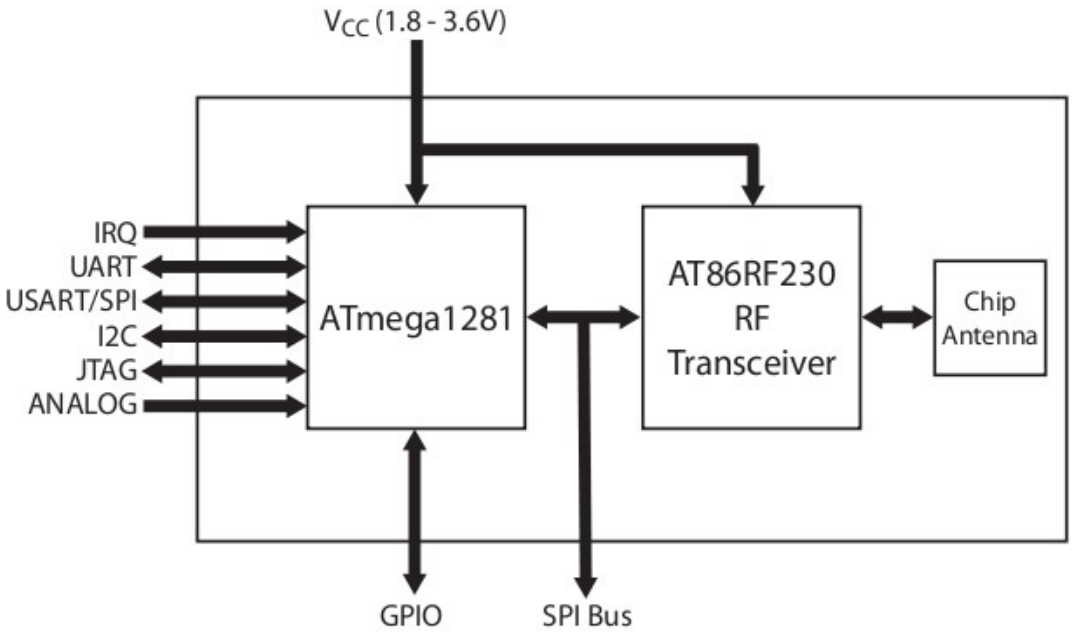


Fig. nº5: Diagrama de blocs del mòdul ATZB-24-A2

2.1.1.1. Microcontrolador

L'Atmega 1281 és un microcontrolador de baix consum, de tipus CMOS de 8 bits basat en l'arquitectura RISC (Reduced Instruction Set Computer) de la casa 'Atmel', familia amb nucli AVR. Posseeix una memòria Flash de 128 Kb, una EEPROM (Electrically- Erasable Programmable Read-Only Memory) de 4 Kb, una memòria SRAM (Static Random Access Memory) de 8 Kb. Amb 32 registres de propòsit general, 6 Timers/Counters i 4 transceptors universals de comunicació sèrie (USART). Una interfície de comunicació sèrie de dos cables (Two Wires Interface, TWI), un port sèrie SPI (Serial Peripheral Interface), un interfície de test JTAG, que compleix els requisits del estàndard 'IEEE 1149.1' , que s'utilitza per a la programació i depuració 'on chip' dels microcontroladors. També conté un convertidor analògic digital de 10 bits amb entrades úniques, diferents i amb una amplificació programable.

A continuació presentem el diagrama de blocs del microcontrolador:

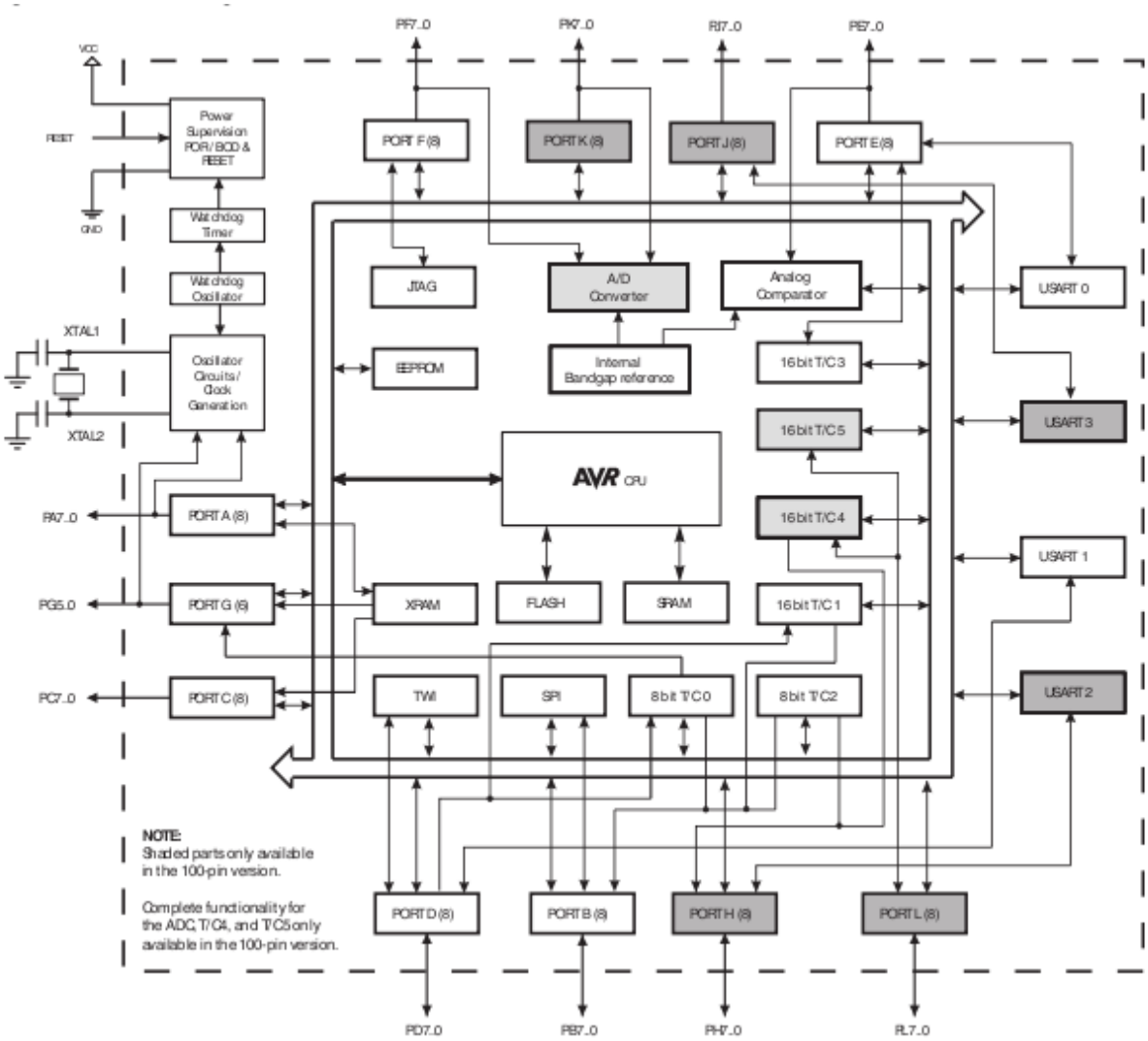


Fig. nº 6: Diagrama de blocs del Atmega 1281.

2.1.1.2. Transceptor

El transceptor AT86RF230, és una ràdio transceptor de baix consum amb alta sensibilitat (-101 dBm), i una distància de llargada de aproximadament 30 metros. Treballa en la banda lliure de radiofreqüència ISM (Industrial-Scientific-Medical) a 2.4 Ghz. També dir que te 16 canals de comunicació i opera amb el protocol estàndard de comunicació IEEE 802.15.4., que pertany a l'especificació d'un conjunt de protocols “**ZigBee**”.

A continuació presentem el diagrama de blocs del transceptor:

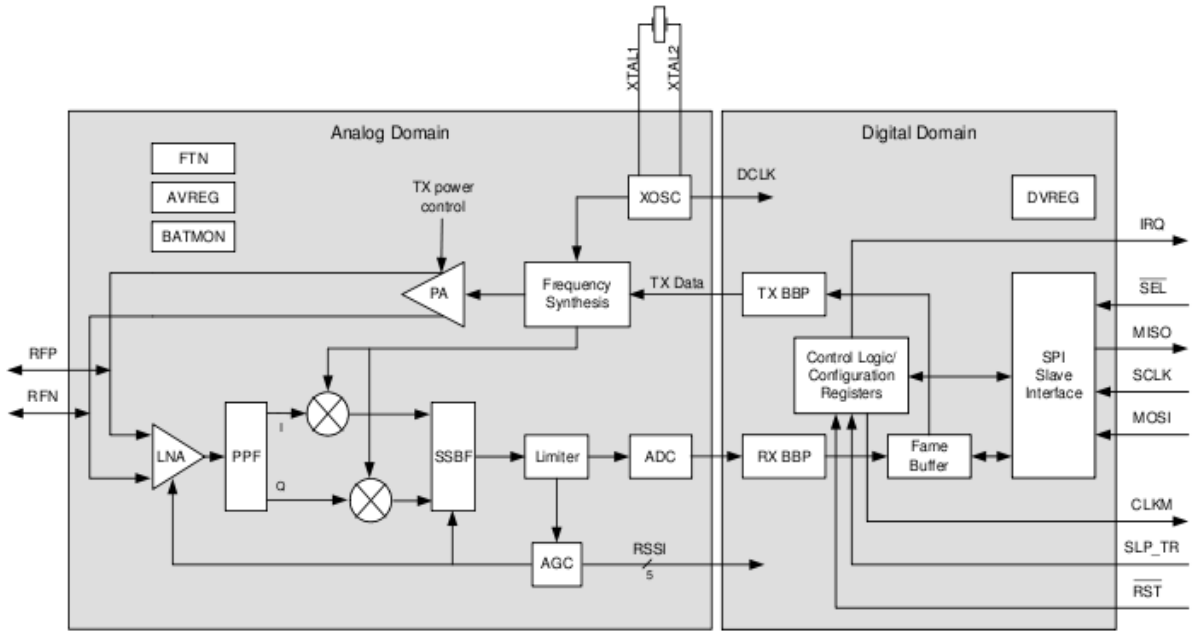


Fig.nº7: Diagrama de blocs del transceptor AT86RF230.

2.1.1.3. Sensors

En les motes hi ha integrats tres sensors, el de temperatura, lluminositat i el d'efecte Hall. En aquest projecte utilitzem només el sensor de temperatura i el de lluminositat.

- **Sensor de temperatura MCP9700A:** El seu funcionament es basa en una resistència que varia amb la temperatura (Thermistor). Converteix la lectura de temperatura en una senyal de voltatge analògic en mili volts. El rang de temperatures detectable és de -40ºC a +150ºC i la precisió de ±2ºC en el rang de 0ºC a +70ºC. Consumeix 6uA. No requereix cap circuit afegit i la sortida Vout de voltatge del sensor es pot connectar directament a la entrada ADC (Analogic Digital Converter) del microcontrolador 'Atmega 1281'. El coeficient de temperatura esta ajustat per proveir 1ºC/bit de resolució, amb un convertidor d'analògic a digital de 8 bits i amb un voltatge de referència de 2.5 volts.

En aquest gràfic es veu la relació de la sortida en volts amb la temperatura ambient:

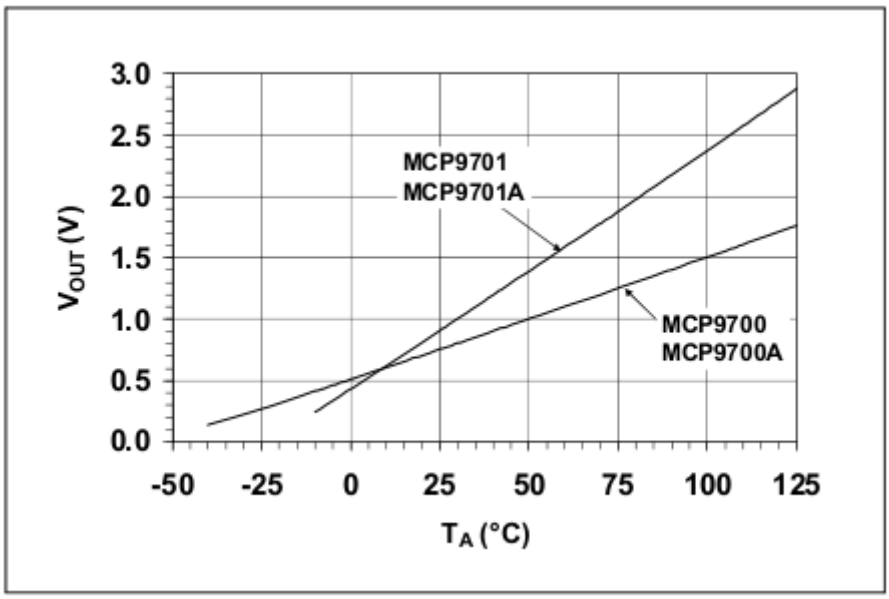


Fig.nº 8: Gràfic sensor temperatura.

- **Sensor de lluminositat PDV-P9003-1:** El seu funcionament es basa en una cel·la foto conductiva, es a dir la seva resistència varia amb la llum que incideix en la cel·la. Aquest gràfic reflexa aquesta variabilitat:

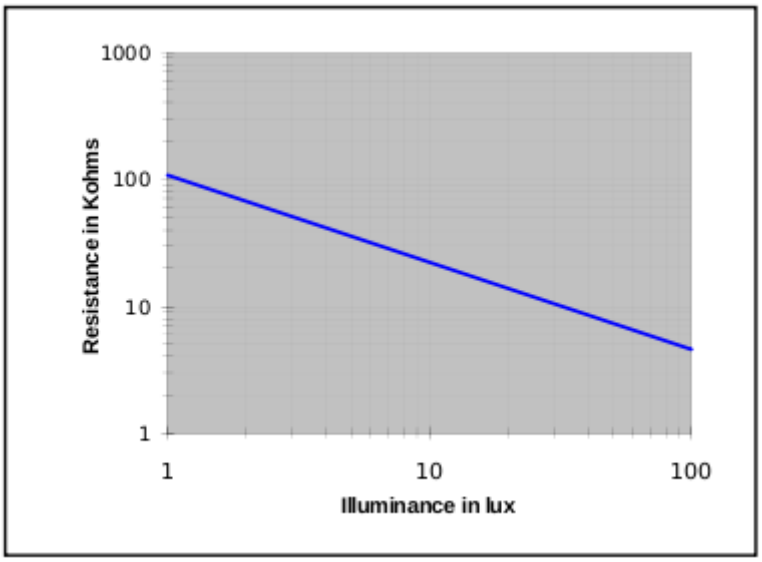


Fig.nº 9: Gràfic sensor lluminositat.

2.1.1.4 Protocol de comunicació

"**ZigBee**" és el nom de l'especificació d'un conjunt de protocols d'alt nivell de comunicació sense fils, per a la utilització de radiodifusió digital de baix consum, basada en el estàndard de comunicació IEEE 802.15.4., de xarxes sense fils d'àrea personal. El seu objectiu son les aplicacions de sistemes encastats, que requereixen comunicacions segures amb baixa taxa de transferència d'enviament de dades i estalvi d'energia de les bateries. Les característiques més importants son:

- Baix consum energètic.
- Topologia de xarxa en malla.
- Fàcil integració, possibilitat de fabricar nodes amb pocs components electrònics.

Es defineixen tres tipus de dispositius "**ZigBee**", depenen de la funció que fan dins la xarxa:

- Coordinador "**ZigBee**"(ZigBee coordinator, ZC): És el dispositiu més complert i ha d'haver un a la xarxa. S'encarrega de controlar la xarxa i els camins que han de seguir els dispositius per connectar-se.
- Encaminador "**ZigBee**"(ZigBee router, ZR): S'encarrega de la interconnexió de dispositius dins la topologia de la xarxa. Ofereix un nivell d'aplicació per a l'execució de codi d'usuari.
- Dispositiu final (ZigBee end device ZED): Posseeix la funcionalitat de comunicar-se amb el node pare (node coordinador o encaminador), però no envia informació a altres nodes.

Existeix altre classificació, si ens basem en la funcionalitat del dispositiu:

- Dispositiu de funcionalitat complerta (FFD): És el node actiu amb la capacitat de rebre missatges en el format 802.15.4. Pot realitzar tasques d'encaminador o coordinador "**ZigBee**", gracies a la seva capacitat de memòria i processament. També es pot utilitzar com a dispositiu de xarxa i interfase amb l'usuari.
- Dispositiu de funcionalitat reduïda (RFD): És el node passiu i te les seves capacitats i funcionalitats reduïdes amb l'objectiu d'aconseguir un baix cost i baix consum. Bàsicament son els sensors o activadors de la xarxa.

"**ZigBee**" permet tres tipus de tipologies de xarxa:

- Topologia en estrella: El node coordinador ocupa la part central.
- Topologia en arbre: El node coordinador és l'arrel de l'arbre.
- Topologia en malla: És la més interessant i permet el restabliment de la connexió dels nodes si, en un moment donat un node de la xarxa falla.

2.1.2. Sistema operatiu TinyOS

El sistema operatiu TinyOS poseeix el llenguatge de programació 'nesC', que s'utilitza per implementar les aplicacions que seran executades en les motes. És un llenguatge de programació que combina dos aspectes de la orientació a objectes que son la orientació a components i la orientació a esdeveniments. La orientació a components proveeix els components primitius, que proporciona el propi sistema operatiu i els components complexes que son externs. Aquests components proporcionen unes interfases que el programador utilitza per crear els seus propis components, aquest procés s'anomena '**wiring**'. En quant a la orientació a esdeveniments , la programació no es seqüencial sinó basada en l'execució d'un esdeveniment, es a dir, el codi s'executarà quan s'executi l'esdeveniment.

Cada component que el programador crea conté dos fitxers amb la extensió '.nc', un de nom 'nom_componentP.nc', amb les seccions 'Module' i 'Implementation', i un altre 'nom_componentC.nc' amb les seccions 'Configuration' i 'Implementation'. En la secció 'Implementation' del fitxer 'nom_componentC.nc' es defineixen les connexions que hi ha entre els diferents components que utilitza l'aplicació, en canvi en la secció 'Configuration' d'aquest mateix fitxer es definiran components creats mitjançant altres components. En la secció 'Module' del fitxer 'nom_componentP.nc', es defineixen les interfases que s'utilitzen en el component i en la secció 'Implementation' tenim el codi de l'aplicació que definirà el comportament de la mota. La estructura d'aquesta secció és molt semblant a la d'un programa en llenguatge C, però adaptat a la programació orientada a objectes (components) i esdeveniments i esta format per les variables de l'aplicació , les funcions que s'han d'implementar, com a conseqüència de les interfases que proporcionem i els esdeveniments que s'han d'implementar degut a les interfases utilitzades.

Tipus principals de dades:

- **uint16_t** : enter sense signe de 16 bits.
- **uint8_t**: enter sense signe de 8 bits.
- **result_t**: s'utilitza com a retorn d'una funció , amb valors 'SUCCES' o 'FAIL'.
- **bool**: valors 'TRUE' o 'FALSE'.

Tipus de funcions:

- **Command:** funció que s'executa de forma sincrònica, la manera de cridar-les es **call** interfase. nomFunció.
- **Task:** son tasques que s'executen concurrentment en l'aplicació de forma semblant als 'threads' del llenguatge Java. La forma de cridar-les és **post** interfase.nomTasca.
- **Event:** son funcions que s'executen quan s'aixeca una senyal del sistema, segueixen el paradigma de la programació orientada a esdeveniments. La forma de invocar-les és **signal** interfase.nomEsdeveniment. Poden portar paràmetres si la interfase és parametritzada **signal** interfase.nomEsdeveniment[paràmetre].

Aquestes tipus de funcions poden portar endavant la paraula reservada **async** per indicar que la seva execució serà de manera asíncrona.
En l'anàlisi funcional de l'aplicació desenvolupada, s'esmentaran els components utilitzats i les interfases utilitzades del sistema operatiu 'TinyOS'.

Fem un resum de les principals característiques d'aquest sistema operatiu:

- Arquitectura basada en components.
- Adaptat als recursos limitats de les motes com son el processament restringit, energia limitada, emmagatzemament i amplada de banda.
- Controlat per esdeveniments que implementa l'usuari.
- Implementació en el llenguatge nesC.
- Petit nucli executable (footprint), del sistema operatiu quan es compila una aplicació, es a dir l'empremta del S.O. és molt petit (400bytes).
- Les interfases son vi-direccionals, ja que proporcionen funcions implementades pel component i esdeveniments implementats per l'usuari.

2.2 Estudi de mercat

A continuació farem un recorregut pels principals fabricants i desenvolupadors de tecnologies per a xarxes de sensors sense fils:

- "**CROSSBOW**": La tecnologia desenvolupada per aquest fabricant ha estat a la avantguarda del mon dels sensors intel·ligents durant més d'una dècada i ha venut cents de milers d'aquests sensors a mes de 4000 clients al llarg del mon. Avui en dia és el lider en quant a tecnologies de sensors sense fils. Desenvolupa plataformes de maquinari i programari que donen solucions a les xarxes de sensors sense fils. Entre els seus productes podem trobar les plataformes, **Mica**, **Mica2**, **Micaz**, **Mica2dot**, **telos**, **telosb**, **Iris** i **Imote2**.
- "**SENTILLA**": Empresa dedicada a les xarxes de sensors sense fils, en col·laboració amb la universitat de Berkeley, ha desenvolupat la plataforma "**Tmote Sky**", compatible amb el sistema operatiu "**TinyOS**". El seu fundador va ser Joseph Polastre, un antic doctorant d'un grup de treball d'aquesta universitat, que en un principi formà l'empresa "**MOTEIV**", desenvolupant les plataformes de maquinari "**Tmote Sky**" i "**Tmote Invent**".
- "**SHOCKFISH**": Aquesta societat anònima ha desenvolupat la plataforma "**Tiny Node**", per aplicacions industrials i una de les seves funcions és servir de pont entre el mon laboral i el mon universitari d'investigació dintre de les tecnologies de sensors sense fils. La plataforma desenvolupada "**Tiny Node**", serveix tant per aplicacions industrials com per aplicacions d'experimentació acadèmica.
- "**Btnode**": Els mòduls que aquest fabricant ha desenvolupat "**Btnodes**", ha sigut amb col·laboració del "**ETH Zurich** " conjuntament amb "**Computer Engineering and Networks Laboratory (TIK)**" i el "**Research Group for Distributed Systems**" també de Zurich.
- "**EMBER**": Aquesta empresa és un dels promotors de la "**ZigBee Alliance**" i les seves solucions compleixen el estàndard IEEE 802.15.4. La tecnologia que desenvolupa esta basada en el protocol "**ZigBee**" i és l'adequada per aplicacions de xarxes sense fils, de baix consum i automatitzades.
- "**SUN Microsystems**": Ha desenvolupat la plataforma "**Sun SPOT**", basada en l'estàndard IEEE 802.15.4. i que funciona sota la màquina virtual "**Java Squawk**".

- "**Nano-RK**": Aquesta empresa ha desenvolupat la plataforma de baix cost i baix consum "**FireFly**" de nodes sense fils, amb la idea de donar suport a aplicacions en temps real.

A continuació presentem una taula on tenim les principals característiques de les plataformes esmentades en aquest apartat:

Nombre	Casa	Microcontrolador	Transmisor	Memoria Programas y datos	Memoria externa	Programación	Comentarios
EM250	Ember	12MHz XAP2b 16-bit	2.4GHz IEEE 802.15.4 Compliant Transciever	5kB RAM	128kB Flash	C	
Mica	Crossbow	Atmel ATMEGA103 4 MHz 8-bit CPU	RFM TR1000 radio 50 kbit/s	128+4K RAM	512K Flash	nesC Programming	TinyOS Support
Mica2	Crossbow	ATMEGA 128L	Chipcon 868/916 MHz	4K RAM	128K Flash		TinyOS, SOS and MantisOS Support
Mica2Dot	Crossbow	ATMEGA 128		4K RAM	128K Flash		
MicaZ	Crossbow	ATMEGA 128	TI CC2420 802.15.4/ZigBee compliant radio	4K RAM	128K Flash	nesC	TinyOS, SOS, MantisOS and Nano-RK Support
Telos	Crossbow	Motorola HCS08		4K RAM			
TelosB	Crossbow	Texas Instruments MSP430 microcontroller	250 kbit/s 2.4 GHz IEEE 802.15.4 Chipcon Wireless Transceiver	10k RAM	48k Flash		Contiki, TinyOS, SOS and MantisOS Support
T-Mote Sky	Sentilla	Texas Instruments MSP430 microcontroller	250 kbit/s 2.4 GHz IEEE 802.15.4 Chipcon Wireless Transceiver	10k RAM	48k Flash		Contiki, TinyOS, SOS and MantisOS Support
IMote	Intel	ARM core 12 MHz	Bluetooth with the range of 30 m	64K SRAM	512K Flash		TinyOS Support
IMote 1.0	Intel	ARM 7TDMI 12-48 MHz	Bluetooth with the range of 30 m	64K SRAM	512K Flash		TinyOS Support
IMote 2.0	Crossbow	Marvel PXA271 ARM 11-400 MHz	TI CC2420 802.15.4/ZigBee compliant radio	32MB SRAM	32MB Flash		Microsoft .NET Micro, Linux, TinyOS Support
Iris	Crossbow	ATmega1281	Atmel AT86RF230 802.15.4/ZigBee compliant radio	8K RAM	128K Flash		nesC TinyOS, MoteWorks Support
SunSPOT	Sun Microsystems	ARM 920T	802.15.4	512K RAM	4 MB Flash		Java Squawk J2ME Virtual Machine
FireFly	Nano-RK	Atmel ATmega 1281	Chipcon CC2420	8K RAM	128K FLASH ROM, 4K EEPROM	C Programming	Nano-RK RTOS Support
BTNode rev.3	BTNode	Atmel ATmega 128L	Bluetooth subsystem: Zeevo ZV4002, supporting AFH/SFH	64+180 Kbyte RAM	128 Kbyte FLASH ROM, 4 Kbyte EEPROM	Standard C Programming	TinyOS Support
eyesIFXv2.1	BTNode	Texas Instruments MSP430F1611	Infineon TDA5250	10 KB RAM	48 KB FLASH		

Fig. nº 10 : Algunes de les plataformes existents en el mercat.

3. Descripció funcional

3.1 Sistema total

L'objectiu d'aquest projecte és el treball, la experimentació i implementació de dos aplicacions bàsiques que exploten les possibilitats d'un sistema bàsic de dos sensors sense fils. Com el nom del projecte indica, aquest sistema de sensors s'aplicarà a una sala de servidors, on el valor de la temperatura de la sala és una variable física crítica, pel correcte funcionament del maquinari present. No controlar el valor d'aquesta temperatura i en el cas de tenir una anomalia en el seu valor, podria suposar la averia del maquinari, dels servidors i el col·lapse de qualsevol empresa, on el seu funcionament depengués del sistema informàtic. Avui en dia la gran part de les mitjanes i grans empreses tenen sales de servidors, on la temperatura ha de ser controlada.

S'ha desenvolupat una aplicació anomenada 'ControlAmbiental' i implementada en Java, que s'encarrega d'enviar els paràmetres de configuració a la mota remota i al mateix temps, implementa una persistència de dades amb les lectures rebudes dels sensors. En l'ordinador connectat amb la mota base s'ha instal·lat un servidor de gestor de base de dades 'MySql', on es troben tres taules per a la persistència dels tres tipus de lectures que es poden rebre, que son la lectura de la temperatura, lluminositat i el valor en volts de les bateries de la mota remota. Tan mateix, aquesta aplicació és la interfície per consola que ens permet parlar amb les motes i també tenir un petit manteniment de la base de dades. El sistema de persistència de dades ens permet, a posteriori, poder fer un anàlisi massiu de les dades rebudes, doncs serveix de ben poc rebre una lectura puntual de la temperatura, en un determinat moment, si no podem emmagatzemar-la.

Dintre de l'aplicació de consola 'ControlAmbiental', s'ha integrat una petita aplicació gràfica desenvolupada en 'Gambas2', que es connecta a la base de dades i ens mostra els llistats de les lectures d'una manera gràfica i més propera al usuari.

L'altre aplicació desenvolupada s'anomena 'MotaRemota', s'ha implementat en 'nesC' i es el programari encarregat de fer funcionar la mota remota. Fa que aquesta mota ens envii les lectures dels seus sensors amb la freqüència de mostreig que li diguem. També li podem configurar dos tipus d'alarmes, de temperatura i de bateria, de manera que si aquests valors surten d'un determinant rang establert, ens envii les lectures pertinents. Finalment podem aturar el tipus de sensor del qual no ens interessi rebre més lectures.

A continuació es mostra un diagrama de blocs del projecte, que resum les explicacions esmentades:

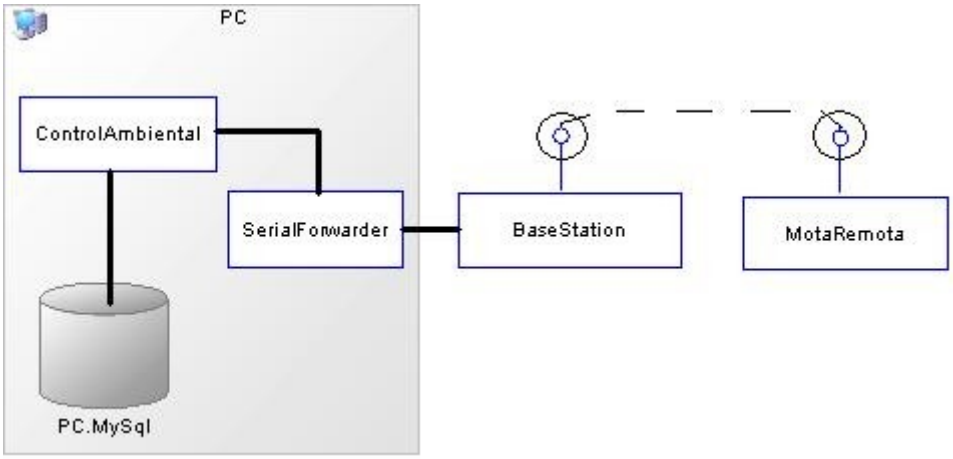


Fig.nº 11: Diagrama de blocs del sistema.

Tenim dos aplicacions que son proveïdes per les pròpies utilitats de TinyOS i al mateix temps son necessàries pel correcte funcionament de l'aplicació. El 'SerialForwarder' és una utilitat desenvolupada en Java, que serveix de pont entre l'aplicació 'ControlAmbiental' i la mota base, mitjançant la connexió d'un socket TCP/IP, per on passen els paquets entrants i sortints. L'aplicació 'BaseStation' resideix en la mota base, esta desenvolupat en 'nesC' i fa que aquesta mota rebi i envii paquets a la mota remota.
En el cas d'aplicar aquest projecte a un cas real, dins una sala de servidors pel control, principalment de la temperatura, que és la variable física més crítica pel correcte funcionament dels servidors, la xarxa de sensors sense fils a instal·lar podria ser la següent:

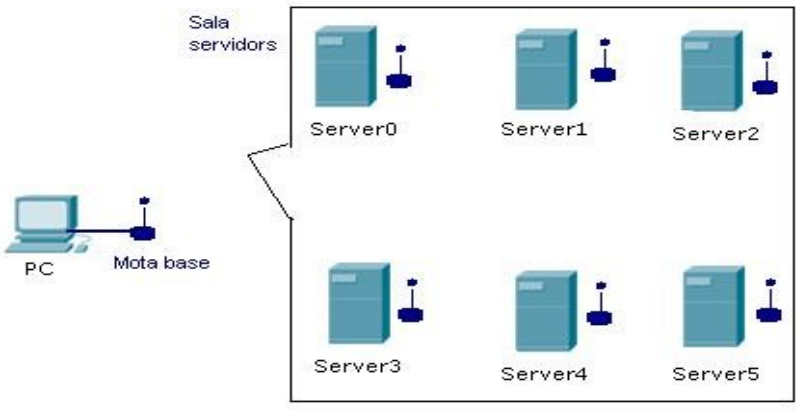


Fig.nº 12: Sala servidors amb xarxa de sensors sense fils.

Si suposem una sala amb sis servidors repartits en un cert espai de la sala, no junts, podríem controlar independentment la temperatura al voltant de cada servidor amb la instal·lació d'una mota base a un ordinador fora de la sala i sis motes remotes al costat de cada servidor. En un principi s'hauria d'adaptar les aplicacions desenvolupades en el projecte, per al funcionament amb més d'una mota remota i provar la comunicació entre la mota base i les motes remotes. Aquest muntatge es considera factible, i el cost de extrapolar les aplicacions per a la comunicació amb sis motes remotes no seria molt costos.

3.2 Interfície d'usuari

La interfície d'usuari correspon a l'aplicació desenvolupada "**ControlAmbiental**" i és on interaccionen l'usuari amb la mota remota mitjançant la introducció via consola d'una sèrie de comandes. Existeixen dos tipus principals de comandes:

- Comandes que interactuen amb la mota remota i deixen l'aplicació en mode escolta, en aquest cas, es rebran lectures del sensor de la mota que estigui activat.
- Comandes que interactuen amb el sistema gestor de base de dades, i tenen la funcionalitat de mostrar les lectures rebudes dels sensors de la mota remota.

Mostrem a continuació un diagrama de casos d'us de l'aplicació. Aquests casos depenen de la comanda introduïda per l'usuari:

Fig.nº 13: Diagrama de casos d'us de l'aplicació "ControlAmbiental".

Mostrem una taula que ens indica la relació entre les comandes que pertanyen al primer grup, es a dir, que interactuen amb la mota remota, amb el corresponen cas d'us:

COMANDA	**CAS D'US**
MAC	Lectura mac
TEMPERATURA	Lectures temperatura
BATERIA	Lectures bateries
LLUM	Lectures lluminositat
ATURAR TEMPERATURA	Aturar sensor temperatura
ATURAR LLUM	Aturar sensor lluminositat
ATURAR BATERIA	Aturar lectures bateries
ALARMA TEMPERATURA	Alarma temperatura
ALARMA BATERIA	Alarma bateries

A continuació mostrem la taula que relaciona el segon grup de tipus de comandes, les que interactuen amb el sistema gestor de base de dades, amb el corresponen cas d'us:

COMANDA	**CAS D'US**
TAULA TEMPERATURA	Registres temperatura
TAULA BATERIA	Registres bateries
TAULA LLUMINOSITAT	Registres lluminositat
ESBORRAR TEMPERATURA	Esborra registres temperatura
ESBORRAR BATERIA	Esborra registres bateries
ESBORRAR LLUMINOSITAT	Esborra registres lluminositat
APLIC GRAF	Aplicació gràfica visualització registres.

En el cas de les comandes del primer grup, en el diagrama d'activitat mostrem com interactua l'usuari amb la mota remota. Primerament l'usuari introdueix una comanda d'activació del sensor, d'un dels tres tipus existents, llum,temperatura o bateries, amb una determinada freqüència de mostratge. La mota remota activa el sensor corresponent i entra en un bucle de lectura de sensor i enviament de lectura a la mota base . L'aplicació “**SerialForwarder**” escolta el port serie mitjançant la connexió al socket i la lectura rebuda és mostrada per l'aplicació de consola. Mentre l'usuari no introdueixi una comanda d'aturada de sensor la mota remota continuarà enviant lectures. Per sortir del mode d'escolta de l'aplicació de consola s'ha d'introduir “Crtl-z”. Una vegada hem sortit del mode d'escolta, podem introduir altre comanda, que pot ser per activar altre sensor, aturar sensor o configurar les alarmes dels sensors.

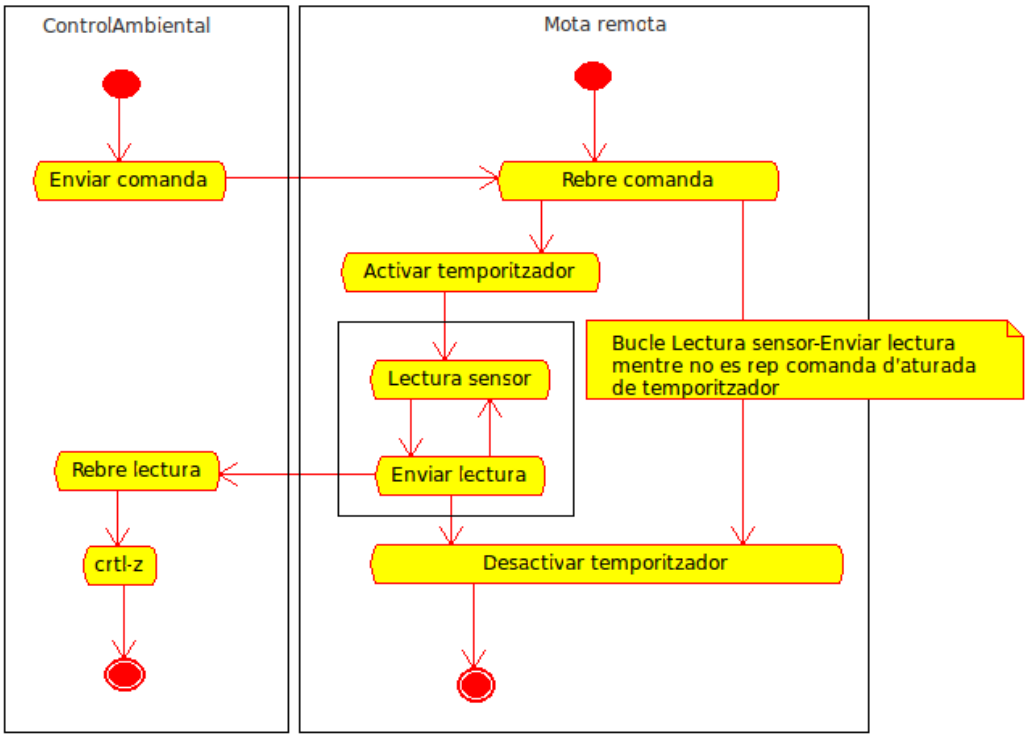


Fig. nº14 : Diagrama activitat comandes primer grup.

En el cas de les comandes del segon grup, únicament interactuen amb el sistema gestor de base de dades, mostrant les llistes de les lectures rebudes:

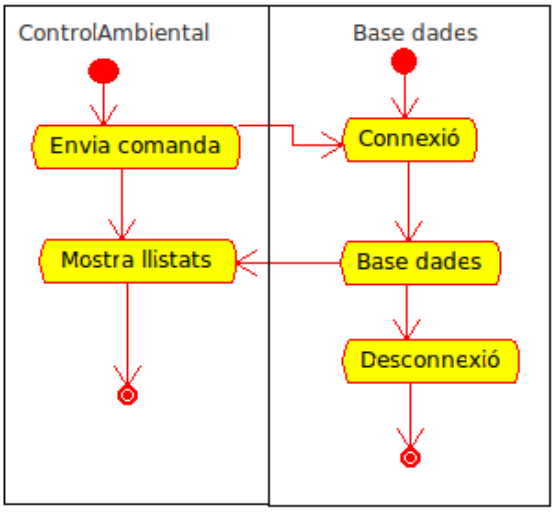


Fig. nº 15 : Diagrama activitat comandes segon grup.

3.3 Aplicació de la mota

Aquesta aplicació correspon a la nomenada “**MotaRemota**” i compren el programari en llenguatge 'nesC', que compilat, resideix en la mota remota i determina el seu comportament. S'han desenvolupat quatre components “**Blink**”, “**TempReporter**”, “**BatReporter**”, “**PhotoReporter**” i “**MacReporter**”. A continuació es fa una petita descripció del funcionament de cadascun, per passar, en els apartats posteriors a la descripció en detall:

- “**Blink**” : Component principal que engloba els demés components i té la funcionalitat d'activar el led groc intermitentment quan la mota remota rep l'alimentació.
- “**TempReporter**” : Aquest component gestiona el sensor de temperatura de la mota remota. Quan el component rep un determinat codi corresponent a la comanda que activa el sensor de temperatura, s'activa el temporitzador amb una determinada freqüència de mostratge. Mentre no s'aturi aquest temporitzador, el component va enviant lectures a la mota base. Igualment es pot configurar un rang de temperatures d'alarma i si el valor llegit pel sensor surt d'aquest rang, la lectura serà enviada, en cas contrari no enviarà cap lectura.
- “**BatReporter**”: Component que gestiona el sensor de les bateries de la mota remota. El funcionament d'aquest component és semblant al anterior, excepte que la configuració de l'alarma és un valor mínim de voltatge de les bateries, quan el valor llegit de les bateries de la mota remota és menor que el valor mínim configurat s'envia la lectura, en cas contrari no.
- “**PhotoReporter**”: Component que gestiona el sensor de lluminositat. El funcionament és molt semblant als components anteriors, excepte que en aquest component no s'ha implementat la configuració d'un valor o rang de valors d'alarma.
- “**MacReporter**” : El comportament d'aquest component és diferent als anteriors, doncs no s'inicia un cicle de lectures, nomes el temporitzador dispara un únic esdeveniment d'enviament d'un missatge amb la direcció mac de la mota remota. Mostrem diagrama d'esdeveniments.

4. Descripció detallada

4.1 Aplicació “ControlAmbiental”

La classe principal on resideix el gruix de l'aplicació és “**MainControl**”. Aquesta classe instància, es a dir, utilitza les altres classes de l'aplicació:

- “**PersistenciaDades**”: Aquesta classe conté tots els mètodes per connectar amb la base de dades i fer les consultes pertinents de les lectures emmagatzemades.
- “**MacMsg**”: Classe que ens permeten rebre el missatge d'informació sobre la direcció mac de la mota remota.
- “**InfMsg**”: Classe que conté els mètodes que ens permeten enviar a la mota remota, el missatge amb els paràmetres de configuració, que l'usuari introdueix mitjançant les comandes.
- “**PhotoMsg**”: Classe que ens permet rebre el missatge amb les lectures de lluminositat.
- “**TempMsg**”: Classe que ens permet rebre el missatge amb les lectures de temperatura.
- “**BatMsg**”: Classe que ens permet rebre el missatge amb les lectures del valor de les bateries.
- “**TimerTask**”: Classe que és proveïda pel paquet de llibreries de “**TinyOS**” i ens permet utilitzar un temporitzador per l'enviament de missatges.
- “**MessageListener**” : Interfase que també proveeix el paquet “**TinyOS**” i ens permet posar l'aplicació en mode d'escolta per rebre els missatges.

Mostrem un diagrama de classes de l'aplicació:

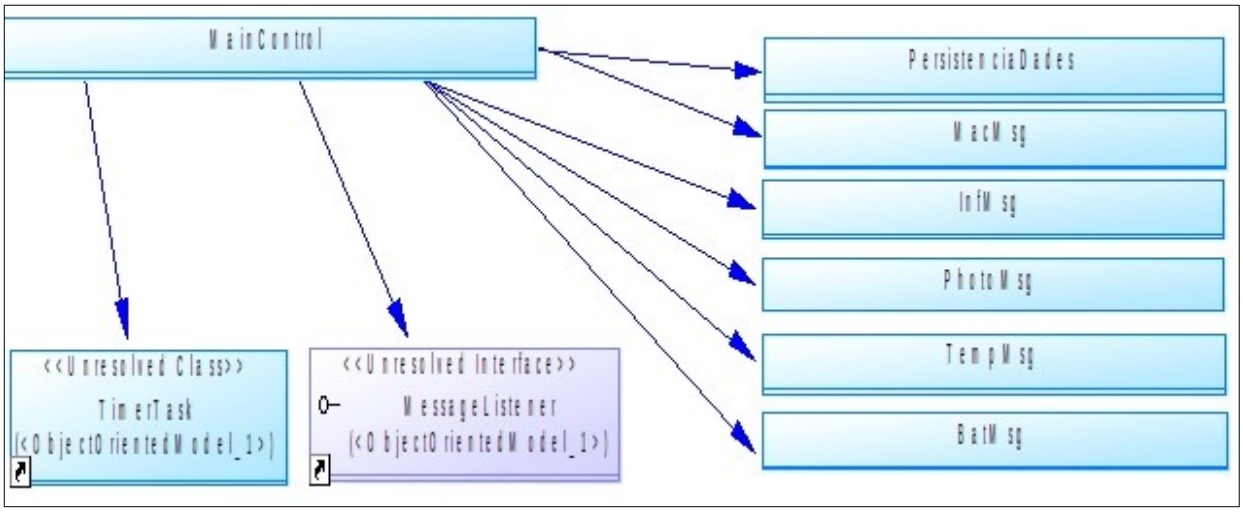


Fig.nº 16: Diagrama de classes de l'aplicació “ControlAmbiental”.

La classe "**MainControl**" realitza principalment les següents tasques:

- Interpreta les comandes introduïdes per l'usuari i envia un missatge tipus "InfMsg" amb els paràmetres de configuració a la mota remota com el tipus de sensor a activar, la freqüència de mostreig, i alarmes.
- Posa a l'aplicació en mode d'escolta per rebre els quatre tipus de missatges de la mota remota, que son la lectura de la mac, lectures sensor temperatura, lluminositat i valor de les bateries. Tanmateix que rep els missatges els insereix a les taules respectives, excepte el missatge d'informació de la mac.
- Connecta amb la base de dades per mostrar les lectures emmagatzemades.
- Executa l'aplicació gràfica integrada en el codi font.

Mostrem a continuació la estructura d'aquesta classe:

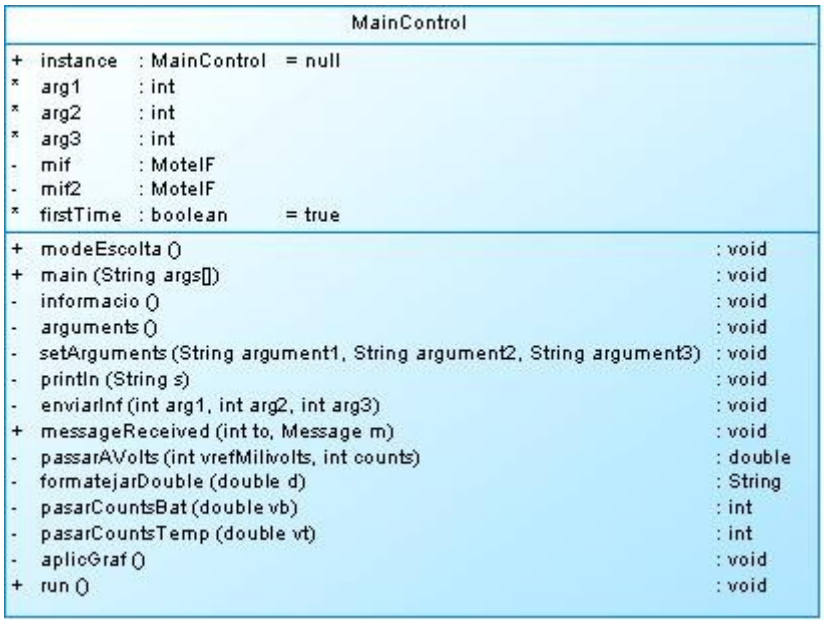


Fig. nº 17: Classe "MainControl".

Com es veu en la figura nº 15, aquesta classe conté el mètode estàtic "main" que s'encarrega d'executar l'aplicació, cridant al mètode estàtic "arguments". En aquest mètode resideix pràcticament la interfície d'usuari, ja que, les comandes que introdueix l'usuari son interpretades amb una sèrie de codis, que son carregats dins la estructura del tipus de missatge "InfMsg". Una vegada aquest missatge conté les dades de configuració, que es carreguen en el mètode "setArguments", és enviat via ràdio amb els mètodes "enviarInf" i "modeEscolta". Ja s'ha esmentat en el apartat anterior que aquesta aplicació "ControlAmbiental" es connecta via socket TCP/IP amb l'aplicació "SerialForwarder" i aquesta al mateix temps, escolta la mota base connectada al port sèrie USB.

Quan s'envia qualsevol de les comandes de configuració de la mota remota, l'aplicació queda en mode “escolta”,mitjançant els mètodes “messageReceived” i “run”, i pot rebre qualsevol tipus de missatge de la mota remota depenent de les comandes que l'usuari hagi introduït. A continuació mostrem la estructura i els camps dels diferents tipus de missatges que utilitza l'aplicació.
L'aplicació pot rebre quatre diferents tipus de missatge que especifiquem en aquesta taula, indicant els tipus de camps d'informació que conté amb una creu:

missatge/camp	**moteId**	**macAddr[8]**	**countsTemp**	**countsBat**	**countsPhoto**	**vrefmilivolts**	**counter**
MacMsg	**X**	**X**					
TempMsg	**X**		**X**			**X**	**X**
BatMsg	**X**			**X**		**X**	**X**
PhotoMsg	**X**				**X**		**X**

Expliquem els diferents tipus de camps:

- **“moteId”**: Camp identificació de la mota que envia el missatge.
- **“macAddr[8]”:** Direcció mac de la mota que envia el missatge.
- **“countsTemp”:** Valor de lectura del sensor de temperatura.
- **“countsBat”:** Valor de lectura de les bateries de la mota
- **“countsPhoto”:** Valor de lectura del sensor de lluminositat.
- “**vrefmilivolts**”: Valor de referència de voltatge en mili volts.
- “**counter**”: Numerador del nombre de paquet que es rep.

A efectes pràctics el camp **“vrefmilivolts”,** no s'utilitza quan es rep els dos tipus de missatges “**TempMsg**” i “**BatMsg**”. El mètode “**messageReceived**” detecta el tipus de missatge que es rep, i en el cas del missatge tipus “**MacMsg**”, aquest no s'emmagatzema a la base de dades, en canvi els tipus “**TempMsg**”, “**BatMsg**” i “**PhotoMsg**”, si que s'emmagatzemen a la base de dades. Per a la persistència de les dades es fa ús de la classe “**PersistenciaDades**” , que detallarem més endavant.

La classe “**MainControl**”, només envia un tipus de missatge, per a la configuració de la mota remota que es “**InfMsg**”. Aquest missatge conté quatre camps d'informació que son:

- “**moteId**”: Identificació de la mota a la qual s'envia el missatge.
- “**arg1**”: Aquest camp porta un determinat codi, i aquest codi pot significar tres tasques diferents com el sensor que volem activar, sensor que volem aturar o alarma que volem configurar.

- "**arg2**": Aquest camp s'utilitza per configurar la freqüència de mostratge de lectura del sensor, pel primer valor del rang de temperatures d'alarma o pel valor mínim de voltatge de l'alarma de les bateries de la mota remota.
- "**arg2**": Aquest camp s'utilitza exclusivament pel segon valor del rang de temperatures que es configuren d'alarma.

Tenim un grup de comandes que l'usuari pot introduir per configurar la mota remota. Aquest grup de comandes carreguen en els camps del missatge tipus "**InfMsg**" uns determinats codis, depenent de la comanda introduïda. Aquest missatge amb els codis és el que s'envia per radio i que rep la mota remota. A continuació mostrem una taula que relaciona la comanda que introdueix l'usuari , amb els codis i valors, que introduïm en els diferents camps d'informació del missatge. En el nostre cas obviarem el valor del camp "**moteId**", ja que, aquest valor sempre és "**1**" , que és la identificació de la única mota remota que tenim.

comanda/camp	**"arg1"**	**"arg2"**	**"arg3"**
"MAC"	**1**	**0**	**0**
"TEMPERATURA"	**2**	**Freqüència mostratge**	**0**
"ATURAR TEMPERATURA"	**3**	**0**	**0**
"BATERIA"	**4**	**Freqüència mostratge**	**0**
"ATURAR BATERIA"	**5**	**0**	**0**
"LLUM"	**6**	**Freqüència mostratge**	**0**
"ATURAR LLUM"	**7**	**0**	**0**
"ALARMA BATERIA"	**8**	**Voltatge mínim**	**0**
"ALARMA TEMPERATURA"	**9**	**Temperatura minima**	**Temperatura màxima**

El missatge "**InfMsg**", amb els seus camps d'informació, és rebut pels diferents components programats a la mota remota, concretament el programari "**MotaRemota**", que detallarem en els apartats posteriors.
El mètode "**messageReceived**", rep els diferents tipus de missatges i s'encarrega d'extreure la informació dels camps, però en el cas de rebre el tipus de missatge "**TempMsg**", necessitem convertir el valor que s'extreu del seu camp "**countsTemp**" a graus centígrads i per això tenim el mètode "**passarAVolts**", que utilitza una determinada formula per passar a mili volts, aquesta és:

$$(\frac{countsTemp}{1024})*(\frac{2405}{1000})$$

Posteriorment al valor que resulta d'aplicar aquesta fórmula, li resta 0.5 i el divideix per 0.01, el resultat és en graus centígrads. Igualment en el cas de rebre el tipus de missatge "**BatMsg**" , també necessitem passar el valor del seu camp "**countsBat**" a mili volts utilitzant el mètode "**passarAVolts**", però en aquest cas no es fa correcció posterior. La formula que apliquem és:

$$(\frac{countsBat}{1024})*(\frac{2405}{1000})$$

En el procés d'enviament del missatge "**InfMsg**", en el cas de configurar les alarmes tant de temperatura com de bateria, necessitem fer la operació contraria a la anterior, es a dir, el valor en volts que introdueix l'usuari per a configurar l'alarma de les bateries, necessitem transformar-la a "**countsBat**" i igualment els valors en graus centígrads que s'introdueix per configurar el rang de temperatures d'alarma, també necessitem transformar-los a "**countsTemp**" . Aquestes transformacions son necessàries perquè una vegada la mota remota rep el valor de les alarmes, dintre dels diferents camps del missatge, necessitem comparar aquests valors, en els components adients del programari "**MotaRemota**" , com explicarem en el apartat posterior. Per passar de volts a "**countsBat**" ho fem amb el mètode "**pasarCountsBat**", aplicant la formula:

$$volts*1024*1000*(\frac{1}{4810})$$

Per passa de graus centígrads a "**countsTemp**" utilitzem el mètode "**pasarCountsTemp**" i apliquem la formula:

$$(temperatura*10240+1024*500)*(\frac{1}{4810})$$

Dintre del mètode "**arguments**", tenim un grup de comandes que l'usuari pot introduir, que no impliquen cap enviament de missatge de configuració a la mota remota, si no que utilitzen la classe "**PersistenciaDades**" per connectar amb la base de dades i treure llistats de les lectures rebudes. Mostrem una taula amb aquestes comandes i la seva explicació:

Comanda	**Funcionalitat**
"TAULA TEMPERATURA"	Mostre llistat de lectures de temperatura
"TAULA BATERIA"	Mostre llistat de lectures de bateria
"TAULA LLUMINOSITAT"	Mostre llistat de lectures de lluminositat
"ESBORRAR TEMPERATURA"	Esborra taula temperatura
"ESBORRAR BATERIA"	Esborra taula bateria
"ESBORRAR LLUMINOSITAT"	Esborra taula lluminositat

Mereix una menció a part, la comanda “**APLIC GRAF**”, aquesta és la última funcionalitat que s'ha desenvolupat. Quan l'usuari introdueix aquesta comanda per la consola, s'obra una petita aplicació gràfica desenvolupada i compilada en el llenguatge “Gambas2”, que és una espècie de “.NET” per els sistemes operatius “Linux”. Aquesta aplicació mostra les taules dels diferents tipus de lectures d'una forma gràfica més còmoda i propera per l'usuari.

Mostrem a continuació la estructura de la classe “**PersistenciaDades**”:

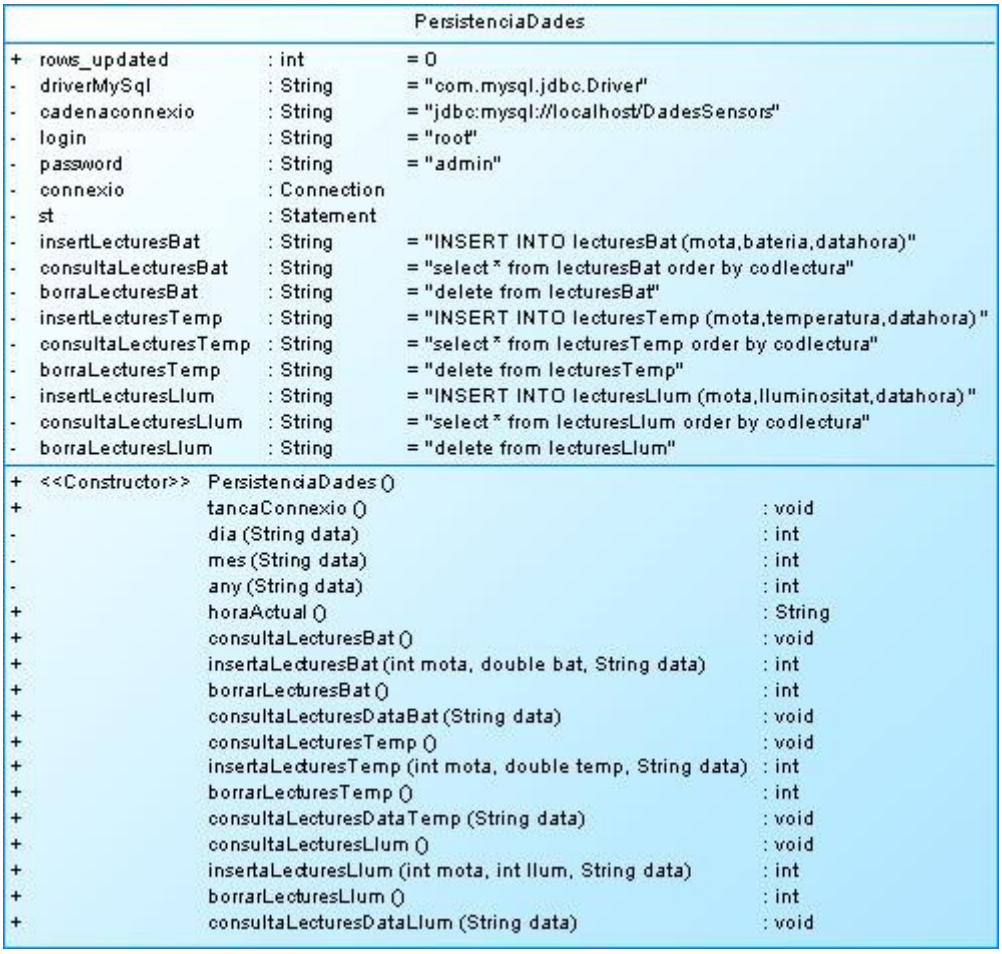


Fig. nº 18: Classe “PersistenciaDades”.

Aquesta classe és utilitzada per la classe principal “**MainControl**”, i la proveeix de quatre funcionalitats principals que son:

- Connexió amb el sistema gestor de base de dades MySql, que s'ha instal·lat en el nostre ordinador.
- Inserció de dades.
- Consulta de dades.
- Eliminació de registres de les diferents taules.

La estructura del model de dades és molt senzill, únicament conte tres taules una per a cada tipus de lectures, mostrem a continuació les taules i els camps del model de dades:

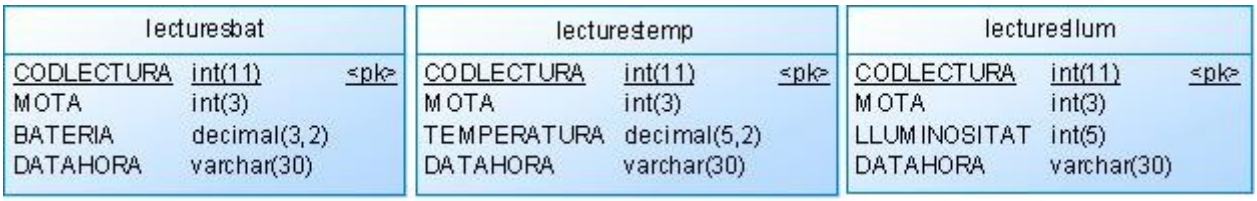


Fig.nº 19: Model de dades.

Les dades de connexió a la base de dades estan configurades, tant en aquesta classe com en l'aplicació gràfica. Com es veu en la figura número 16, les dades de connexió comprenen els atributs “**cadenaconnexio**”, “**login**” i “**password**”. Els mètodes de consulta de les diferents taules son “**consultaLecturesBat**”, “**consultaLecturesTemp**” i “**consultaLecturesLlum**”. Les consultes filtrades pels camps “**DATAHORA**”, de les taules correspon als mètodes, “**consultaLecturesDataBat**”, “**consultaLecturesDataTemp**” i “**consultaLecturesDataLlum**”. Els mètodes d'inserció de dades son, “**insertaLecturesTemp**”, “**insertaLecturesBat**” i “**insertaLecturesLlum**” . Els mètodes d'eliminació de registres son , “**borrarLecturesTemp**”, “**borrarLecturesBat**” i “**borrarLecturesLlum**”, destacar que aquests mètodes eliminen tots el registres de la taula, i s'han d'utilitzar amb precaució.

4.2 Aplicació “MotaRemota”

Aquesta aplicació compren el programari en llenguatge 'nesC', que compilat, resideix en la mota remota i determina el seu comportament. A continuació descriurem el seus components:

- “**Blink**” : Component principal que engloba els demés components i té la funcionalitat d'activar el led groc intermitentment quan la mota remota rep l'alimentació.
- “**TempReporter**” : Aquest component gestiona el sensor de temperatura de la mota remota. Quan l'esdeveniment '**rebre**' del component, rep el codi apropiat de la mota base, activa el temporitzador amb una determinada freqüència de mostreig, que al mateix temps activa el procés de lectura del sensor de temperatura, aquest procés passa per diferents estats fins que envia a la mota base les lectures de la temperatura. Aquest procés es va repetint fins que l'esdeveniment '**rebre**' , rebi el codi de la mota base que aturi el temporitzador i tanqui el sensor. Ara bé, mentre aquest cicle de lectures de temperatura esta en marxa l'esdeveniment '**rebre**' , pot rebre un codi d'alarmes i en aquest moment el procés de lectures abans d'enviar les lectures, compararà el seu valor amb el rang de temperatures d'alarma. Si el valor es troba dins el rang establert, la lectura no s'enviarà, però el cicle continua llegint els valors de temperatura i si en algun cas el valor surt del rang llavors la lectura de temperatura serà enviada per radio. El procés continua fins que es rep altre configuració d'alarmes o fins que s'aturi el temporitzador i com a conseqüència el sensor. Completem l'explicació del component amb un diagrama d'esdeveniments i d'estats. En aquest diagrama podem observar que el cicle de lectura del sensor s'inicia quan es rep el codi corresponent a la comanda '**TEMPERATURA**', amb la freqüència de mostreig. L'esdeveniment “**Rep InfMsg**” correspon a l'esdeveniment '**rebre**', que hem esmentat anteriorment. Aquest esdeveniment rep el missatge amb els paràmetres, tant d'activació del sensor, freqüència de mostreig com el rang d'alarmes i també el codi d'aturar el sensor. L'esdeveniment “**Lectura temperatura**”, compara el valor llegit de temperatura amb el valor dels paràmetres rebuts i decideix si s'ha d'enviar la lectura,en cas d'haver alarmes, o s'ha d'aturar el temporitzador i el sensor:

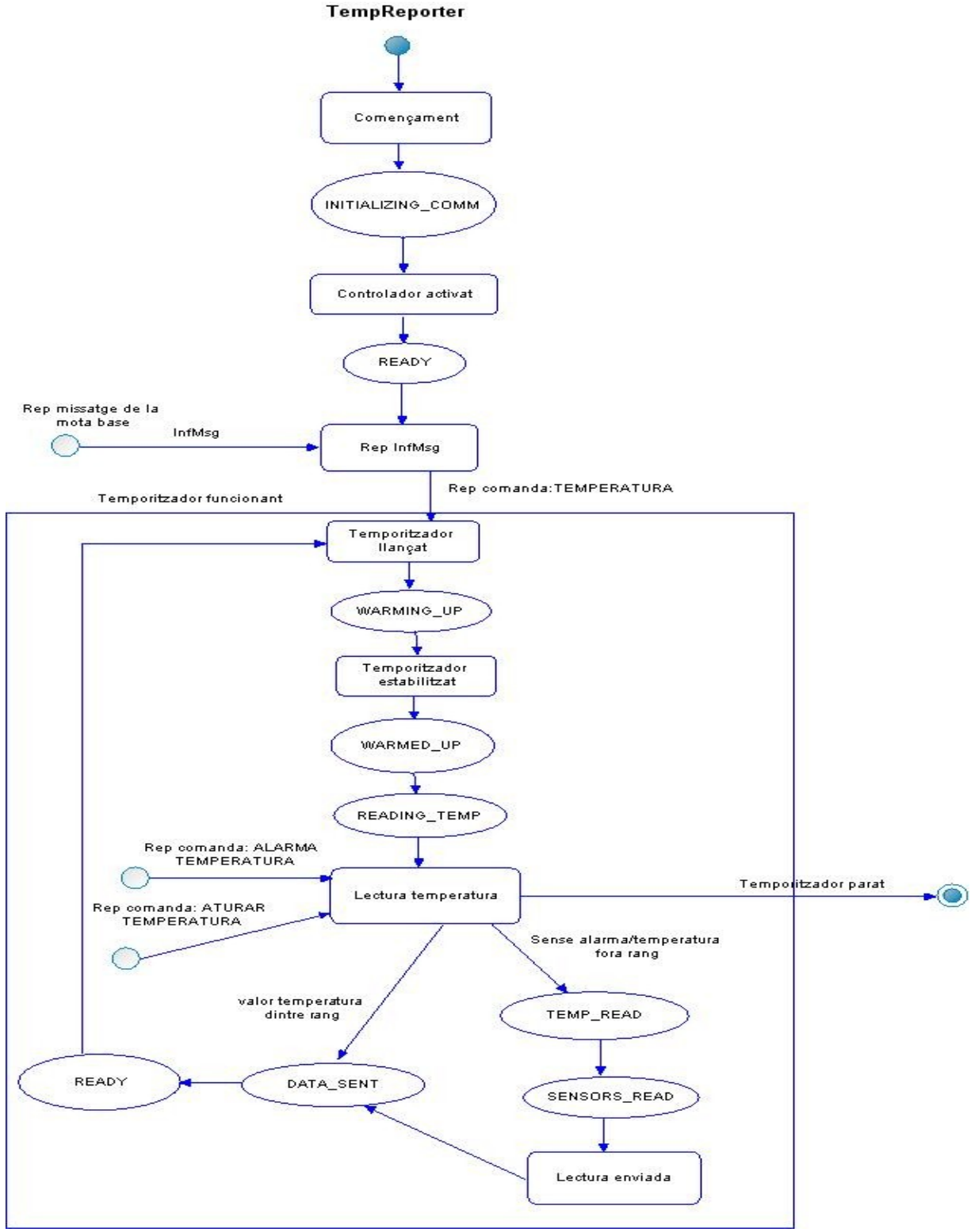


Fig.nº 20: Diagrama d'estats del component 'TempReporter'.

- "**BatReporter**": Component que gestiona el sensor de les bateries de la mota remota. El funcionament d'aquest component és semblant al anterior, excepte que la configuració de l'alarma és un valor mínim de voltatge de les bateries, en canvi en el component anterior l'alarma era un rang de temperatures amb un valor mínim i un altre màxim en graus centígrads. Quan l'esdeveniment "**Rep InfMsg**", rep el codi corresponent a la comanda "**BATERIA**", s'activa el temporitzador que dispara el proces de lectura del sensor de les bateries. Durant aquest proces es pot configurar l'alarma amb un valor mínim del voltatge. Posteriorment es pot aturar el proces de lectures i aturar el sensor, amb la comanda "**ATURAR BATERIA**" . Completem la explicació amb el diagrama d'estats corresponent a la figura número 22.
- "**PhotoReporter**": Component que gestiona el sensor de lluminositat. El funcionament és molt semblant als components anteriors, excepte que en aquest component no s'ha implementat la configuració d'un valor o rang de valors d'alarma. Conseqüentment quan l'esdeveniment "**Rep InfMsg**", rep el codi corresponent a la comanda "**LLUM**", s'activa el temporitzador, el proces de lectura de sensor i l'enviament de missatges de lectures. El proces s'atura si es rep el codi corresponent a la comanda "**ATURAR LLUM**" .Mostrem diagrama d'estats i esdeveniments per completar l'explicació, corresponent a la figura número 23.
- "**MacReporter**" : El comportament d'aquest component és diferent als anteriors, doncs no s'inicia un cicle de lectures, nomes el temporitzador dispara un únic esdeveniment d'enviament d'un missatge amb la direcció mac de la mota remota. Mostrem diagrama d'esdeveniments:

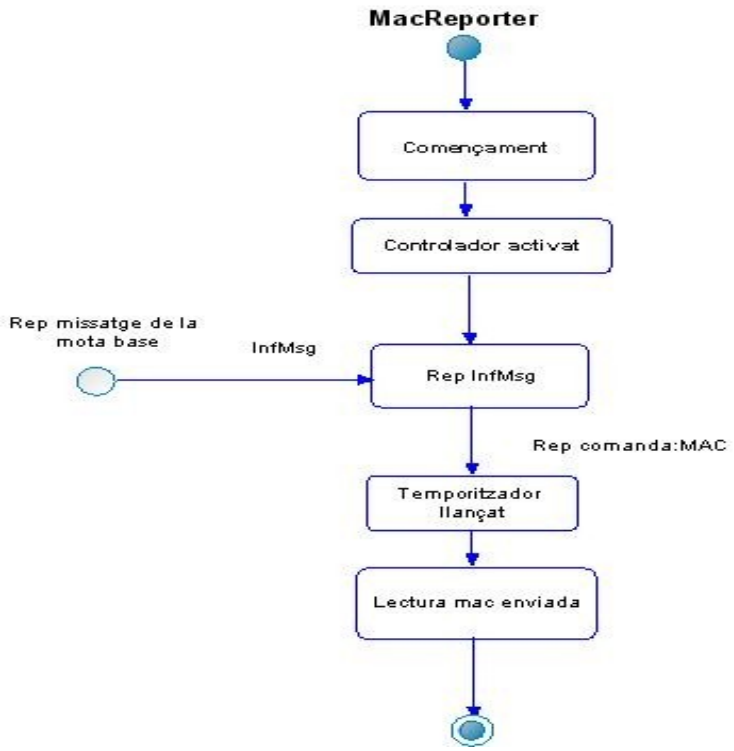


Fig.nº 21: Diagrama d'esdeveniments component "MacReporter".

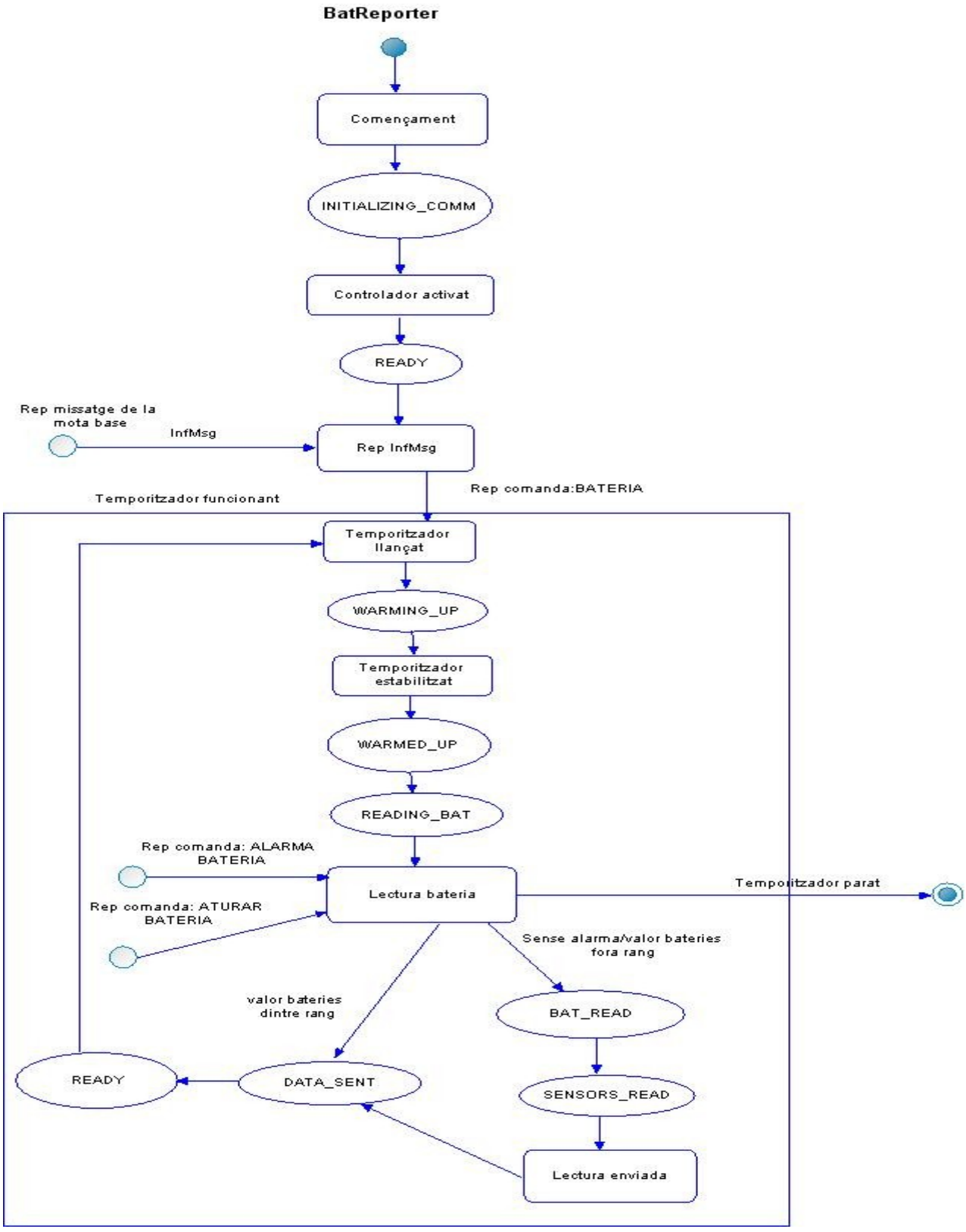


Fig.nº 22: Diagrama d'estats del component "BatReporter".

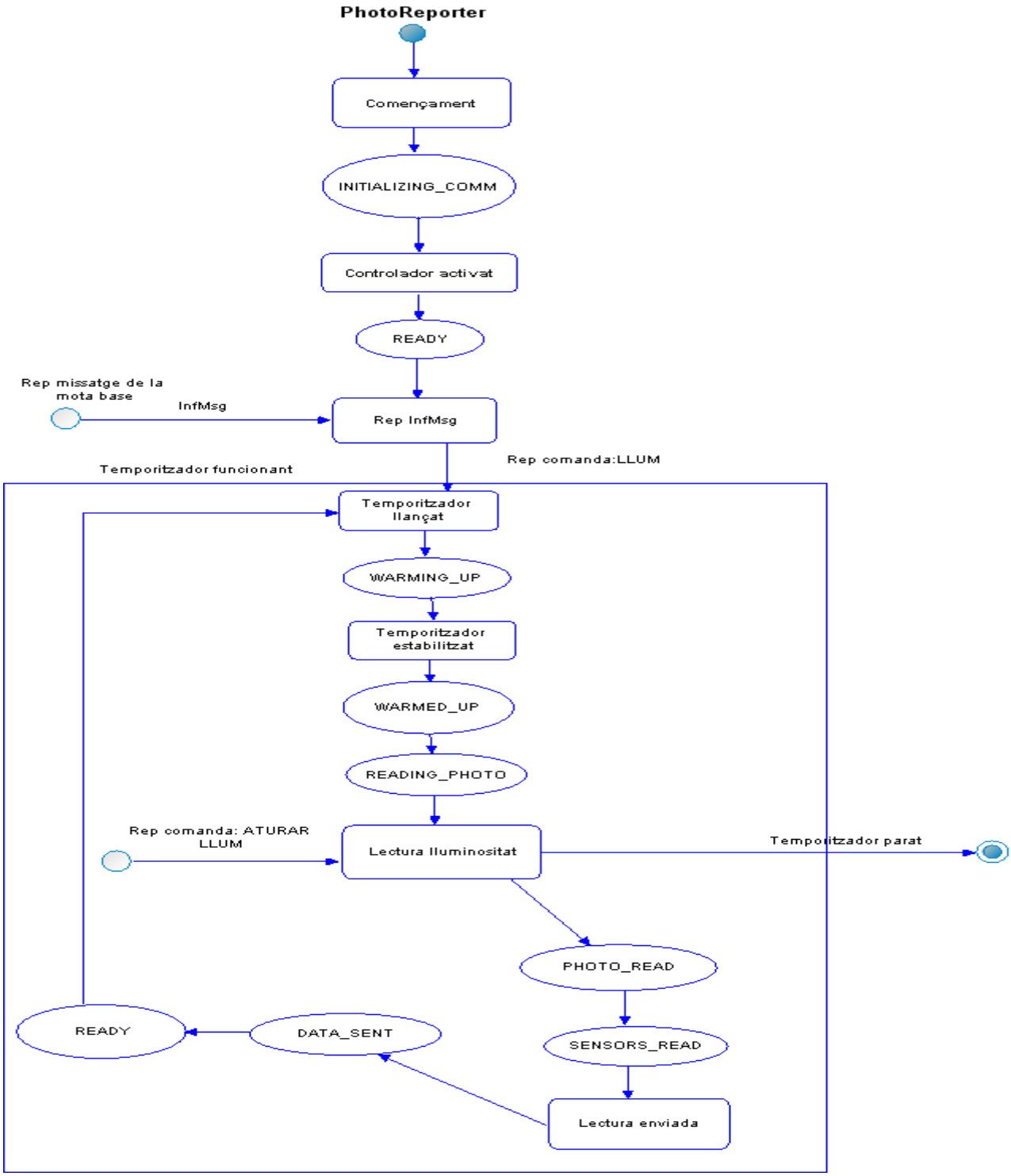


Fig. nº 23: Diagrama d'estats component "PhotoReporter".

4.2.1 Component “TempReporter”

Després d'exposar les funcionalitats principals d'aquest component en el apartat anterior, en aquest apartat detallarem les interfases que utilitza i la forma d'interactuar amb la interfície d'usuari. A continuació es fa una breu descripció de les interfases utilitzades:

- “**Boot**” : Interfase utilitzada per enllaçar l'arranc del sistema amb la inicialització de l'aplicació instal·lada.
- “**Timer**” : Interfase que proveeix les llibreries per utilitzar el temporitzador del sistema.
- “**TimerStabilize**” : Temporitzador que s'utilitza per deixar un temps de preparació al sensor, abans de ser llegit.
- “**SinkPowerSensors**” : Interfase que proveeix les llibreries per activar i aturar els sensors.
- “**LecturaAdc**”: Interfase que proveeix les llibreries per llegir la senyal analògica del sensor i convertir-la a digital. Llegeix convertidor analògic-digital.
- “**Atm128AdcConfig**”: Interfase que proveeix els mètodes per configurar els paràmetres del convertidor analògic a digital del microcontrolador 'Atmega', com per exemple el voltatge de referència o el canal per llegir la senyal del sensor.
- “**powerSerie**”: Interfase que s'utilitza per preparar la radio, és un controlador d'esdeveniments.
- “**Packet**”: Interfase que proveeix els mètodes per accedir a les propietats i contingut dels diferents tipus de missatges “ActiveMessage”.
- “**AMSend**”: Interfase que proveeix els mètodes per l'enviament de missatges de tipus “ActiveMessage”.
- “**Receive**”: Interfase que proveeix els mètodes per rebre els missatges “ActiveMessage”, en el nostre cas amb direcció “broadcast”.
- “**Leds**”: Interfase que proveeix els mètodes pel control dels “leds” de la mota.

Aquest component utilitza dos tipus de missatges. Per enviar les lectures del sensor de temperatura carrega les dades dintre del missatge “**TempMsg**”, definit en el fitxer “**Msgs.h**”, amb la següent estructura:

Arguments missatge "TempMsg":	**Explicació:**
"moteId"	Camp identificació de la mota que envia el missatge.
"countsTemp"	Valor de lectura del sensor de temperatura.
"vrefmilivolts"	Valor de referència de voltatge en mili volts.
"counter"	Numerador del nombre de paquet que es rep.

Per rebre paràmetres de configuració, rep un missatge de tipus “**InfMsg**”, definit en el fitxer “**Msgs.h**”, amb la següent estructura:

Arguments missatge "InfMsg":	**Explicació:**
"moteId"	Camp identificació de la mota a la que s'envia el missatge.
"arg1"	Codi del sensor a activar, codi del sensor a aturar o codi de l'alarma a configurar.
"arg2"	Freqüència de mostreig o primer valor del rang de temperatures d'alarma.
"arg3"	Segon valor del rang de temperatures d'alarma.

Aquest component quan rep un missatge de tipus “**InfMsg**”, només s'activa i executa alguna acció quan l'argument “**arg1**”, conté els següents codis, corresponents a tres comandes diferents introduïdes per l'usuari:

Comanda:	**Codi:**	**Explicació:**
"TEMPERATURA"	2	Inicialitza temporitzador per llegir sensor temperatura.
"ATURAR TEMPERATURA"	3	Atura temporitzador i sensor temperatura.
"ALARMA TEMPERATURA"	9	Configura paràmetres d'alarma de temperatura.

Hi ha dos esdeveniments principals que determinen el funcionament d'aquest component, l'esdeveniment “**event** message_t * Receive.receive” i l'esdeveniment “**event** void LecturaAdc.readDone”, el primer correspon a la interfase “**Receive**” i el segon a la interfase “**LecturaAdc**”.
Quan l'esdeveniment “**Receive.receive**”, rep un missatge de tipus “**InfMsg**”, on el valor en el argument “**arg1**” és igual a “**2**”, activa la conta enredera del temporitzador del sistema “**Timer.startPeriodic**”, amb el valor del argument “**arg2**”, com a valor de la freqüència de mostratge. Quan el valor en mili segons de la freqüència expira, comença el procés de lectura del sensor de temperatura. Això es reflexa en l'esdeveniment del temporitzador del sistema “**Timer.fired**”, que comença el procés de lectura del sensor, cada vegada que el temps del temporitzador expira. El cicle de lectura del sensor i enviament del missatge tipus “**TempMsg**” , es repeteix periòdicament, fins que l'esdeveniment “**Receive.receive**” rebi un missatge tipus “**InfMsg**” amb el valor del seu argument “**arg1**” igual a “**3**”. En aquest moment s'atura el temporitzador amb el mètode “**Timer.stop**” i també s'atura el sensor amb el métode “**_turnOffSensors()**”. Mostrem un diagrama de flux amb el funcionament d'aquest esdeveniment:

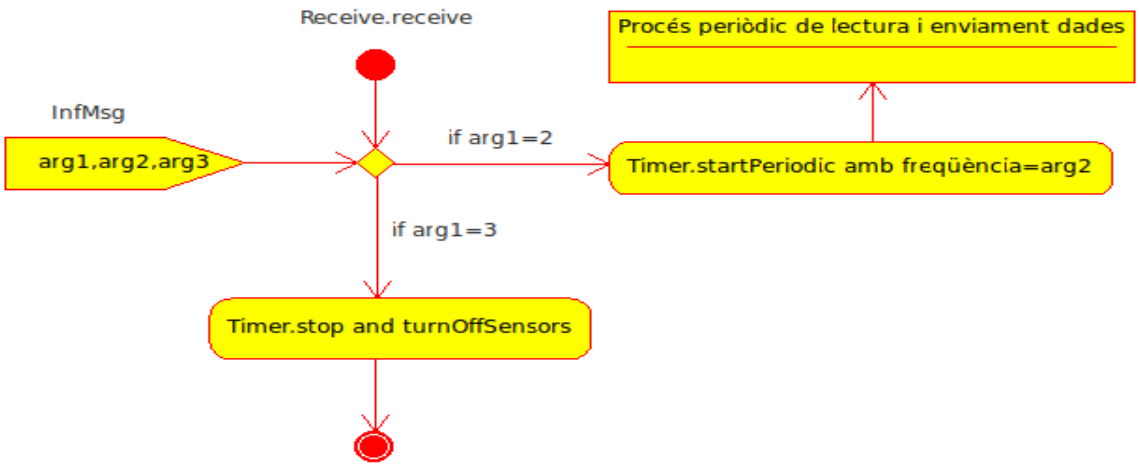


Fig.nº 24 : Diagrama flux de l'esdeveniment "Receive.receive" en "TempReporter".

En el proces periòdic que s'executa, de lectura del valor del sensor de temperatura i enviament de les lectures, intervé l'esdeveniment "**event** void LecturaAdc.readDone**"**, que es produeix quan s'ha lligit el valor del sensor . Quan aquest esdeveniment detecta un valor del argument "**arg1**" igual a "**9**", sap que l'usuari a configurat una alarma de temperatura, on el rang de valors de la temperatura mínima i la temperatura màxima resideixen , respectivament, en els arguments "**arg2**" i "**arg3**". A les hores, en cada cicle de lectura del sensor de temperatura, l'esdeveniment compara el valor obtingut amb el rang de temperatures d'alarma, si el valor es dintre d'aquest rang la lectura no s'envia per radio, ja que l'estat del proces es posa a "**estat=DATA_SENT**", en canvi si el valor llegit del sensor surt del rang d'alarma establert l'estat del proces es posa a "**estat=TEMP_READ**" i permet que s'envii les dades de la lectura de temperatura per ràdio, mitjançant els mètodes de la interfase "**AMSend**". Acompanyem l'explicació amb el diagrama de flux de l'esdeveniment:

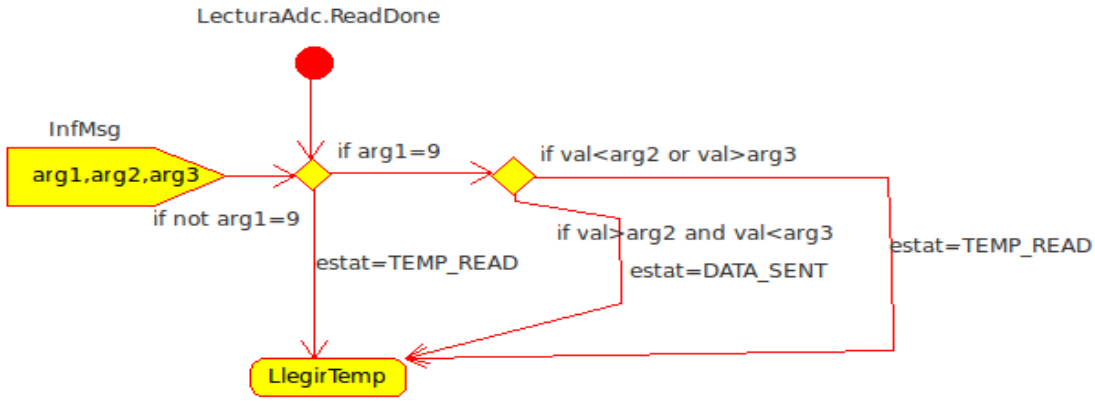


Fig.nº25 : Diagrama flux de l'esdeveniment "LecturaAdc.ReadDone" en "TempRporter".

4.2.2 Component “BatReporter”

Aquest component és l'encarregat de llegir el sensor que detecta el valor de les bateries de la mota remota. Aquest valor és critic doncs si baixa de 2,56 volts que marquen les especificacions de la mota, pot ser que les lectures de temperatura i lluminositat rebudes siguin lleugerament valors erronis. Aquest component utilitza exactament les mateixes interfases que el component anterior, on ja hem explicat les funcionalitats de cada interfase que s'utilitza. Aquest component utilitza dos tipus de missatges. Per enviar les lectures del sensor de les bateries utilitza el missatge “**BatMsg**” , definit en el fitxer “**Msgs.h**”, amb la següent estructura:

Arguments missatge "BatMsg":	**Explicació:**
"moteId"	Camp identificació de la mota que envia el missatge.
"countsBat"	Valor de lectura de les bateries de la mota
"vrefmilivolts"	Valor de referència de voltatge en mili volts.
"counter"	Numerador del nombre de paquet que es rep.

Igual que el component anterior per rebre paràmetres de configuració, rep missatges de tipus “**InfMsg**”, definit en el fitxer “**Msgs.h**” amb la següent estructura:

Arguments missatge "InfMsg":	**Explicació:**
"moteId"	Camp identificació de la mota a la que s'envia el missatge.
"arg1"	Codi del sensor a activar, codi del sensor a aturar o codi de l'alarma a configurar.
"arg2"	Freqüència de mostreig o primer valor del voltatge mínim d'alarma a configurar.
"arg3"	En aquest component no s'utilitza.

Aquest component nomes executa alguna acció quan rep en el missatge de tipus “**InfMsg**”, en el argument “**arg1**”, un dels següents codis corresponents a tres comandes determinades:

Comanda:	**Codi:**	**Explicació:**
"BATERIA"	4	Inicialitza temporitzador per llegir sensor bateries.
"ATURAR BATERIA"	5	Atura temporitzador i sensor bateries.
"ALARMA BATERIA"	8	Configura paràmetres d'alarma de bateries.

Igual que en el component anterior, en aquest tenim dos esdeveniments que interaccionen remotament amb la interfície d'usuari. L'esdeveniment "**event** message_t * Receive.receive", que rep el missatge "**InfMsg**" i interpreta el codi emmagatzemat en l'argument "**arg1**". Si aquest codi val "**4**", dispara el temporitzador del sistema, amb la freqüència de mostratge en l'argument "**arg2**", i quan el temporitzador expira comença el proces de lectura del sensor i enviament de les dades. Per aturar aquest proces ha de rebre un codi amb valor igual a "**5**", que aturarà el temporitzador i tancara el sensor de bateria. Amb la funció "**Timer.startPeriodic**" s'activa la conta enrere del temporitzador i amb les funcions "**Timer.stop**" i "**_turnOffSensors",** es desactiven temporitzador i sensor. Adjuntem diagrama de flux d'aquest esdeveniment:

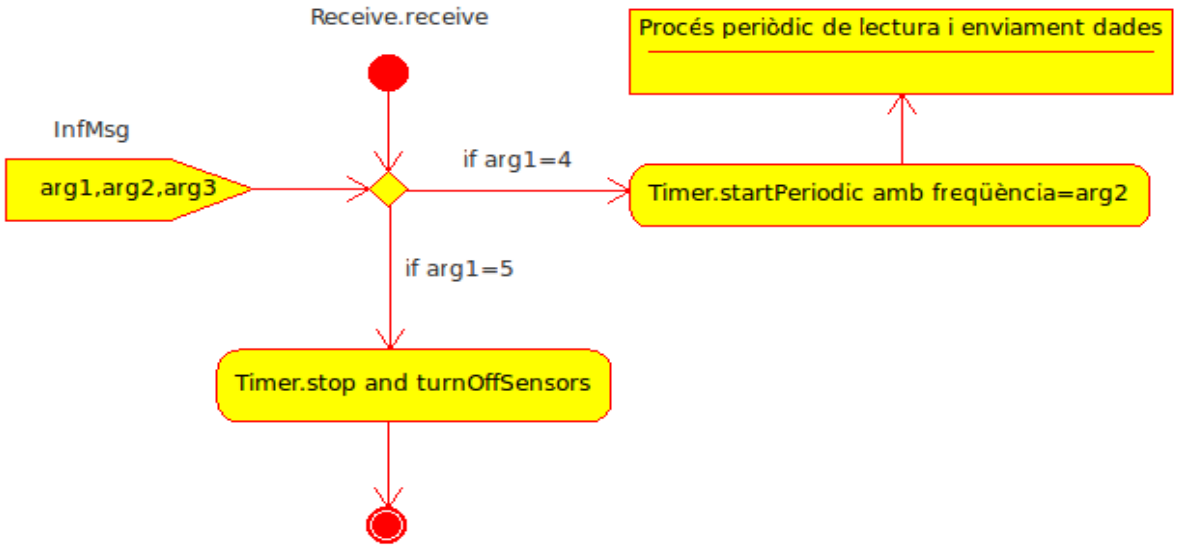


Fig.nº26 : Diagrama flux de l'esdeveniment "Receive.receive" en "BatReporter".

El segon esdeveniment "**event** void LecturaAdc.readDone", modifica el cicle periòdic de lectura de sensor i enviament de dades en el cas de detectar un valor de "**8**" en l'argument "**arg1**" de "**InfMsg**". En aquest component la configuració de l'alarma consisteix en introduir per part de l'usuari, un valor de voltatge mínim , que anirà emmagatzemat en "**arg2**" . Si el valor llegit del sensor és més gran que el valor configurat en "**arg2**", l'esdeveniment posarà la variable d'estat del proces a "**estat=DATA_SENT**" i la lectura del sensor no serà enviada, en canvi en el cas contrari la variable d'estat del proces serà "**estat=BAT_READ**" i la lectura s'enviarà indicant un nivell de les bateries més baixes que el valor mínim configurat. Indicar que mentre el temporitzador no s'aturi amb l'esdeveniment anterior, l'usuari pot modificar el nivell mínim de voltatge de l'alarma amb la comanda pertinent. Adjuntem el diagrama de flux d'aquest esdeveniment:

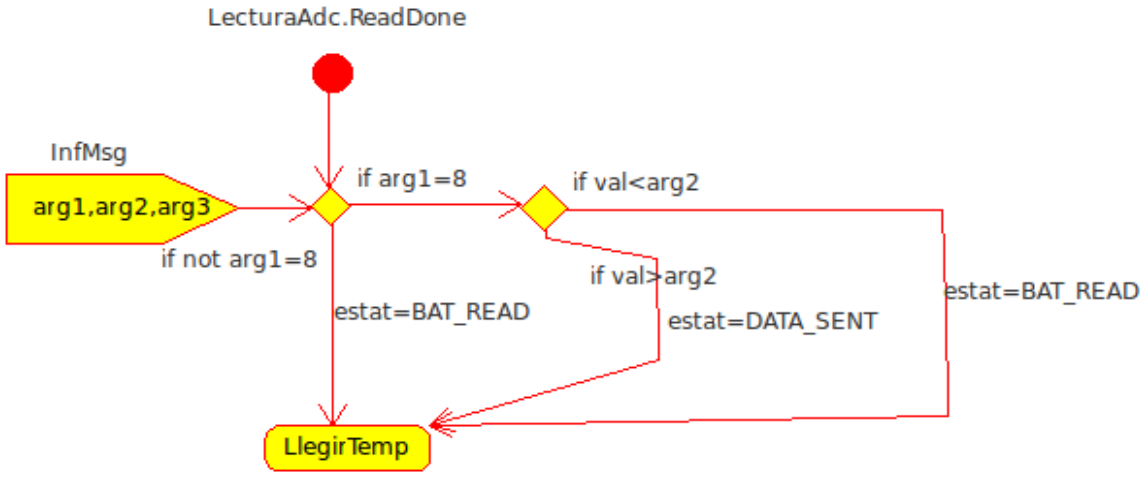


Fig.nº27 : Diagrama flux de l'esdeveniment "LecturaAdc.ReadDone" en "BatReporter".

4.2.3 Component "PhotoReporter"

Component que s'encarrega de la gestió del sensor de lluminositat. Utilitza les mateixes interfases que els dos components anteriors. Aquest component no implementa la configuració d'alarma. Igualment que els components anteriors, utilitza dos tipus de missatges. Missatge tipus "**PhotoMsg**" per l'enviament de les lectures del sensor, definit en el fitxer "**Msgs.h**", amb la estructura:

Arguments missatge "PhotoMsg":	**Explicació:**
"moteId"	Camp identificació de la mota que envia el missatge.
"countsPhoto"	Valor de lectura del sensor de lluminositat.
"counter"	Numerador del nombre de paquet que es rep.

També utilitza el missatge tipus "**InfMsg**", per a la configuració dels paràmetres de lectura del sensor:

Arguments missatge "InfMsg":	**Explicació:**
"moteId"	Camp identificació de la mota a la que s'envia el missatge.
"arg1"	Codi del sensor a activar, codi del sensor a aturar .
"arg2"	Freqüència de mostreig.
"arg3"	En aquest component no s'utilitza.

Aquest component nomes executa alguna acció quan rep en el missatge de tipus “**InfMsg**”, en el argument “**arg1**”, un dels següents codis corresponents a dos comandes determinades:

Comanda:	**Codi:**	**Explicació:**
"LLUM"	6	Inicialitza temporitzador per llegir sensor lluminositat.
"ATURAR LLUM"	7	Atura temporitzador i sensor lluminositat.

En aquest component tenim l'esdeveniment “**event** message_t * Receive.receive”, igual que en els anteriors, encarregat d'activar el temporitzador que dispara el proces de lectura del sensor de lluminositat, en el cas de rebre un valor de “**6**” en l'argument “**arg1**” del missatge tipus “**InfMsg**”. Amb la freqüència determinada en “**arg2**” s'inicia el proces de lectura i enviament de dades. L'usuari pot aturar aquest proces amb la introducció de la comanda “**ATURAR LLUM**” , que enviarà un codi “**7**” emmagatzemat a “**arg1**” a la mota i l'esdeveniment d'aquest component rebrà. Adjuntem diagrama de flux de l'esdeveniment “**Receive.receive**” del component:

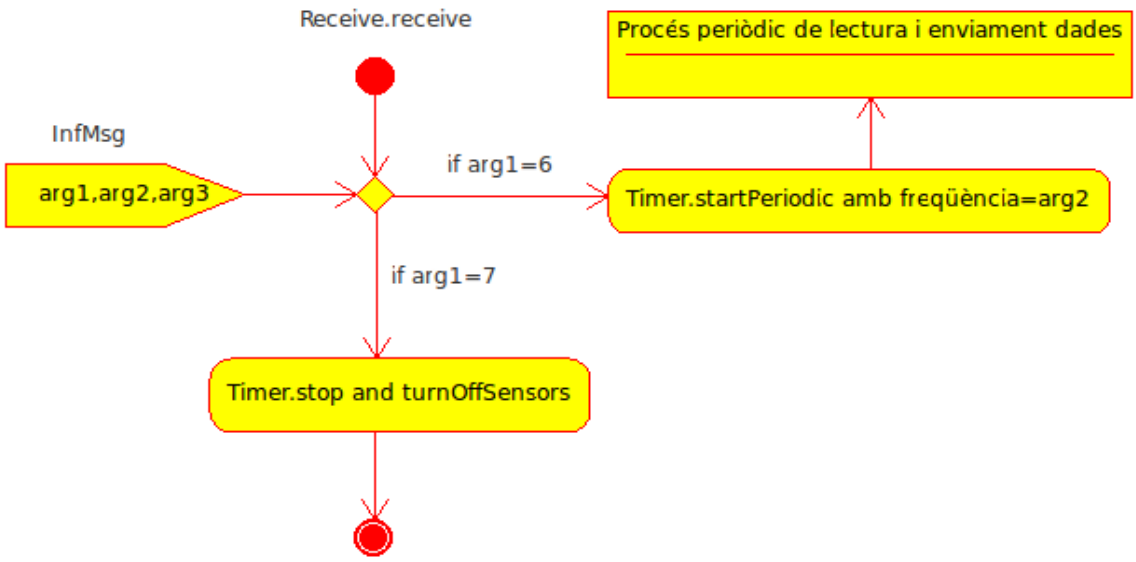


Fig.nº28 : Diagrama flux de l'esdeveniment “Receive.receive” en “PhotoReporter”.

4.2.4 Component “MacReporter”

El component “**MacReporter**”, conté totes les interfases dels components anteriors, excepte les que pertanyen al convertidor d'analògic a digital, que son “**LecturaAdc**” i “**Atm128AdcConfig**”. En canvi conté la interfase “**LocalIeeeEui64**”, que ens permet llegir la direcció mac de la mota. Igual que els components anteriors utilitza dos tipus de missatges. Missatge tipus “**MacMsg**” per l'enviament de la direcció mac de la mota, definit en el fitxer “**Msgs.h**” amb la estructura:

Arguments missatge "MacMsg":	**Explicació:**
"moteId"	Camp identificació de la mota que envia el missatge.
“macAddr[8]”	Direcció mac de la mota que envia el missatge.

També utilitza el missatge tipus “**InfMsg**” per rebre el codi emmagatzemat en “**arg1**”, que activarà el proces d'enviament de la direcció mac de la mota remota. En aquest cas quan l'esdeveniment “**Receive.receive**”, rep un missatge amb el valor “arg1”=”**1”,** dispara el procediment “**Timer.StartOneShot**”, que enlloc d'iniciar un proces periòdic d'enviament de dades, nomes envia una vegada la direcció mac i el temporitzador s'atura. En el codi del component llancem el procediment “**Timer.StartOneShot(500)**”, això executa el temporitzador i després de 0.5 segons s'executa l'esdeveniment “**Timer.Fired()**”, que executa els procediments de la interfase “**AMSend**” anomenat “**AMSendMac.send”,** per l'enviament de la direcció mac. Adjuntem diagrama de flux de l'esdeveniment “**Receive.receive**” :

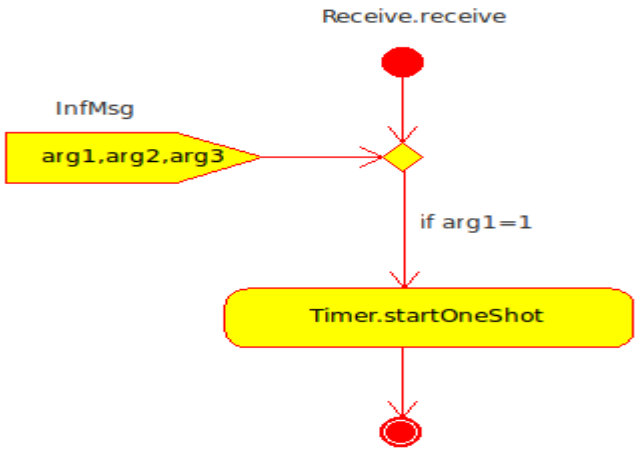


Fig.nº 29 : Diagrama flux de l'esdeveniment “Receive.receive” en “MacReporter”.

4.2.5 Component “Blink”

Per últim el component “**Blink**”, conte les interfases “**Boot**”, “**Timer**” i “**Leds**”, que ja s'ha esmentat anteriorment. Aquest component inclou el components anteriors i és una especie de **“Main**” de l'aplicació “**MotaRemota**” que, com el seu nom indica, resideix en la mota remota. Apart de la funció esmentada, aquest component, quan la mota remota rep l'alimentació de les bateries, amb l'esdeveniment “**Boot.booted**”, que arranca l'aplicació, llença el procediment “**Timer0.startPeriodic(500)**”, que te la funció de llençar el temporitzador i fer parpellejar el “led” de color groc cada mig segon. El “led” s'activa en l'esdeveniment “**Timer0.fired**”, que llença el procediment “**Leds.led1Toggle()**”.

5. Viabilitat tècnica

Aquest projecte consisteix en el treball, la experimentació i implementació de dos aplicacions bàsiques que exploten les possibilitats d'un sistema bàsic de dos sensors sense fils. S'adapta a un entorn acadèmic per l'aprenentatge principalment del llenguatge “**nesC**” , que és el que s'utilitza per programar la major part de les plataformes de sensors sense fils existents en el mercat. La viabilitat d'aquest projecte comportaria una serie de condicions que s'esmenten en l'apartat següent, quan es parla de tasques importants que encara queden per desenvolupar. Es a dir evolucionar el projecte al mon real, al context físic d'una sala de servidors de qualsevol empresa, suposaria el pas següent a desenvolupar.

6. Conclusions

Principalment l'obstacle més important ha sigut l'aprenentatge d'aquesta tecnologia, amb l'avantatge que "**TinyOS**" , és un sistema operatiu de programari lliure i podem trobar l'ajuda de la comunitat de programadors que utilitzen aquest sistema per programar motes de sensors de baix cost i pocs recursos de comunicació i processament. Les aplicacions que aquesta tecnologia pot tenir son, gaire bé, infinites en diferents camps, però el cost d'aprenentatge i desenvolupament és alt, valorat en relació al temps que necessitem. Aquest projecte és pràcticament un experiment de laboratori i s'allunya bastant d'arribar a ser una aplicació real per a qualsevol empresa mitjana. Problemes trobats en el desenvolupament del projecte han sigut bastants, però esmentem principalment:

- Instal·lació de totes les eines necessàries de desenvolupament per a que tot el sistema funciones amb els exemples. S'ha d'estar familiaritzat amb sistemes Linux.
- Afegir un sistema anti-caigudes per a que la mota fes un reinicie, en cas de caiguda, es a dir un sistema de "**watchdog**", que en el nostre cas no s'ha sabut com implementar.
- Aïllar cada component, es a dir cada component dedicar-lo a un sensor de la mota.
- Implementar el tipus de missatge encarregat de configurar les formes de treballar dels sensors de la mota remota. Es a dir, el missatge que rep la mota remota amb els paràmetres.

Per últim, queden encara importants tasques a desenvolupar, que s'esmenten a continuació:

- Evolucionar l'aplicació desenvolupada per ser capaç de comunicar-se amb mes quantitat de motes i de diferents models, clar esta, compatibles les unes amb les altres. Les motes estarien instal·lades en punts estratègics de les sales de servidors.
- Ampliar el sistema d'explotació amb més funcionalitats i el sistema de persistència de les dades amb mes filtres de cerca i més taules.
- Interfase gràfica, enlloc de l'aplicació per consola, per això tenim diverses tecnologies com Swing de Java, aplicació web amb Java, amb Php, Python, Gambas2, etc.
- Gestió massiva de les dades amb gràfiques de variabilitat de les diferents tipus de lectures, amb una interpretació de les mateixes, per detectar possibles problemes referents als sistemes de refrigeració de les sales de servidors.

7. Glossari

- **ADC :** Convertidor de senyal analògica a digital.
- **GPIO :** Portes programables d'entrada/sortida de propòsit general d'un microcontrolador.
- **EEPROM :** Memòria del tipus programable i esborrable nomes de lectura.
- **IEEE :** Institut d'enginyeria elèctrica i electrònica.
- **ISM :** Banda lliure de radiofreqüència (Industrial-Scientific-Medical).
- **CMOS :** (Complementary Metal Oxid Semiconductor), família de portes lògiques utilitzades per a la fabricació de circuits integrats.
- **RISC :** (Reduced Instruction Set Computer), tecnologia utilitzada en certs microprocessadors dedicats que conten amb un reduït conjunt d'instruccions.
- **RAM :** Tipus de memòria d'accés aleatori.
- **UART :** (Universal Asynchronous Receiver/Transmiter), transceptor universal de comunicació serie asíncron.
- **USART :** (Universal Synchronous/Asynchronous Receiver/Transmiter), transceptor universal de comunicació serie síncron/asíncron.
- **USB :** (Universal Serial Bus), connector serie universal.
- **MAC :** (Medium Acces Control Layer), capa de control d'accés al medi.
- **JTAG :** (Joint Test Action Group),estàndard 'IEEE 1149.1' , s'utilitza per a la programació i depuració 'on chip' dels microcontroladors.
- **SPI :** (Serial Peripheral Interface), estàndard de comunicacions utilitzat principalment per a la transferència d'informació entre circuits integrats.

8. Bibliografia

[1] Web de l'assignatura:
http://cv.uoc.edu/app/mediawiki14/wiki/P%C3%A0gina_principal

[2] Web de TinyOS:
http://www.tinyos.net/.

[3] Protocol de comunicació:
http://es.wikipedia.org/wiki/ZigBee

[4] Projectes de "Tecnológico Fundació Deusto":
http://www.tecnologico.deusto.es

[5] Dades tècniques microcontrolador Atmega 1281:
http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf

[6] Dades tècniques sensor lluminositat:
http://www.advancedphotonix.com/ap_products/pdfs/PDV-P9003-1.pdf

[7] Dades tècniques sensor temperatura:
http://ww1.microchip.com/downloads/en/DeviceDoc/21942e.pdf

[8] Yeti 2. Plugin de "TinyOS" para Eclipse:
http://tos-ide.ethz.ch/wiki/index.php

[9] Llenguatge de programació "NesC":
www.nesc.sourceforge.net

[10] Web d'Atmel:
http://www2.atmel.com/

9. Annexos

9.1 Manual d'usuari

La interfície d'usuari correspon a l'aplicació "**ControlAmbiental**". En aquest punt farem una descripció de les funcionalitats d'aquesta aplicació, amb exemples d'execució. Aquesta aplicació desenvolupada en Java és la encarregada de interactuar, mitjançant una connexió de socket TCP/IP i la utilitat "**SerialForwader**", amb les motes, enviant missatges de configuració a la mota remota i reben lectures d'aquesta. Al mateix temps també afegeix funcionalitats de manteniment amb la base de dades. El seu funcionament és interactiu mitjançant una sèrie de comandes, les quals anirem explicant detalladament. En el moment de l'execució de l'aplicació, aquesta ens demana una primera comanda i depèn de la comanda que introduïm, podem tenir fins a una tercera petició, es a dir, el programa analitzarà la comanda introduïda i demanarà les dades necessàries. El nom de les comandes és prou significatiu sobre les tasques que executen i el text es pot introduir tant en lletres majúscules com en lletres minúscules.

- Comanda "**AJUDA**" : La comanda més necessària, que com el seu nom indica ens presenta un resum per consola de totes les comandes i una petita explicació de les tasques que realitza:

```
==================================
CONTROL -  AMBIENTAL      versió 1.1
==================================
----------------------------------
Per l'ajuda introduïu: AJUDA
----------------------------------
Introduïu comanda:
ajuda
------------------------------------------------------------
     Entrada de comandes:
------------------------------------------------------------
     Primera:             Segona:           Tercera:
------------------------------------------------------------
(1)  ---                  ---                ---
(2)  MAC                  ---                ---
(3)  TEMPERATURA          Freqüència         ---
(4)  BATERIA              Freqüència         ---
(5)  LLUM                 Freqüència         ---
(6)  ATURAR TEMPERATURA   ---                ---
(7)  ATURAR BATERIA       ---                ---
(8)  ATURAR LLUM          ---                ---
(9)  ALARMA BATERIA       Voltatge mínim     ---
(10) ALARMA TEMPERATURA   Temp. mínima       Temp. màxima.
(11) TAULA TEMPERATURA    Data registres     ---
(12) TAULA BATERIA        Data registres     ---
(13) TAULA LLUMINOSITAT   Data registres     ---
(14) ESBORRAR TEMPERATURA ---                ---
(15) ESBORRAR BATERIA     ---                ---
(16) ESBORRAR LLUMINOSITAT ---               ---
(17) APLIC GRAF           ---                ---
------------------------------------------------------------
     Resultats:
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
(1)  Modalitat d'escoltar per port serie, per sortir 'Ctrl-z'
(2)  Demanem la 'mac' a la mota remota.
(3)  Demanem lectures temperatura amb una determinada freqüència de mostreig (min.1500 milisegons).
(4)  Demanem lectures bateria amb una determinada freqüència de mostreig (min.1500 milisegons).
(5)  Demanem lectures lluminositat amb una determinada freqüència de mostreig (min.1500 milisegons).
(6)  Tanca lectures de temperatura (atura sensor temperatura).
(7)  Tanca lectures de bateria (atura sensor bateria).
(8)  Tanca lectures de lluminositat (atura sensor lluminositat).
(9)  Mostra lectures bateria, quan el valor és igual o inferior al voltatge mínim.
(10) Mostra lectures de temperatura quan el valor és igual o menor a la temp.mínima o igual o mes gran que la temp.màxima.
(11) Mostra lectures de temperatura registrades a la base de dades amb una determinada data.Si la data es deixa en blanc, surten tots el registres.
(12) Mostra lectures de bateries registrades a la base de dades amb una determinada data.Si la data es deixa en blanc, surten tots el registres.
(13) Mostra lectures de lluminositat registrades a la base de dades amb una determinada data.Si la data es deixa en blanc, surten tots el registres.
(14) Esborra tots els registres de temperatura de la base de dades.
(15) Esborra tots els registres de valors de les bateries de la base de dades.
(16) Esborra tots els registres de lluminositat de la base de dades.
(17) Petita aplicació gràfica per la visualització de les taules de lectures de la base de dades.
(---)Cap entrada,en blanc.
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Fig.nº30: Resultat comanda "AJUDA", amb totes les comandes.

- Comanda “**MAC**” : Demana a la mota remota la seva direcció de mac i l'aplicació queda en estat d'escolta, per sortir premem 'Crtl-z' .Aquesta funcionalitat no te persistència de dades.

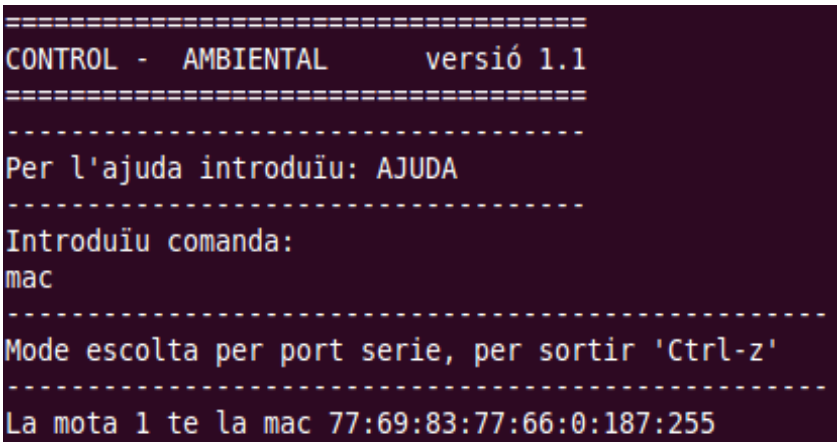


Fig.nº 31: Resultat comanda “MAC”.

- Comanda “**TEMPERATURA**” : Demana lectures de temperatura a la mota remota, amb una determinada freqüència de mostreig en mili segons. L'aplicació queda en estat d'escolta reben lectures que son inserides en la taula adient de temperatures i mentre no aturem el temporitzador del sensor de temperatura, anirem reben aquestes lectures.

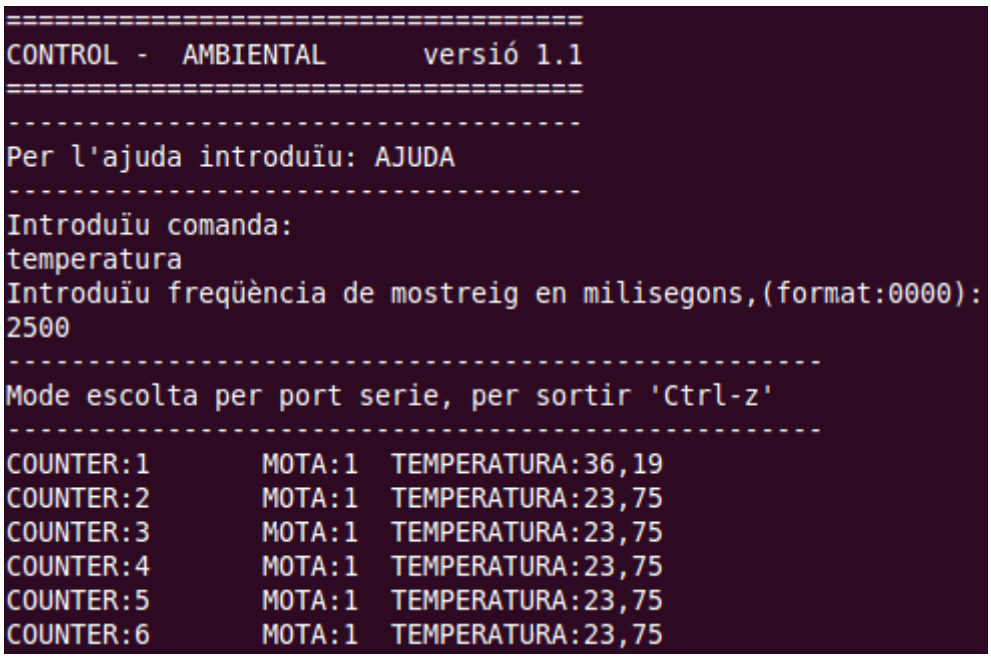


Fig.nº 32: Resultat comanda “TEMPERATURA”.

- Comanda “**BATERIA**”: Demana lectures del valor de les bateries a la mota remota, amb una determinada freqüència de mostreig en mili segons. L'aplicació queda en estat d'escolta reben lectures que son inserides en la taula adient de valor de bateries i mentre no aturem el temporitzador del sensor de bateries, anirem reben aquestes lectures.

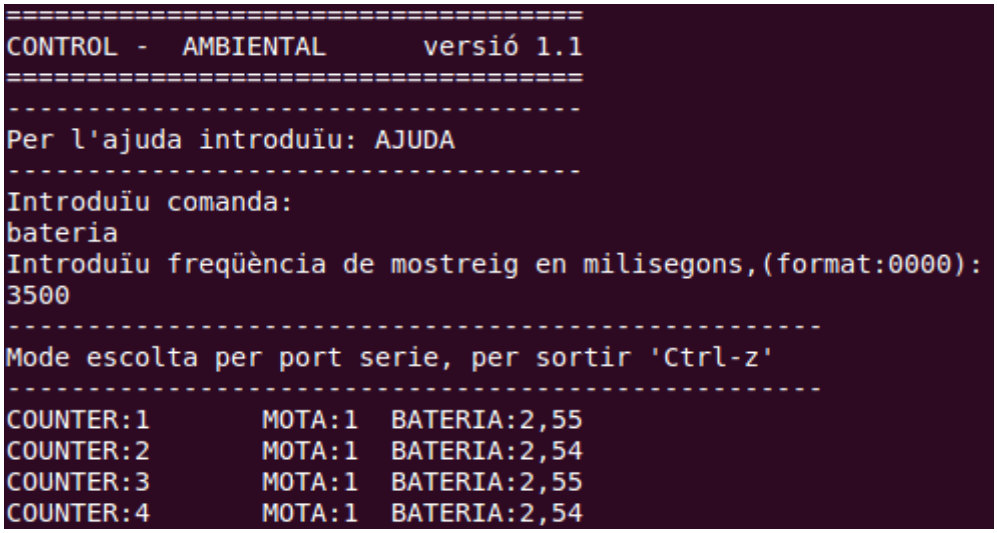


Fig.nº 33: Resultat comanda “BATERIA”.

- Comanda “**LLUM**”: Demana lectures de lluminositat a la mota remota, amb una determinada freqüència de mostreig en mili segons. L'aplicació queda en estat d'escolta reben lectures que son inserides en la taula adient de lluminositats i mentre no aturem el temporitzador del sensor de llum, anirem reben aquestes lectures.

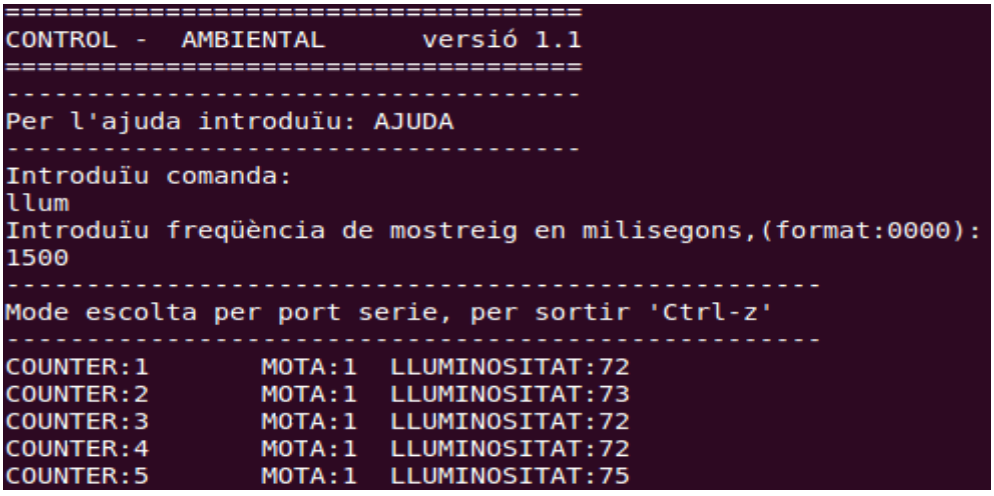


Fig.nº 34: Resultat comanda “LLUM”.

- Comandes “**ATURAR TEMPERATURA**”, “**ATURAR BATERIA**” i “**ATURAR LLUM**”: Aquestes tres comandes les detallarem juntes, doncs la seva tasca és aturar els respectius sensors de la mota remota, es a dir, parem de rebre les lectures adients quan l'aplicació esta en estat d'escolta. En realitat envia un codi a la mota remota que fa aturar el temporitzador i tancar el sensor corresponent. Si hem activat en un principi, un determinat sensor amb algunes de les comandes següents com “**TEMPERATURA**”, “**LLUM**” o “**BATERIA**”, mentre no aturem el sensor corresponent amb les comandes d'aturar, l'aplicació en estat d'escolta, continuarà reben lectures, que seran guardades a la base de dades.
- Podem activar els tres tipus de sensor i rebre les tres tipus de lectures, per posteriorment anant aturant consecutivament el sensor que ens interessi. En aquest exemple presentem com l'aplicació, en estat d'escolta, rep les tres tipus de lectures amb diferents freqüències de mostratge:

```
-----------------------------------------------------
Mode escolta per port serie, per sortir 'Ctrl-z'
-----------------------------------------------------
COUNTER:18      MOTA:1  BATERIA:2,50
COUNTER:14      MOTA:1  TEMPERATURA:22,81
COUNTER:15      MOTA:1  TEMPERATURA:22,81
COUNTER:19      MOTA:1  BATERIA:2,50
COUNTER:16      MOTA:1  TEMPERATURA:22,81
COUNTER:1       MOTA:1  LLUMINOSITAT:198
COUNTER:20      MOTA:1  BATERIA:2,50
COUNTER:17      MOTA:1  TEMPERATURA:22,81
COUNTER:18      MOTA:1  TEMPERATURA:22,81
COUNTER:21      MOTA:1  BATERIA:2,50
COUNTER:19      MOTA:1  TEMPERATURA:22,81
COUNTER:2       MOTA:1  LLUMINOSITAT:198
COUNTER:20      MOTA:1  TEMPERATURA:22,81
COUNTER:22      MOTA:1  BATERIA:2,49
```

Fig.nº 35: Lectures amb diferents freqüències de mostratge.

- Comanda “**ALARMA BATERIA**”: Per utilitzar aquesta comanda , abans hem hagut d'activar el sensor de bateries amb la comanda “BATERIA”, doncs en cas contrari no rebrem cap lectura. Aquesta comanda d'alarma ens demana introduir un voltatge mínim, per sota del qual ens arribaran lectures de bateries. Si el valor de la lectura es superior al voltatge mínim indicat no rebrem cap lectura d'alarma. En aquest cas d'exemple, com sospitem que les bateries de la mota remota estan baixes, indiquem un voltatge mínim de 2.60 i veiem que es reben lectures d'alarma:

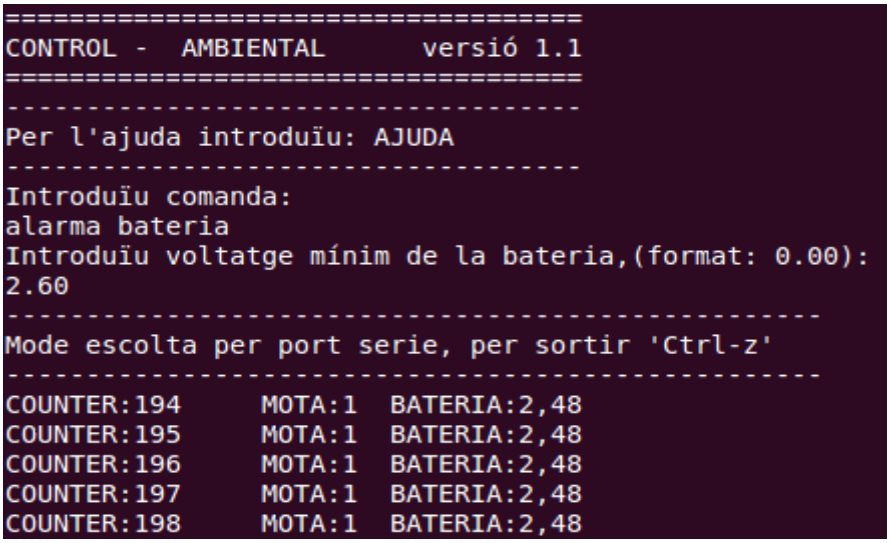


Fig.nº 36: Resultats comanda "ALARMA BATERIA".

- Comanda "**ALARMA TEMPERATURA**": Igual que en el cas anterior, per utilitzar aquesta comanda hem hagut d'activar el sensor de temperatura amb la comanda "**TEMPERATURA**". Aquesta comanda d'alarma ens demana un rang de temperatures, es a dir una temperatura mínima i una altre màxima, si la temperatura que llegeix el sensor esta dintre d'aquest rang, no es reben lectures, en canvi si la lectura surt del rang establert es reben lectures d'alarma. En el següent exemple, no es reben lectures de temperatura, fins que la temperatura del sensor no surt del rang indicat, apropant-lo a una font de calor:

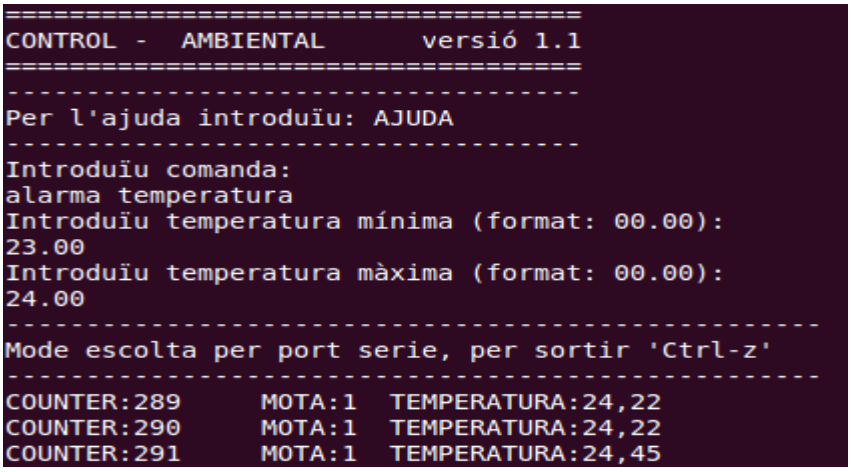


Fig. nº 37 : Resultats de la comanda "ALARMA TEMPERATURA".

- Comanda “**TAULA TEMPERATURA**” : Arribem a les comandes que connecten amb la base de dades i ens mostren les dades rebudes. Aquesta comanda ens demanarà que introduïm una determinada data, amb un format determinat i ens mostrarà les lectures de temperatura rebudes en la data indicada. Al mateix temps, si deixem en blanc la data , mostrarà totes les lectures de temperatura que conté la base de dades. La capçalera conté el codi del registre,la mota, temperatura i la data amb la precisió de mili segons. Mostrem un exemple:

```
====================================
CONTROL -  AMBIENTAL      versió 1.1
====================================
------------------------------------
Per l'ajuda introduïu: AJUDA
------------------------------------
Introduïu comanda:
taula temperatura
Introduïu la data dels registres de temperatura a mostrar .(format: DD/MM/YYYY):
26/05/2011
Còdig  Mota Temperatura  Data
------------------------------------------------------------
672       1     22.81      26/5/2011 18:17:27:169
673       1     22.81      26/5/2011 18:17:28:653
674       1     23.04      26/5/2011 18:17:30:94
675       1     22.81      26/5/2011 18:17:31:556
676       1     22.81      26/5/2011 18:17:46:210
677       1     22.81      26/5/2011 18:17:47:673
678       1     22.81      26/5/2011 18:17:49:133
679       1     22.81      26/5/2011 18:17:50:604
680       1     22.81      26/5/2011 18:17:52:65
681       1     22.81      26/5/2011 18:17:53:525
682       1     22.81      26/5/2011 18:17:54:986
683       1     23.04      26/5/2011 18:22:48:360
684       1     23.04      26/5/2011 18:22:49:426
685       1     22.81      26/5/2011 18:22:50:900
686       1     23.04      26/5/2011 18:22:52:343
687       1     23.04      26/5/2011 18:22:53:813
688       1     23.04      26/5/2011 18:22:55:266
689       1     22.81      26/5/2011 18:22:56:732
690       1     22.81      26/5/2011 18:22:58:205
691       1     23.04      26/5/2011 18:22:59:661
```

Fig. nº 38: Resultat comanda “TAULA TEMPERATURA”.

- Les comandes “**TAULA BATERIA**” i “**TAULA LLUMINOSITAT**”, tenen el mateix comportament que la comanda anterior, mostrant respectivament lectures de bateries i de lluminositat d'una determinada data, que si deixem en blanc, mostren totes les lectures de la base de dades. Presentem dos exemples:

```
==================================
CONTROL -  AMBIENTAL      versió 1.1
==================================
----------------------------------
Per l'ajuda introduïu: AJUDA
----------------------------------
Introduïu comanda:
taula bateria
Introduïu la data dels registres de valors de la bateria a mostrar .(format: DD/MM/YYYY):
26/05/2011
Còdig  Mota Bateria  Data
-------------------------------------------------------
640      1    2.91     26/5/2011 18:17:4:796
641      1    2.5     26/5/2011 18:17:6:628
642      1    2.5     26/5/2011 18:17:9:54
643      1    2.49     26/5/2011 18:17:11:507
644      1    2.5     26/5/2011 18:17:13:938
645      1    2.5     26/5/2011 18:17:26:549
646      1    2.5     26/5/2011 18:17:28:588
647      1    2.5     26/5/2011 18:17:31:32
648      1    2.5     26/5/2011 18:17:46:54
649      1    2.5     26/5/2011 18:17:48:126
650      1    2.5     26/5/2011 18:17:50:558
```

Fig.nº 39: Resultats comanda “TAULA BATERIA”.

```
==================================
CONTROL -  AMBIENTAL      versió 1.1
==================================
----------------------------------
Per l'ajuda introduïu: AJUDA
----------------------------------
Introduïu comanda:
TAULA LLUMINOSITAT
Introduïu la data dels registres de lluminositat a mostrar .(format: DD/MM/YYYY):
26/05/2011
Còdig  Mota Lluminositat  Data
-------------------------------------------------------------
596      1    198     26/5/2011 18:17:50:335
597      1    198     26/5/2011 18:17:54:721
598      1    193     26/5/2011 18:22:49:151
599      1    192     26/5/2011 18:22:53:541
600      1    192     26/5/2011 18:22:57:931
601      1    191     26/5/2011 18:23:11:136
602      1    192     26/5/2011 18:23:15:519
```

Fig.nº 40: Resultats comanda “TAULA LLUMINOSITAT”.

- Les comandes "**ESBORRAR TEMPERATURA**", "**ESBORRAR BATERIA**" i "**ESBORRAR LLUMINOSITAT**", com el seu nom indica, esborren les dades de les respectives taules a la base de dades, on son emmagatzemades les lectures de temperatura, bateries i lluminositat. Hem de tenir precaució amb aquestes comandes, ja que, eliminen totes les dades de lectures d'un determinat sensor. Mostrarem un exemple amb les lectures de temperatura:

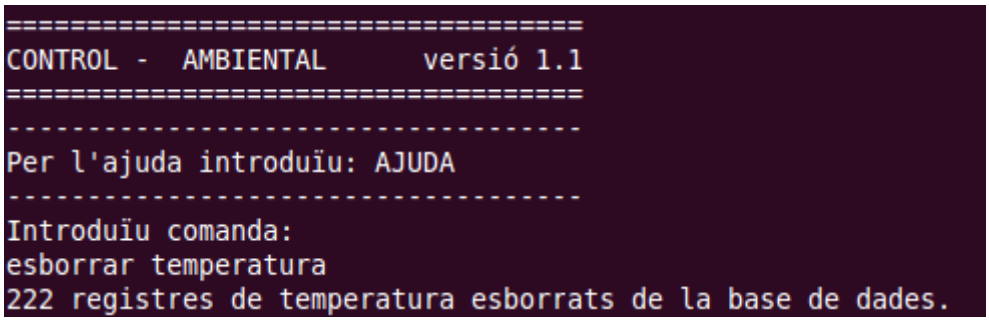


Fig.nº 41: Resultats comanda "ESBORRAR TEMPERATURA".

- Comanda "**APLIC GRAF**": Amb aquesta comanda iniciem una petita aplicació gràfica, que ens mostrarà les tres tipus de lectures d'una forma més propera al usuari:

Fig.nº 42: Aplicació gràfica.

A continuació passem a esmentar les funcionalitats de l'aplicació gràfica. Primerament haurem de connectar amb la base de dades premem el botó "**Connecta**". Les dades de connexió ja venen configurades i no s'han de canviar, excepte que al instal·lar el sistema gestor de base de dades MySql, haguérem canviat algun paràmetre de connexió. Al connectar amb la base de dades veiem que s'activen els botons del quadre "**Dades Sensors**", situats a la dreta amb el nom "**Executa**". El seu funcionament és intuïtiu i mostrem tres exemples d'execució:

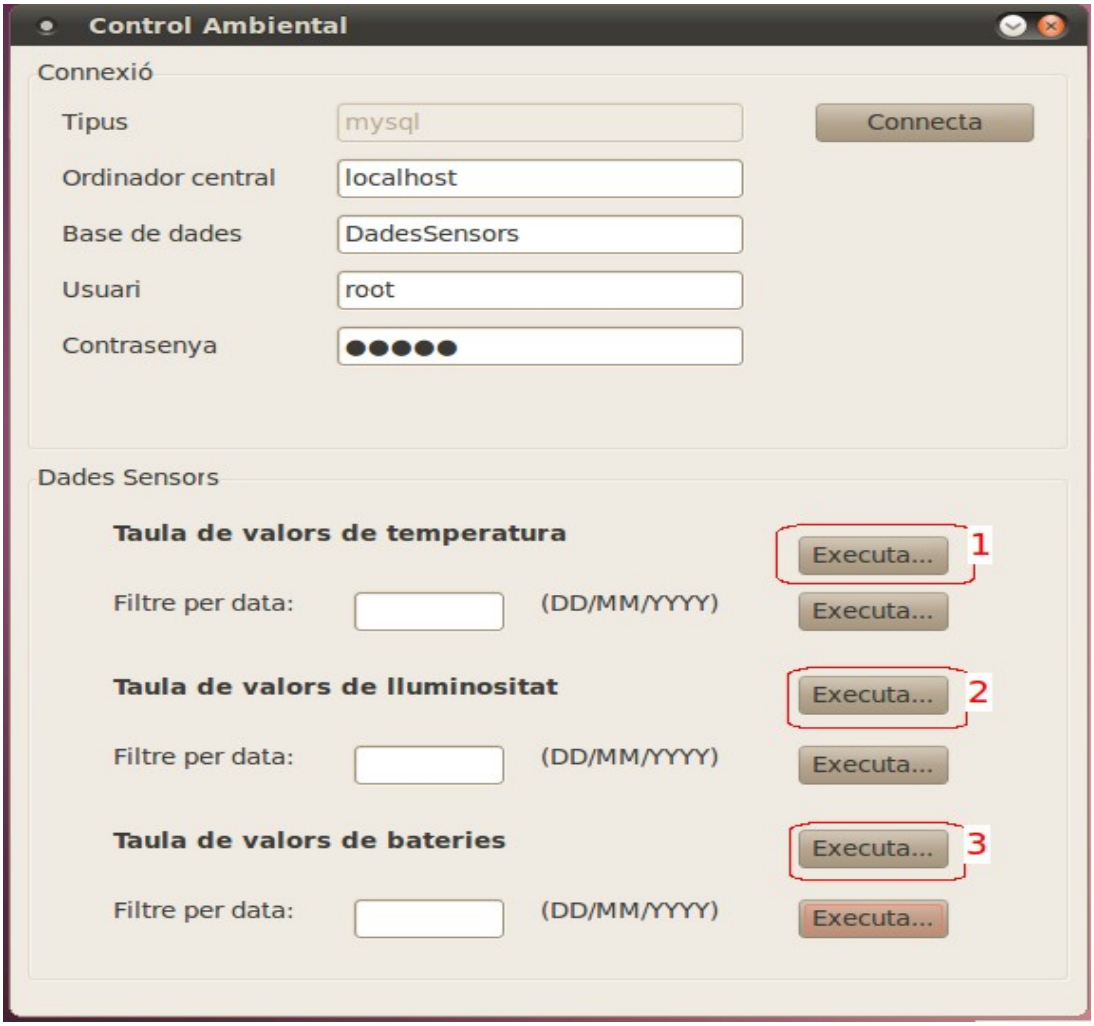


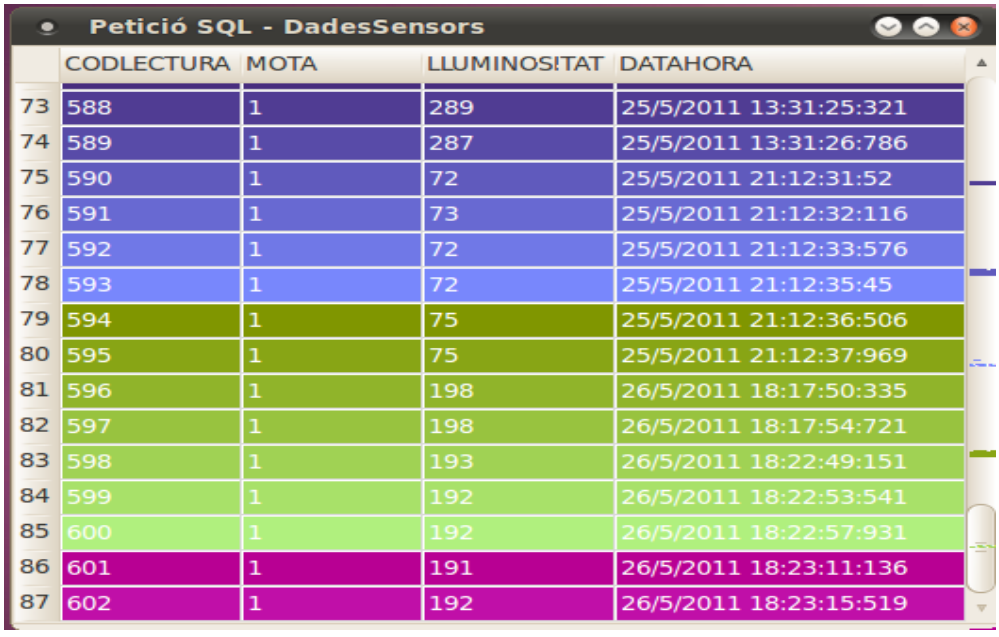
Fig.nº 43: Execució de l'aplicació gràfica.

Si premem el botó “Executa” etiquetat amb 1, obtindrem la taula de lectures de temperatura**:**

	CODLECTURA	MOTA	TEMPERATURA	DATAHORA
1	752	1	23.04	26/5/2011 19:33:4:251
2	753	1	23.28	26/5/2011 19:33:6:290
3	754	1	23.04	26/5/2011 19:33:8:728
4	755	1	23.04	26/5/2011 19:33:11:174
5	756	1	23.04	26/5/2011 19:33:13:606
6	757	1	23.04	26/5/2011 19:33:16:57
7	758	1	23.04	26/5/2011 19:33:18:484
8	759	1	23.04	26/5/2011 19:33:20:927
9	760	1	23.04	26/5/2011 19:33:23:373
10	761	1	23.04	26/5/2011 19:33:25:817
11	762	1	23.04	26/5/2011 19:33:28:252
12	763	1	23.04	26/5/2011 19:33:30:693
13	764	1	23.04	26/5/2011 19:33:33:125
14	765	1	23.04	26/5/2011 19:33:35:571

Fig. nº 44: Taula de lectures de temperatures.

Si premem el botó “Executa” etiquetat amb 2, obtindrem la taula de lectures de lluminositat:



	CODLECTURA	MOTA	LLUMINOSITAT	DATAHORA
73	588	1	289	25/5/2011 13:31:25:321
74	589	1	287	25/5/2011 13:31:26:786
75	590	1	72	25/5/2011 21:12:31:52
76	591	1	73	25/5/2011 21:12:32:116
77	592	1	72	25/5/2011 21:12:33:576
78	593	1	72	25/5/2011 21:12:35:45
79	594	1	75	25/5/2011 21:12:36:506
80	595	1	75	25/5/2011 21:12:37:969
81	596	1	198	26/5/2011 18:17:50:335
82	597	1	198	26/5/2011 18:17:54:721
83	598	1	193	26/5/2011 18:22:49:151
84	599	1	192	26/5/2011 18:22:53:541
85	600	1	192	26/5/2011 18:22:57:931
86	601	1	191	26/5/2011 18:23:11:136
87	602	1	192	26/5/2011 18:23:15:519

Fig. nº 45: Taula de lectures de lluminositat.

Si premem el botó “**Executa**” etiquetat amb 3, obtindrem la taula de lectures de valors de bateries:

	CODLECTURA	MOTA	BATERIA	DATAHORA
1	704	1	2.46	26/5/2011 19:34:23:344
2	705	1	2.46	26/5/2011 19:34:25:388
3	706	1	2.46	26/5/2011 19:34:27:823
4	707	1	2.46	26/5/2011 19:34:30:274
5	708	1	2.46	26/5/2011 19:34:32:702
6	709	1	2.46	26/5/2011 19:34:35:154
7	710	1	2.46	26/5/2011 19:34:37:584
8	711	1	2.46	26/5/2011 19:34:40:24
9	712	1	2.46	26/5/2011 19:34:42:465
10	713	1	2.46	26/5/2011 19:34:44:911

Fig.nº 46: Taula de lectures de valor de les bateries.

Els botons que es troben sota els botons etiquetats, mostren el mateix resultat però filtrant les lectures per una determinada data, la qual s'ha d'introduir amb el format indicat.