Sistema de Control Ambiental Flexible per a la Indústria Agroalimentària (SCAFIA)

Joan Carles Martínez Rodríguez
Enginyeria Tècnica en Informàtica de Sistemes

Consultor: Jordi Bécares Ferrés

Lliurament: 10/06/2011

A Eugènia, per la teva paciència, comprensió i suport aquests anys d'estudi.

A la meva familia i amics.

Entre tots m'heu ajudat a superar els mals moments, i arribar al final.

I no m'oblido de tu Mixu, fidel i silenciosa companya d'estudis. M'has fet bona companyia molts anys, fins on vas poder. Sempre recordaré les nits d'estudi en les que m'acompanyaves fins que, ja cansat, decidia parar màquines i marxar a dormir...

Terrassa, 10 de juny de 2011

Resum

El present treball està basat en el desenvolupament d'un sistema que permeti fer una demostració pràctica útil amb els materials que l'àrea de sistemes encastats posa a la nostra disposició. De forma resumida es tracta de motes/nodes amb possibilitat de comunicació inhalàmbrica entre ells i un sistema operatiu TinyOS programable amb el llenguatge nesC.

De possibilitats de dissenyar sistemes viables amb les eines anteriors n'hi han moltes, i en aquest cas s'ha triat per desenvolupar un sistema de control climàtic, dissenyat segons les necessitats d'un hivernacle, de forma que es recull periòdicament les dades que els nodes/motes recopilen amb els sensors que porten integrats, i les transmeten a un programari instal·lat a un PC on es fan les següents accions:

- Portar un registre d'auditoria tant dels valors climàtics, alarmes del sistema així com els accessos al sistema per variar valors climàtics, accionament de mecanismes, i comunicacions amb les motes/nodes. També es guarden les dades climàtiques en un format que permet el seu anàlisis estadístic

- Presentar una possible solució viable per controlar automàticament els valors climàtics segons les dades rebudes, i també permetre l'accionament a voluntat de mecanismes de control

A més s'ha volgut augmentar la capacitat de sensors afegint un que permet monitoritzar la humitat relativa, ja que la mota/node integra un de lluminositat, tensió i temperatura, a més d'un sensor Hall asíncron.

La voluntat per tant es la de presentar un sistema viable autònom que permet monitoritzar de forma contínua els valors climàtics de l'hivernacle, i que a més porta un registre d'auditoria per fer una anàlisi de les condicions climàtiques en qualsevol moment, i de les accions que s'han fet per mantenir-les.

Paraules clau: **TinyOS, nesC, sistema encastat, Zigbee, control climàtic, xarxa inhalàmbrica, sensor, WSN**

Àrea TFC: **Sistemes encastats**

Índex de continguts

Índex de figures

1. Introducció

1.1 Justificació

El control ambiental esdevé cabdal en la maximització de la producció, i en la disminució del risc de pèrdues en la indústria agroalimentària. Podem trobar clars exemples en:

- **Producció agrícola mitjançant hivernacles**. Si considerem la producció de tomàquets, aquesta necessita un ambient controlat entre els 20-30º C durant el dia i 10-17ºC durant la nit segons l'espècie, a més d'una humitat relativa entre els 60-80%. Aquestes característiques són variables i no són les mateixes durant la maduració, que durant la pol·linització

- **Estabular animals per a producció**. El cas concret de la producció d'ous mitjançant gallines engabiades, o en llibertat però en recinte tancat, fa necessari un control exhaustiu de les condicions ambientals per evitar una excessiva mortalitat, i millorar les condicions de producció (tant en producció neta com en reducció de costos per alimentació). Alguns autors (Marslen i Morris, 1987) estimen els següents valors:
  - Temperatura de confort: entre els 18.0 i els 25.0. Valors superiors o inferiors tenen efecte positiu o negatiu sobre el consum d'aliment, la superació dels 21ºC porten a un menor consum i en conseqüència a la disminució dels ous produïts
  - La temperatura letal mínima és 35,5ºC

- **Emmagatzematge de la producció**. Exigeix un control ambiental molt precís, on fins i tot sol controlar-se la composició de l'aire, i on la temperatura i la humitat sol fixar-se depenen del producte a emmagatzemar (fruites, llegums, llavors, ...)

Veiem doncs que el control exhaustiu de les condicions ambientals pot ser la diferència entre guanys i pèrdues en la producció, o del producte ja produït. Aquest control pot fer-se de forma automatitzada amb la instal·lació dels sensors corresponents dins del lloc a controlar, i un control extern que rebi les seves lectures. Aquest control hauria de reaccionar davant valors límits, ja sigui:

- Emetent un senyal d'alarma
- Executant un pla de contramesures ambiental
- Comprovant l'eficàcia de la contramesures ambiental executada

Aquest control també hauria de registrar els valors dels sensors rebuts cada cert interval de temps, i composar un dietari (log) que pugui ser consultable. Així es pot disposar de dades per identificar el moment òptim productiu o d'emmagatzematge, o fins i tot de validació del procés de control ambiental davant d'una auditoria per danys

1.2 Descripció del projecte

El que es pretén es aconseguir un sistema d'instal·lació flexible de sensors, i una monitorització constant del sistema ambiental de la producció, limitat en aquest cas als valors de temperatura, lluminositat, accionaments i humitat. Amb la monitorització constant es reforçarà el control ambiental, i el sistema exercirà com a auditor de les condicions ambientals en el lloc d'instal·lació. Una utilitat serà el demostrar que s'han respectat les condicions de climatització en tot moment, i també el d'obtenir alarmes de incompliment per tal d'aplicar mesures correctores.

Un altre objectiu és fer que el programari prengui decisions operatives per mantenir unes condicions ambientals òptimes, i que aquestes es desenvolupin de forma efectiva. Com s'ha dit l'objectiu principal del programari al PC de control serà enregistrar els valors, i permetre la seva consulta, però s'estudiarà de quina manera es podria afegir una interfície de control de tipus industrial, que permeti el control d'un sistema de ventilació forçada, alarmes sonores i lluminoses, control d'humitat i de finestres que permeti un control automatitzat de les condicions ambientals. A més es permetrà fer-ho també de forma manual.

Per major claredat els objectius anteriors s'aplicaran a un projecte de control i d'auditoria ambiental d'hivernacles de qualsevol mida i forma, tot i que amb petites modificacions es podria utilitzar en altres instal·lacions agroalimentàries:

- Els sistemes de sensors sense fils permeten el seu desmuntatge fàcilment per procedir, entre collita i collita, a la retirada de restes, desmuntatge de bastidors, preparació del sòl, ... Per exemple es poden instal·lar a l'extrem d'una pica que es clavarà al terra on es cregui més convenient, al portar la seva pròpia alimentació elèctrica no és necessària cap instal·lació addicional que el faria molt estàtic.

- El sensor de llum permet establir si es de dia o de nit, i tenir-ho en compte per al control ambiental. Així es pot decidir obrir o no airejadors/finestres, enlloc de forçar mecànicament la refrigeració, o bé activar rec per vaporització

- El sensor d'accionaments farà servir el sensor d'efecte Hall. Així amb l´ús de solenoides, si no hi ha una altra manera, que s'activin a la vegada que el dispositiu, es pot obtenir una lectura de posició del mecanisme. També es podria obtenir aquesta lectura de forma mecànica, lligada al moviment del mecanisme que s'ha forçat des del control. D'aquesta manera si des de el PC de control s'ha decidit obrir airejadors/finestres, de forma mecànica es pot transmetre al sensor, utilitzant una palanca, i aquest tornar la lectura al PC per validar que s'ha fet l'acció demanada

  Amb petites modificacions es pot controlar també l'ús d'aire forçat (ventilador). Igual que abans fent servir un solenoide que s'activi en paral·lel, i que faci variar el sensor d'efecte Hall. O de forma mecànica amb una palanca, si la pressió de l'aire força l'obertura de la sortida de l'aire a l'exterior

- Amb l'adició d'un sensor d'humitat es podrà també monitoritzar el moment més adequat per obrir el rec o la vaporització d'aigua. Serviria per refrigerar, i també per augmentar la humitat relativa segons la situació climàtica

- El sensor de temperatura com és lògic ens servirà per acabar de conèixer les condicions ambientals, potser la més important, dins l'hivernacle, i prendre diverses decisions segons les lectures de la resta de sensors:
  - Obrir airejadors/finestres
  - Forçar la ventilació activant electroventiladors
  - Vaporitzar aigua per baixar la temperatura
  - Activar alarmes per alta o baixa temperatura

Un diagrama simplificat d'exemple del sistema, aplicat a un hivernacle, seria el següent:

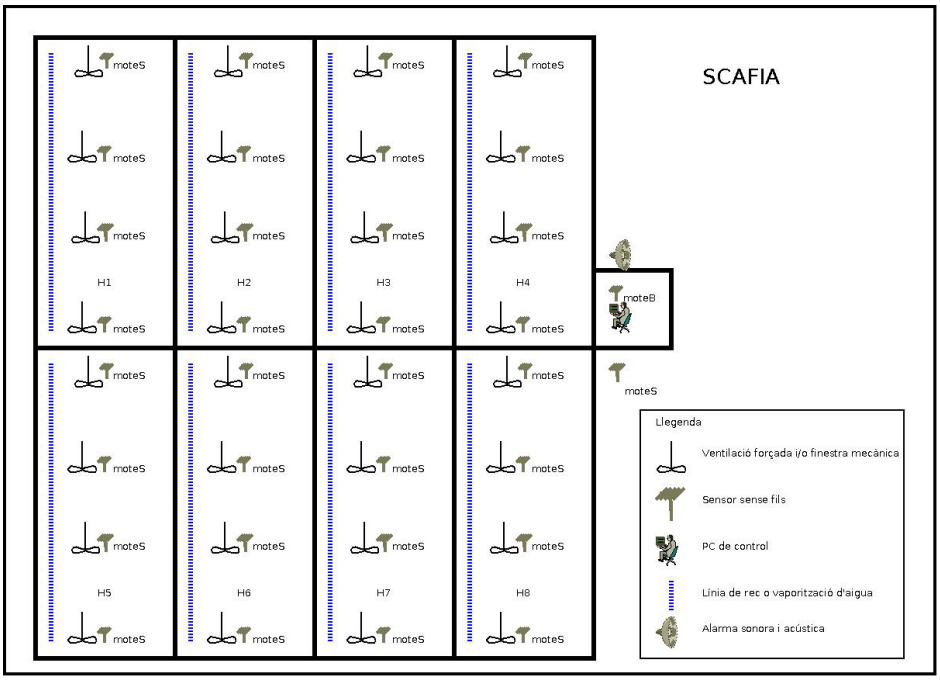


Figura 1.1. Exemple de sistema de control ambiental aplicat a un conjunt d'hivernacles

A l'esquema podem veure 8 àrees separades físicament (H1..H8), que correspondrien a hivernacles. A dins de cada hivernacle s'instal·larien tants sensors, anomenats moteS (*mS*), sense fils com fossin necessaris, per cobrir l'àrea que es cregui convenient i obtenir el detall desitjat. Fins i tot cada hivernacle podria contenir espècies amb requeriments ambientals diferents, respecte als altres hivernacles, sempre i que ocupin àrees tancades dins de l'hivernacle. Apart s'han representat els sistemes de refrigeració i rec que idealment serien controlats pel PC de control.

Existeix una petita àrea de treball, fora dels hivernacles, on instal·laríem el PC de control amb el programari de gestió de la xarxa de sensors sense fils, i idealment també controlaria tots els mecanismes ambientals. Al PC s'instal·laria un sensor que anomenarem moteB (*mB*) que faria d'enllaç entre el PC i la xarxa de sensors. A l'exterior d'aquesta àrea se situarien els elements d'alarma que avisarien de problemes ambientals greus a alguns dels hivernacles.

Fora dels hivernacles, i de l'àrea on situaríem l'element de control del sistema, s'ha situat un altre moteS (*mS*). La seva missió és obtenir dades ambientals de l'exterior dels hivernacles, recollides pels seus sensors, i afegir-les al dietari (log). L'objectiu és guardar dades per a estadístiques (per exemple per avaluar el rendiment tèrmic dels hivernacles en comparació amb les condicions exteriors).

1.3 Objectius

Com s'ha expressat a la introducció l'objectiu general del sistema és el control i registre de valors ambientals, i de les maniobres que es facin per mantenir-los. Aquest objectiu general és divideix en fites inferiors amb les tasques a desenvolupar, i que després les veurem reflectides en el cronograma configurant el pla de treball que ha de permetre assolir l'objectiu general.

Tenim el desplegament teòric de les següents tasques i subtasques a completar:

- **Xarxa de sensors**
    - Estudi tècnic dels nodes sensors
    - Estudi de la programació dels diferents tipus de nodes. Aquests es subdivideixen en:
        - Sensor d'enllaç. Estarà connectat al PC i farà de pont entre el sistema de control, i la resta de sensors desplegats a l'hivernacle. També portarà el control de sortida de potència per tal d'accionar els diferents sistemes que disposa l'hivernacle (ventilació, polvorització d'aigua, obertura de finestres i alarmes generals)
        - Sensor de mostra. Estarà situat a diferents llocs de l'hivernacle amb l'objectiu de monitoritzar les condicions ambientals. Per limitacions en les unitats disponibles només es disposa d'un node sensor, però la seva configuració és replicable per establir una veritable xarxa de sensors
        - Sensor de referència. Estarà situat a l'exterior de l'hivernacle per tenir referència dels valors ambientals a l'exterior. Aquests valors són necessaris a la tasca d'auditoria, però a causa que només tenim dos sensors, i un ha de fer de pont entre PC i l'altre sensor, serà aquest node-pont el que també farà aquesta funció de referència.
    - Programació de la xarxa de nodes, que haurà d'incloure:
        - Posada en marxa de la xarxa de sensors. Validació dels nodes actius
        - Lectura de valors dels sensors. Cada node ha de llegir els valors dels seus sensors de bateria, llum, temperatura, humitat i accionament. Aquesta lectura es farà de forma periòdica, marcada a la consola del sistema, excepte el sensor d'accionament que serà per interrupció quan detecti un accionament (canvi d'estat)
        - Transmissió dels valors dels sensors. Mentre la connexió entre nodes sigui possible es farà la transmissió dels valors llegits pels sensors, segons es van produint. En cas de caiguda de la xarxa s'haurà d'avisar com una alarma crítica del sistema, ja que no rebem dades del que està passant a l'entorn a controlar
        - Recepció de comandes d'accionament. Estarà limitat al node que estarà connectat al PC. En principi només s'accionaran 4 possibles mecanismes de control ambiental:
            - Ventiladors, i senyalització lluminosa. Per forçar l'entrada d'aire més fresc de l'exterior, davant de situació de temperatura límit
            - Polvorització d'aigua, i senyalització lluminosa. Per controlar la temperatura abans de la temperatura límit, o bé augmentar la humitat si aquesta és menor a la desitjada
            - Obertura de finestres, i senyalització lluminosa. Anirà accionada conjuntament amb els ventiladors per tal d'evacuar l'aire calent davant d'una situació límit, o bé de forma individual quan hi hagi un excés d'humitat
            - Senyalització acústica i lluminosa quan es rebassin els valors límits màxims (se suposa que les mesures de control han fallat). En aquest cas s'espera l'accionament manual per part d'un operador manual dels mecanismes que han de permetre retornar a la situació climàtica desitjada

- **Aplicació de la consola de control**
    - Estudi de la consola de control del sistema:
        - Fer consultes sobre les dades ja emmagatzemades
        - Permetre la configuració dels valors límits ambientals i d'altres
        - Permetre l'enviament de comandes al node de control d'accionaments o bé per comprovar una lectura d'un node sensor
    - Tractament automàtic de les dades rebudes pels sensors, i també de les maniobres de control ambiental. Els sensors enviaran les dades periòdicament, i també el control sobre la interfície de potència. Aquestes dades tindran un procés pel programari de control, que consistirà:
        - Gravació de les dades al dietari. Es definirà un fitxer de dietari de totes les sondes. Tindrà un format que permeti el seu tractament per altres programaris i que sigui intel·ligible per a un operador humà
        - Gravació de les dades de telemetria dels sensors respecte als valors de bateria, lluminositat, temperatura i humitat. El format d'aquest fitxer ha de permetre el seu tractament posterior, per exemple .CSV, per programari per càlcul estadístic
        - Mostra dels valors de les dades a la consola de control. Es tracta que la consola mostri per defecte l'estat del sistema en temps real, entenent aquest com la última dada rebuda. La mostra dels valors ha de sr intel·ligible per a un operador humà
    - Comunicació directa amb el node d'enllaç que controla els accionaments, per tal d'accionar, o desactivar, un d'ells de forma manual si és necessari (enviament de comandes)
    - Programar la reacció del sistema davant de les dades
        - Valor límit temperatura. Accionament de ventiladors, polvorització d'aigua, obertura de finestres o alarmes generals del sistema segons els valors considerats
        - Valor límit humitat. Davant de baixa humitat s'accionaria la polvorització d'aigua, si la temperatura ho permet. En cas d'alta humitat s'obririen les finestres superiors de l'hivernacle per evacuar l'excés
        - Valor límit de tensió d'alimentació. Avís a la consola de quin node està a prop de quedar sense funcionament per falta de bateria
        - Caiguda de la xarxa de nodes. Avís a la consola de control del sistema i activació de l'alarma crítica del sistema

- **Placa de control de potència (interfície)**. Es tracta de la presentació teòrica de com es podria connectar una placa de relès, o similar, per controlar potència, a un node. Per tant l'objectiu es presentar les adaptacions que s'haurien de fer, i el material necessari, per activar un dispositiu de potència a requeriment del sistema de control. El que si es presentarà en la realització del programari adequat del node corresponent

Aquest desplegament teòric fa que s'estableixin els següents objectius, que s'hauran d'anar assolint progressivament fins completar el total del projecte:

1. Node enllaç
    a. Pont entre aplicació de control i xarxa de nodes sensors
        i. Recepció de missatges des de la xarxa de sensors

   ii. Transmissió d'ordres des de l'aplicació de control cap a un node sensor concret
   b. Control de sistemes d'accionament mitjançant les seves sortides (veure punt 5)
      i. Sortida 1: obertura de finestres i la seva senyalització lluminosa
      ii. Sortida 2: ventiladors i la seva senyalització lluminosa
      iii. Sortida 3: polvorització d'aigua i la seva senyalització lluminosa
      iv. Sortida 4: alarma, i la seva senyalització lluminosa i acústica

2. Node sensor
   a. Lectura periòdica dels valors dels seus sensors
   b. Transmissió immediata dels valors dels seus sensors

3. Node sensor de referència (simulat sobre el node d'enllaç)
   a. Lectura periòdica dels valors dels seus sensors
   b. Transmissió immediata dels valors dels seus sensors

4. Aplicació de la consola de control
   a. Control de la xarxa de nodes
      i. Validació dels nodes actius actuals
      ii. Mostrar els nodes amb problemes d'alimentació elèctrica (baixa bateria)
      iii. Enviament de comandes de verificació cap a un node de la xarxa, o tots
      iv. Caiguda de la xarxa de nodes. Avís a la consola de control del sistema
   b. Configuració dels valors límits ambientals
   c. Control dels accionaments (veure punt 5)
      i. Mostrar l'estat actual dels accionaments
      ii. Enviament manual d'una ordre per accionar un o varis dels subsistemes de control ambiental (ventiladors, polvorització o obertura de finestres)
   d. Escriptura al fitxer de dietari de les dades ambientals rebudes pels nodes sensors
   e. Mostrar els valors ambientals de les dades dels nodes sensors a la consola de control
   f. Programar la reacció del sistema davant de les dades referents a:
      i. Valor límit temperatura
      ii. Valor límit humitat
      iii. Valor límit de tensió d'alimentació
   g. Escriptura d'un fitxer de dietari per recollir les decisions preses de forma automàtica, i també les manuals, pel sistema (accionaments) i també de les caigudes de la comunicació i la resta d'alarmes

5. Extensió de les característiques de les motes/nodes
   a. Implementar un sensor d'humitat
      i. Model recomanat
      ii. Programació de la interfície
      iii. Implementació en un node sensor
      iv. Verificació del funcionament en el sistema
   b. Placa de control de potència
      i. Disseny recomanat
      ii. Programació de la interfície
      iii. Implementació en el node d'accionaments

1.4 Enfocament i metodologia emprada en el desenvolupament

Seguint la definició clàssica de gestió d'un projecte informàtic, el podem fixar com el procés de direcció i control que ens porta a la seva definició, posada en marxa i avaluació, i tot dintre d'un control de recursos i terminis fixats per a la seva realització. En aquest cas la definició més clara és la del termini límit de lliurament, fixat per al final del present quadrimestre, i els recursos estan limitats al maquinari i programari disponible i les hores-treball que pugui, personalment, dedicar al seu desenvolupament. En aquest cas em toca fixar com a cap de projecte la coordinació necessària per portar-lo a la seva fi, si és possible amb èxit, i a més la seva realització pràctica. En aquest cas la metodologia seguida ha estat fixar els següents objectius generals:

- Definir unes funcionalitats determinades
- Respectar els terminis que s'han marcat al llarg del projecte per al lliurament de cada part i el tot
- Certificar el compliment de les fites i objectius fixats del projecte

No es parla en aquest cas de respectar un pressupost determinat, perquè no era necessari fer cap inversió addicional, ja que es disposava del material necessari, i les hores de treball teòricament facturables a un client no existeixen com a tal.

En qualsevol cas es tracta d'un projecte i com a tal existeixen uns riscos inherents al seu desenvolupament:

- Fer-ho excessivament gran i no assolir els terminis de lliurament amb tots els objectius realitzats
- Mal coneixement de la tecnologia emprada
- Qualitat esperada i especificacions estables

El risc directe més gran és el mal coneixement de la tecnologia emprada, en aquest cas els sistemes encastats i el seu sistema operatiu (TinyOS) i llenguatge de programació (nesC). Per aquest motiu s'ha necessitat un esforç addicional en conèixer la tecnologia i així avançar en el desenvolupament del projecte. Com es veurà al punt 8.2, del capítol 8, aquest desconeixement ha fet que certes parts del projecte puguin ser millorades si s'hagués disposat de més de temps.

El mètode que s'ha seguit en el desenvolupament del programari ha estat el cicle de vida en cascada pels següents motius:

- El coneixement de les tasques a realitzar és alt i permet fer una anàlisis funcional correcte
- El nivell de programació tot i ser de complexitat alta a nivell de motes/nodes, pel desconeixement inicial, no és molt gran
- S'ha de fer una fase de proves abans de la instal·lació per comprovar que es compleixen les funcionalitats demanades
- El nou sistema s'instal·laria una sola vegada
- Per assegurar la viabilitat del projecte aquest s'ha de fer en etapes tancades i amb una data de lliurament després d'una fase de proves

El cicle de vida en cascada s'adapta a aquestes demandes i característiques. Permet definir les etapes clàssiques com l'estudi d'oportunitat (que fer) i l'anàlisi (quines característiques són necessàries). Per facilitar el tractament del cicle de vida s'ha fet servir un diagrama de Gantt on s'especificaven les durades de les tasques, i les precedències corresponents, que ens havien d'assegurar l'èxit en la implementació del projecte.

Finalment la realització de la memòria exigeix un gran rigor documental, de forma que es pugui documentar les tasques realitzades a cada moment, els problemes trobats, la planificació i les desviacions d'aquesta així com les referències i informacions tècniques que s'han consultat. Per aquest motiu s'han disposat d'apunts, gairebé diaris, de l'avançament del projecte, i la memòria s'ha construït a partir d'aquests i la guia que disposem com a material didàctic.

1.5 Planificació del projecte

1.5.1 Planificació teòrica del projecte

El pla de treball, definit pels objectius del punt 1.3, del capítol 1, s'ha de reflectir en una ordenació lògica, de forma que l'assoliment d'una tasca permet la continuació del fil del projecte fins la seva finalització. Per aquest motiu s'estableixen una sèrie de condicionants, que ja estan expressats a la llista de tasques i objectius. A mode resumit i ordenat es tractaria de les següents precedències:

1. Coneixement tècnic dels nodes amb sensors
2. Establiment de la xarxa de sensors (comunicació)
3. Lectura de valors dels sensors instal·lats als nodes
4. Transmissió de les lectures anteriors
5. Recollida de dades en un node enllaç xarxa – PC de control
6. Tractament a la consola de control del sistema instal·lada al PC de control
    - Gravar dades ambientals
    - Iniciar si s'escau els procediments de modificació ambiental
    - Gravar les maniobres d'assegurament ambiental

Aquest ordre genera el següent pla general cronològic de treball, expressat amb un diagrama de Gantt:

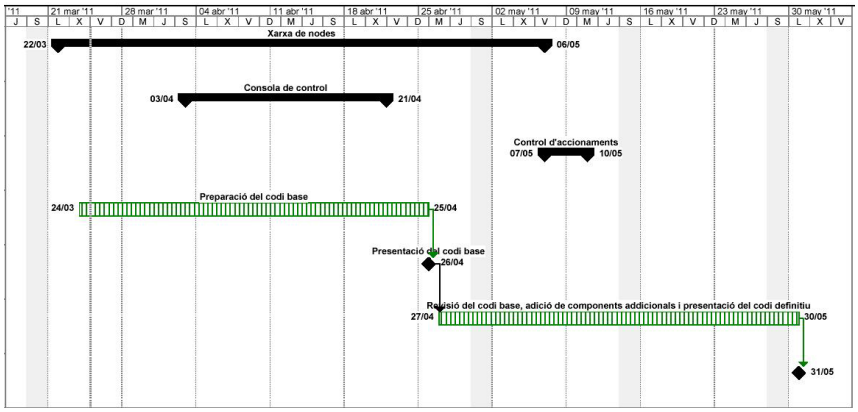


Figura 1.2. Pla general de treball projecte SCAFIA

Totes les tasques, sobretot les de programació de nodes i centre de control, lògicament se subdivideixen en altres tasques com són l'anàlisi, programació, verificació del funcionament i documentació, però per evitar un diagrama massa carregat s'ha decidit optar per indicar les dates inicials i finals de cada tasca.

Les feines principals tenen dos dates principals:

- El dia **26/04/2011**, que s'havia de presentar una primera maqueta funcional
- El dia **31/05/2011**, que s'havia de presentar el projecte definitiu

Aquestes dates marquen fites importants a complir al calendari. La primera ens permet fer una aproximació al projecte final, sense polir, on la majoria de funcionalitats han d'estar implementades, i la resta almenys en vies de solució. Per aquest motiu la primera fita marca una finalització del projecte, i tot i que a partir d'aquesta data, i fins la segona fita, no hi han especificacions, s'entén que es dedica a solucionar els problemes pendents de la primera part. També en la segona part del projecte s'ha deixat a l'estudi de la implementació d'un nou sensor (humitat) als nodes i el control de potència dels accionaments ambientals (interfície de potència).

A partir d'aquest punt es desplega cada part per separat, amb el seu diagrama de Gantt, per tenir una millor visió del pla de treball.

1.5.1.1 Xarxa de nodes

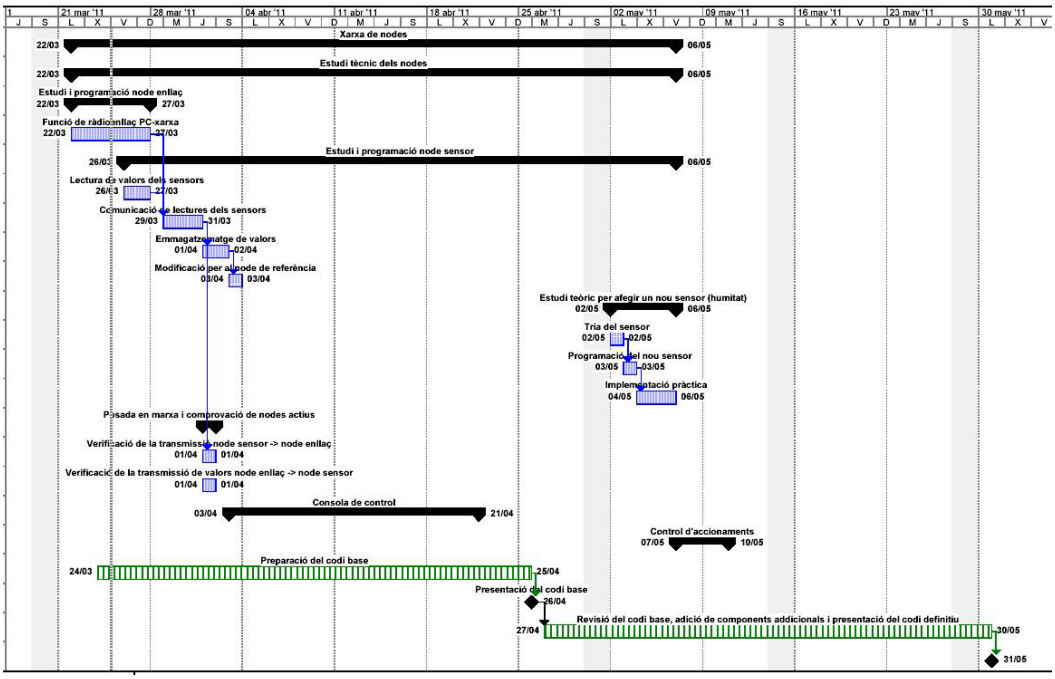


Figura 1.3. Diagrama de Gantt del desenvolupament previst de la xarxa de nodes

Tasca	Duració	Inici	Fi
Xarxa de nodes	**40 dies**	**22/03/2011**	**06/05/2011**
Estudi tècnic dels nodes	40 dies	22/03/2011	06/05/2011
Estudi i programació node enllaç	6 dies	22/03/2011	27/03/2011
Funció de radioenllaç PC-xarxa	6 dies	22/03/2011	27/03/2011
Estudi i programació node sensor	36 dies	26/03/2011	06/05/2011
Lectura de valors dels sensors	2 dies	26/03/2011	27/03/2011
Comunicació de lectures dels sensors	3 dies	29/03/2011	31/03/2011
Emmagatzematge de valors	2 dies	01/04/2011	02/04/2011
Modificació per al node de referència	1 dia	03/04/2011	03/04/2011
Estudi teòric per afegir un nou sensor (humitat)	5 dies	02/05/2011	06/05/2011
Tria del sensor	1 dia	02/05/2011	02/05/2011
Programació del nou sensor	1 dia	03/05/2011	03/05/2011
Implementació pràctica	3 dies	04/05/2011	06/05/2011

Posada en marxa i comprovació de nodes actius	1 dia	01/04/2011	01/04/2011
Verificació de la transmissió node sensor -> node enllaç	1 dia	01/04/2011	01/04/2011
Verificació de la transmissió de valors node enllaç -> node sensor	1 dia	01/04/2011	01/04/2011

1.5.1.2 Consola de control

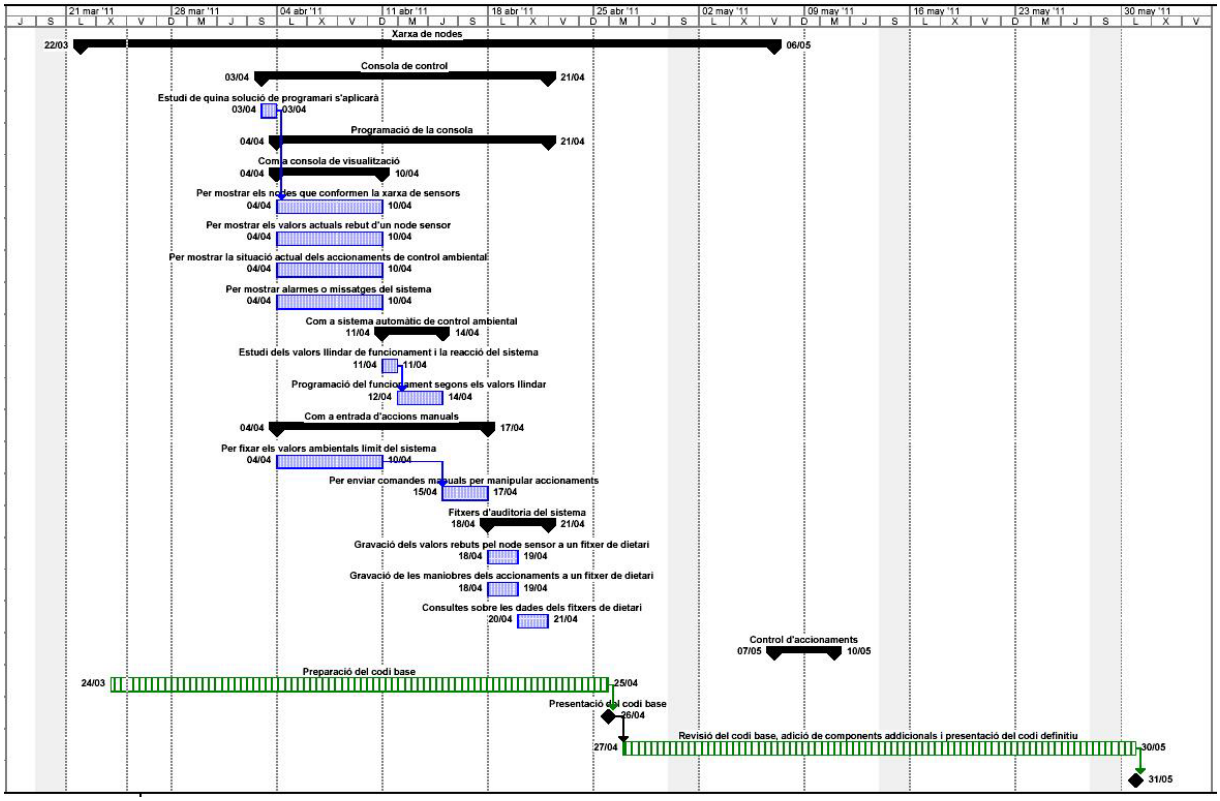


Figura 1.4. Diagrama de Gantt del desenvolupament previst de la consola de control

Tasca	Duració	Inici	Fi
Consola de control	**16 dies**	**03/04/2011**	**21/04/2011**
Estudi de quina solució de programari s'aplicarà	1 dia	03/04/2011	03/04/2011
Programació de la consola	15 dies	04/04/2011	21/04/2011
Com a consola de visualització	5 dies	04/04/2011	10/04/2011
Per mostrar els nodes que conformen la xarxa de sensors	5 dies	04/04/2011	10/04/2011
Per mostrar els valors actuals rebut d'un node sensor	5 dies	04/04/2011	10/04/2011
Per mostrar la situació actual dels accionaments de control ambiental	5 dies	04/04/2011	10/04/2011
Per mostrar alarmes o missatges del sistema	5 dies	04/04/2011	10/04/2011
Com a sistema automàtic de control ambiental	4 dies	11/04/2011	14/04/2011
Estudi dels valors llindar de funcionament i la reacció del sistema	1 dia	11/04/2011	11/04/2011
Programació del funcionament segons els valors llindar	3 dies	12/04/2011	14/04/2011
Com a entrada d'accions manuals	11 dies	04/04/2011	17/04/2011
Per fixar els valors ambientals límit del sistema	5 dies	04/04/2011	10/04/2011
Per enviar comandes manuals per manipular accionaments	2 dies	15/04/2011	17/04/2011
Fitxers d'auditoria del sistema	4 dies	18/04/2011	21/04/2011
Gravació dels valors rebuts pel node sensor a un fitxer de dietari	2 dies	18/04/2011	19/04/2011
Gravació de les maniobres dels accionaments a un fitxer de dietari	2 dies	18/04/2011	19/04/2011
Consultes sobre les dades dels fitxers de dietari	2 dies	20/04/2011	21/04/2011

1.5.1.3 Control d'accionaments

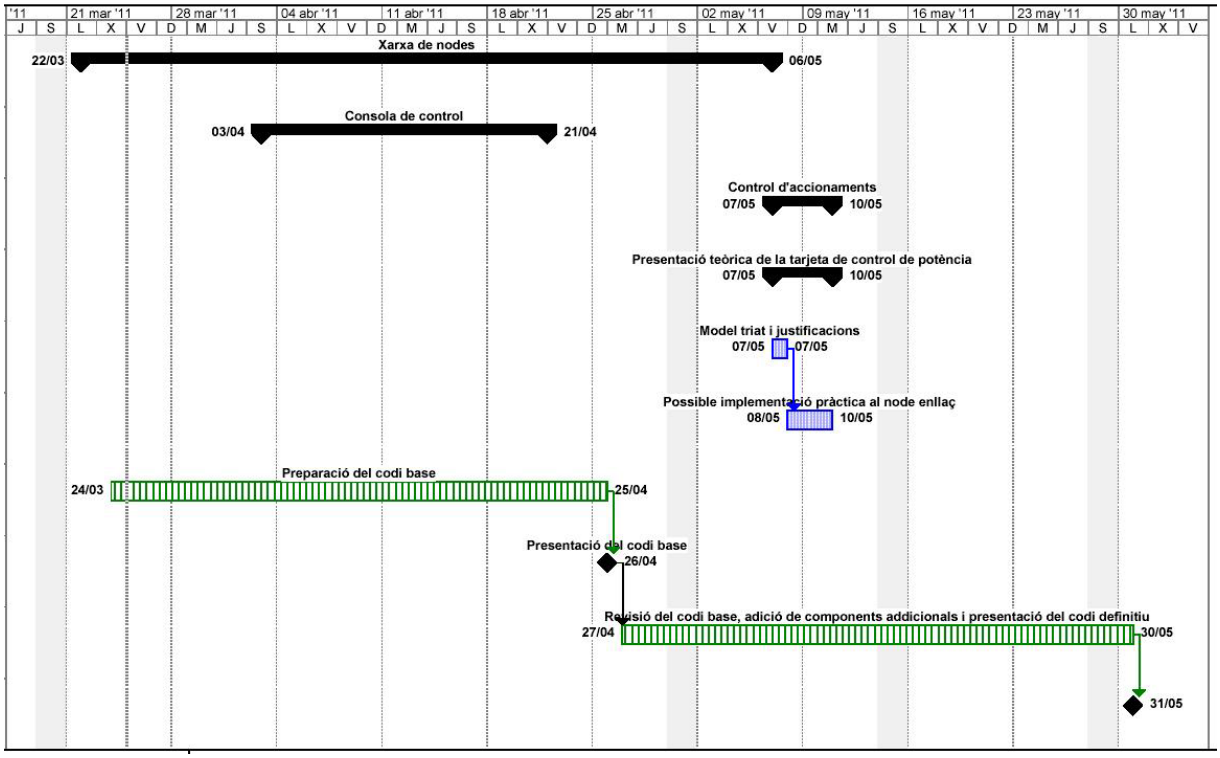


Figura 1.5. Diagrama de Gantt del desenvolupament previst del control d'accionaments

Tasca	Duració	Inici	Fi
Control d'accionaments	**4 dies**	**07/05/2011**	**10/05/2011**
Presentació teòrica de la targeta de control de potència	4 dies	07/05/2011	10/05/2011
Model triat i justificacions	1 dia	07/05/2011	07/05/2011
Possible implementació pràctica al node enllaç	3 dies	08/05/2011	10/05/2011

1.5.1.4 Llistat de totes les tasques del pla de treball

Tasca	Duració	Inici	Fi
Xarxa de nodes	**40 dies**	**22/03/2011**	**06/05/2011**
Estudi tècnic dels nodes	40 dies	22/03/2011	06/05/2011
Estudi i programació node enllaç	6 dies	22/03/2011	27/03/2011
Funció de radioenllaç PC-xarxa	6 dies	22/03/2011	27/03/2011
Estudi i programació node sensor	36 dies	26/03/2011	06/05/2011
Lectura de valors dels sensors	2 dies	26/03/2011	27/03/2011
Comunicació de lectures dels sensors	3 dies	29/03/2011	31/03/2011
Emmagatzematge de valors	2 dies	01/04/2011	02/04/2011
Modificació per al node de referència	1 dia	03/04/2011	03/04/2011
Estudi teòric per afegir un nou sensor (humitat)	5 dies	02/05/2011	06/05/2011
Tria del sensor	1 dia	02/05/2011	02/05/2011
Programació del nou sensor	1 dia	03/05/2011	03/05/2011
Implementació pràctica	3 dies	04/05/2011	06/05/2011
Posada en marxa i comprovació de nodes actius	1 dia	01/04/2011	01/04/2011
Verificació de la transmissió node sensor -> node enllaç	1 dia	01/04/2011	01/04/2011
Verificació de la transmissió de valors node enllaç -> node sensor	1 dia	01/04/2011	01/04/2011
Consola de control	**16 dies**	**03/04/2011**	**21/04/2011**
Estudi de quina solució de programari s'aplicarà	1 dia	03/04/2011	03/04/2011
Programació de la consola	15 dies	04/04/2011	21/04/2011
Com a consola de visualització	5 dies	04/04/2011	10/04/2011
Per mostrar els nodes que conformen la xarxa de sensors	5 dies	04/04/2011	10/04/2011
Per mostrar els valors actuals rebut d'un node sensor	5 dies	04/04/2011	10/04/2011
Per mostrar la situació actual dels accionaments de control ambiental	5 dies	04/04/2011	10/04/2011
Per mostrar alarmes o missatges del sistema	5 dies	04/04/2011	10/04/2011
Com a sistema automàtic de control ambiental	4 dies	11/04/2011	14/04/2011
Estudi dels valors llindar de funcionament i la reacció del sistema	1 dia	11/04/2011	11/04/2011
Programació del funcionament segons els valors llindar	3 dies	12/04/2011	14/04/2011
Com a entrada d'accions manuals	11 dies	04/04/2011	17/04/2011
Per fixar els valors ambientals límit del sistema	5 dies	04/04/2011	10/04/2011
Per enviar comandes manuals per manipular accionaments	2 dies	15/04/2011	17/04/2011
Fitxers d'auditoria del sistema	4 dies	18/04/2011	21/04/2011
Gravació dels valors rebuts pel node sensor a un fitxer de dietari	2 dies	18/04/2011	19/04/2011
Gravació de les maniobres dels accionaments a un fitxer de dietari	2 dies	18/04/2011	19/04/2011
Consultes sobre les dades dels fitxers de dietari	2 dies	20/04/2011	21/04/2011
Control d'accionaments	**4 dies**	**07/05/2011**	**10/05/2011**
Presentació teòrica de la targeta de control de potència	4 dies	07/05/2011	10/05/2011
Model triat i justificacions	1 dia	07/05/2011	07/05/2011
Possible implementació pràctica al node enllaç	3 dies	08/05/2011	10/05/2011
Preparació del codi base	**29 dies**	**24/03/2011**	**25/04/2011**
Presentació del codi base	1 dia	26/04/2011	26/04/2011
Revisió del codi base, adició de components addicionals i presentació del codi definitiu	**28 dies**	**27/04/2011**	**30/05/2011**
Presentació del codi definitiu del sistema	1 dia	31/05/2011	31/05/2011

1.5.2 Planificació real

La planificació anterior va ser calculada abans de començar el projecte, bàsicament per estructurar clarament la feina a fer, i que teòricament havia de portar-nos fins a un final amb èxit.

Al llarg del semestre ha estat evident que algunes de les parts presentaven més dificultats que altres, per exemple comprendre el nou entorn de treball (TinyOS i nesC), i això ha fet que les dates estimades en enllestir les tasques s'han modificat. Tot i que el resultat final a cada fita de lliurament, lògicament s'ha respectat. Les modificacions de la planificació són presentades en els següents apartats.

1.5.2.1 Xarxa de nodes

La planificació parlava de 14 dies en total, desenvolupar les motes/nodes, i s'ha trigat 18 dies en la part principal:

- **Estudi i programació del node enllaç**. Va començar el dia 22/03/2011 i vaig tenir força problemes en establir una comunicació bàsica, no va ser fins el dia 29/03/2011 que ho vaig aconseguir i va facilitar-me la resta de la programació dels nodes
- **Estudi i programació del node sensor**. La programació principal l'havia acabat el 10/04/2011 tot i que fins el dia 25/04/2011 he fet algun afegit. La modificació pel node de referència l'he fet el 24/04/2011
- **La posada en marxa i comprovació de nodes actius**. S'ha anat fent al llarg del desenvolupament tant del programari del node, com de la consola, tot i que bàsicament el dia 10/04/2011 ja estava també aconseguit

1.5.2.2 Consola de control

La planificació parlava de 17 dies i he trigat 10 dies en desenvolupar l'aplicació de consola, falta la part de consultes que s'ha decidit que queda fora de la consola.

- **La programació de la consola** ha començat sobre el dia 10/04/2011, i la majoria dels missatges de visualització estaven resolts sobre el 14/04/2011
- **El sistema automàtic de control ambiental**, quan s'ha d'accionar, i la part de fitxers d'auditoria s'ha finalitzat sobre el dia 20/04/2011, tot i que s'han anat depurant algunes errades i missatges fins el dia 24/04/2011
- La planificació parlava de 5 dies en total, per trobar el component adequat i integrar-ho a la mota, i he trigat els 5 dies:

1.5.2.3 Estudi teòric per afegir un nou sensor

El vaig realitzar en el temps estimat:

- **Tria del sensor**: vaig investigar un sensor de l'empresa Sensirion, però el vaig desestimar ja que era massa complex per integrar-ho (funcionava amb tràfic de paquets de dades, rellotges i I^2C). Finalment vaig trobar un humistor (Honeywell HIH-5031) que funciona d'una forma molt semblant a una resistència que varia proporcionalment al paràmetre a mesurar, com per exemple una LDR (sensible a la llum) o el termistor, que són els components integrats a la placa de la mota
- **Programació del nou sensor**. Vaig estudiar la corba de transferència de l'humistor i així treure les fórmules necessàries per afegir-les al programa, també vaig modificar el component SensorP.nc i SensorBP.nc per afegir el control sobre el nou sensor
- **Implementació pràctica**. Finalment vaig muntar els sensors, un a cada mote, fent servir un petit circuit muntat sobre una *protoboard* i amb una resistència de 100 kOhms. El humistor utilitza el convertidor analògic digital ADC3 de l'ATZB-24-A2

1.5.2.4 Control d'accionaments (manual i automàtic)

Les tasques referents a l'enviament de comandes manuals, i automàtiques segons alarmes, per implementar accionaments i la targeta de control de potència van lligades, i s'han fet pràcticament alhora, i també dintre del temps estimat:

- S'ha implementat un nou component que s'afegeix al node enllaç i es diu **ActionP.nc**. Aquest component rep les ordres des de el programari de control i simula sobre el LED 0 (vermell) que activa o desactiva un accionament. Com que són 4 els possibles accionaments, i només tenim un LED, no es pot mostrar de forma simultània com quedarien uns accionaments activats mentre desactivem altres, però la idea d'aquest nou component és que sigui la base i pugui evolucionar si disposem de la targeta adequada, i les sortides de l'ATZ-24-A2. Es veurà doncs que:
    - Les alarmes accionen i desactiven automàticament les sortides adequades (sempre de forma simulada sobre el LED 0)
    - L'encarregat humà de la instal·lació podrà activar o desactivar un accionament de forma manual des de una comanda en línia

1.5.3 Dietari

Les tasques anteriors, presentades de forma general, s'han desenvolupat segons el següent calendari, que documenta de forma quasi diària tot el treball fet:

17/03/2011
- Instal·lació del programari de desenvolupament:
    - Màquina virtual i sistema operatiu UBUNTU 10.04 amb entorn TinyOS 2.1.1 preinstal·lat
    - IDE Eclipse

18/03/2011
- Proves de compilació i càrrega del programa BlinkCOU a les motes

19/03/2011 a 21/03/2011
- Lectura dels mòduls i materials d'ajuda de la web TinyOS
- Definició del projecte: objectius, raons, actors, ...

22/03/2011 a 25/03/2011
- Preparació del pla de treball del projecte, revisió i lliurament

26/03/2011 a 29/03/2011
- Detecció d'un problema en la càrrega de programari als nodes. Revisió de tota la instal·lació del sistema i llibreries, fins trobar la solució al problema

30/03/2011
- Inici de l'estudi de la mota/node remota mS. Fixar requeriments necessaris pel projecte

31/03/2011
- Estudi del programari BlinkCOU i BaseStation. Que seran la base de part de les comunicacions i reaccions del sistema

10/04/2011
- Estudi de l'obtenció de la MAC del mote/node, reaccions a pulsació de botons i funcionament de l'ADC
- Fixar periodicitat de lectures de l'ADC
- Implementar component MoteIdP.nc

11/04/2011
- Començament de la implementació del component SensorsP.nc

12/04/2011
- Implementació i millora dels components HallP.nc, MoteIdP.nc i SensorsP.nc
- La mota/node no reacciona als canvis de temps o interrogacions des del sistema. Existeix un problema amb la interfície Receive

13/04/2011
- Solucionat el problema al component SensorsP.nc i la interfície Receive

- Es continua la implementació del programari a la mota mS, finalitzant el component HelloMsgP.nc

14/04/2011

- S'utilitza l'aplicació mig per generar les classes Java que permeten accedir als valors enviats des de HallMsgP.nc, HelloMsgP.nc i SensorsP.nc
- Es programa al sistema i a SensorsP.nc la configuració de canvi de temps a la mota i la recepció del missatge de posada en marxa
- Es finalitza la programació de les reaccions de la consola de sistema a: recepció de missatge del sensor Hall, valors dels sensors, pulsació del botó d'usuari, canvi de temps de lectura de sensors i missatge de posada en marxa

15/04/2011

- Definir un disseny que permeti controlar si la xarxa de nodes està activa
- Definir el sistema de dietari (logs) i sistema estadístic
- Estudi del canvi de comunicació ràdio a sèrie per implementar en sensor de referència sobre la mota/node d'enllaç (mB)

17/04/2011

- Definir i començar la programació de la reacció del sistema als valors climàtics rebuts (alarmes i accionaments)

18/04/2011

- Programar els fitxers de sortida d'auditoria (log) i estadístic
- Programar la recepció de comandes de consola al sistema i el seu tractament
- Definir i programar un fitxer .ini que conté els valors de l'entorn

19/04/2011

- Reprogramar SensorsP.nc a SensorsBP.nc que serà el component amb comunicació sèrie que s'instal·larà al node d'enllaç simulant ser un node de referència climàtica
- Fer una funció d'impressió en pantalla, i també al fitxer d'auditoria, de tots els missatges que es van produint al sistema
- Canvi en la programació dels nodes/motes per que l'inici sempre sigui correcte i no arribin missatges no desitjats
- Modificació final de BaseStation.nc per convertir-ho en la mota/node d'enllaç (mB)

20/04/2011

- Reestudi de característiques de la consola de sistema de Java. Es fan canvis per detectar la caiguda de la xarxa de nodes, s'inclou també la ruta del *socket* que a partir d'ara és configurable i la reprogramació forçada del temps de lectura de sensors d'un node que el té incorrecte

22/04/2011 a 25/04/2011

- Comprovació de reaccions del sistema davant alarmes per valors climàtics
- Programació final de la detecció de caiguda de la xarxa
- Estudi d'un possible sensor d'humitat
- Realització de la documentació de la PAC 2
- Lliurar la feina desenvolupada el 25/04/2011, que inclou les versions estables mB, mS i consola de sistema en Java

26/04/2011

- Revisió del tractament de la línia de comandes de la consola ja que s'ha detectat algun problema
- Canvi en els valors enters 'llargs' que s'utilitzen com a constants en nesC afegint 'L' al valor enter (xxxL)

27/04/2011

- Estudi de la integració de l'aplicació SerialForwarder en la consola. Finalment es desestima per manca de temps

28/04/2011
- Revisió de tot codi de la consola de sistema, compactant tot el que es pugui, bàsicament missatges de sortida, i passant funcions auxiliars a una classe secundària

29/04/2011
- Estudi i cerca de sensors d'humitat. Candidats amb estudi exhaustiu: Sensirion SH71 i Honeywell HIH-5031
- Estudi de l'actuació sobre sortides del microcontrolador ATMEL ATZ

30/04/2011
- Disseny d'un possible sistema de control de potència aprofitant MOSFET i la baixa senyal de sortida de l'ATZ
- Tria de l'humistor HIH-5031 com a solució al sensor d'humitat a implementar

03/05/2011
- Compra dels components necessaris per implementar el nou sensor a les motes/nodes
- Estudi de l'estructura de la memòria del projecte

06/05/2011 a 11/05/2011
- Revisió de les correccions suggerides pel consultor al lliurament de la PAC2, que inclouen la fixació en temps de compilació de TOS_NODE_ID, la gestió de connexió sèrie (tractar errors EBUSY), el millorar la comunicació ràdio amb *PacketAcknowledgements*, conversió valor ADC a un valor correcte de tensió de bateria i lluminositat, i cerca de documentació per tractar el *warning* detectat al compilar els components de la mote mB

12/05/2011 a 14/05/2011
- Programar el component ActionP.nc que s'afegirà a la mota/node mB, i que simularà una comunicació bidireccional per al control d'accionaments
- Prova del nou component ActionP.nc tant en mode manual, missatges via línia de comandes, com automàtic als valors climàtics
- Muntar els 2 sensors d'humitat, un a cada mote, programar la conversió d'ADC a humitat relativa a la consola del sistema, i provar el seu funcionament

19/05/2011 a 23/05/2011
- Prova final del sistema. Consola de línia de comandes, reaccions manuals i automàtiques sobre la mota base (mB), recepció correcta de valors dels sensors, verificació dels missatges de consola i generació del fitxer d'auditories i estadístiques

24/05/2011 a 31/05/2011
- Preparació de la documentació per al lliurament de la PAC3

20/05/2011 a 10/06/2011
- Preparació de la documentació per al lliurament de la memòria del projecte

1.6 Recursos materials

Per desenvolupar aquest projecte s'ha disposat dels següents recursos de maquinari i programari:

- Un PC Intel (R) core (TM)2 Quad CPU Q6600@2.4 GHz, 2 GB RAM.
  Programari instal·lat:
  - Sistema operatiu Microsoft Windows XP Professional versió 2002, service pack 3, amb les següents aplicacions instal·lades
    - Paquet ofimàtic Microsoft Office 2000
    - Microsoft Project 2003 (realització de diagrames de Gantt)
    - diaw.exe 0.97.1 (programa per generar diagrames)

  - TinyCAD v. 2.60.01 (esquemes electrònics)
  - Màquina virtual VMWare© Player, versión 3.0.1
    Programari instal·lat en aquesta màquina virtual:
    - Sistema operatiu Ubuntu versió 10.04
      Programari instal·lat al sistema operatiu UBUNTU:
      - IDE Eclipse amb YETI *plug-in*
      - TinyOS 2.1.1
        - SerialForwarder
      - meshprog (bootloader TinyOS)
      - JAVA versió 1.6

- 2 unitats de sensor inhalàmbric mote COU_1_2, que tenen encastats
  - Sensor de llum
  - Sensor efecte Hall
  - Sensor de tensió d'alimentació
  - Sensor de temperatura
  - Modificada per afegir un sensor d'humitat capacitiu HIH-5031 (Honeywell)

1.7 Productes obtinguts

Tot el desenvolupament anterior ha revertit en la creació dels següents productes:

- Les motes/nodes COU_1_2 s'han modificat amb l'adició d'un nou sensor, en aquest cas un humistor Honeywell HIH-5031, que permet la lectura en continu del valor de la humitat relativa ambiental

- S'han dissenyat 2 perfils de mote/node del sistema
  - mota/node remot (**mS**) : per instal·lar al lloc a controlar. La seva funció principal és la d'enviar els valors climàtics i operatius, que recullen periòdicament els seus sensors
  - mota/node base (**mB**) : per instal·lar en un port USB del PC on s'executa la consola del sistema. La seva funció principal és la de fer pont entre el sistema i la xarxa de nodes/motes remotes. A més també inclou la lectura de sensors, igual que mS, per implementar així un node/mote de referència climàtica de la instal·lació. Finalment simularà l'accionament de mecanismes climàtiques amb les sortides del microcontrolador (en aquest cas l'activació/desactivació del seu LED 0)

- **ScafiaCon**: aplicació que farà les tasques d'auditoria i control climàtic del sistema (manual o automàtic). S'instal·larà al PC on es connecta el node/mota mB

Els capítols 3 i 4 d'aquesta memòria es centraran en la descripció exhaustiva de cadascun d'aquests components.

1.8 Descripció de l'estructura de desenvolupament del projecte

A continuació es desenvoluparan nous capítols amb l'objectiu d'aprofundir en la descripció del projecte:

- **Capítol 2**. L'estat actual tecnològic que ha permès desenvolupar l'estructura funcional del projecte. Estudi de les parts que conformen la solució dissenyada , i també la presentació de solucions reals o casos amb èxit que existeixen en aquest moment
- **Capítol 3**. Descripció funcional del disseny del projecte, explicacions del per què del seu funcionament, i la interacció entre els diferents components

- **Capítol 4**. Explicació tècnica del capítol 3 de forma detallada (diagrama de casos d'ús, flux, ...)
- **Capítol 5**. Modificacions fetes al maquinari del que disposem, per poder complir amb les condicions pactades del projecte
- **Capítol 6**. Viabilitat tècnica, anàlisi crític del sistema desenvolupat i de la validesa com a solució
- **Capítol 7**. Estudi econòmic de la realització del projecte, cost de producció i implantació
- **Capítol 8**. Conclusions sobre el grau de compliment dels objectius inicials del projecte. Presentació de millores i grau de coneixement aconseguit sobre les tecnologies emprades
- **Capítol 9.** Glossari amb els termes que apareixen al projecte
- **Capítol 10.** Bibliografia i recursos utilitzats
- **Annexos**. Contenen tota la informació associada d'utilitat del projecte per a la seva consulta. S'inclouen llistats de codi de programa, manuals d'usuari i execució, i exemples de fitxers de sortida

2. Antecedents

La tecnologia emprada en el desenvolupament del projecte ha tingut una evolució tecnològica fins el dia d'avui, que ha possibilitat la seva aplicació per resoldre els problemes o objectius plantejats. Els següents continguts intenten mostrar com a ajudat aquest camí recorregut a conformar una solució vàlida als objectius plantejats.

2.1 Estat de l'art

El sistema base del projecte és una xarxa de sensors inhalàmbrics (WSN - *Wireless Sensor Network*), amb la comesa principal d'obtenir els valors climàtics de l'espai d'instal·lació i transmetre'ls a una aplicació pel seu registre. Les característiques principals dels dispositius que formen aquesta xarxa són les següents:

- Estan desplegats espacialment en l'espai a controlar
- Són autònoms un dels altres
- Es comuniquen entre si via ràdio (ZigBee, especificació 802.15.4)
- A nivell de maquinari estan formats almenys per un microcontrolador, un o varis sensors, un radiotransceptor, un port sèrie i una de font d'alimentació pròpia que li dona autonomia
- El seu funcionament està pensat per reduir al màxim el seu consum energètic
- Baix cost econòmic

A més la xarxa WSN s'autoorganitza, la seva topologia no és fixa i preprogramada, permetent també el multicamí per encaminar els paquets de dades en cas de caigudes de nodes adjacents. L'objectiu principal és la seva persistència i simplicitat per permetre la comunicació. El número de components que pot formar una xarxa WSN pot ser de milers, i per tant l'àrea de desplegament, o superfície d'instal·lació, pot ser molt extensa. A més l'alta concentració de nodes també pot fer que el cost energètic de traspàs d'informació entre ells sigui reduït.
Les topologies comuns d'una xarxa WSN són:

- **Estrella** (*star*): on cada node està connectat directament al *gateway*, és la més simple i els nodes han d'arribar via ràdio al node base o *gateway*, així que les xarxes no són molt extenses (el seu límit és l'abast ràdio de l'enllaç ràdio del node remot al node base)
- En **arbre** (*cluster tree*): existeix una jerarquia de nodes, a nivell de capes o nivells, que enruta els paquets fins al *gateway*. Permet fer xarxes molt extenses, ja que la distància entre nodes la podem col·locar al límit de l'enllaç ràdio entre els nodes remots de cada nivell de l'arbre

- En **malla** (*mesh*): els nodes estan interconnectats oferiment diversos camins als paquets, millors quan més densa és la xarxa a nivell de nodes, per arribar fins el *gateway* o node base. Aquesta característica reforça la seguretat de funcionament de la xarxa, ja que obtenim camins redundants

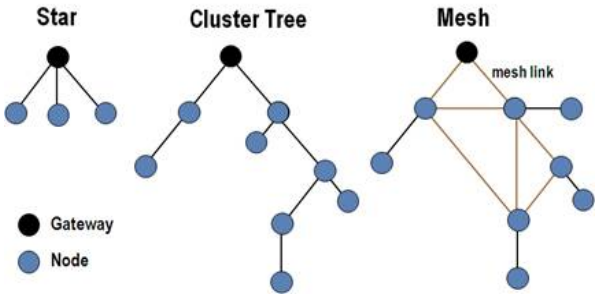


Figura 2.1. Topologies de xarxa WSN

Les aplicacions són moltes des del control domòtic, inmòtic, seguretat, control industrial, ... Qualsevol que necessiti la mesura de valors i la seva transmissió de forma eficient, i que serà especialment afavorida per les característiques descrites dels nodes o components. El desplegament d'una xarxa d'aquest tipus és molt ràpida, ja que no està lligada en principi a cap preinstal·lació de cablejat elèctric o de comunicació, permetent la modificació d'instal·lació dels seus components pràcticament en qualsevol moment. Podem dir que les paraules claus d'una WSN són la rapidesa i flexibilitat de desplegament.

Com a conclusió podem considerar que els objectius principals d'una xarxa WSN són el de mantenir una xarxa de sensors amb un cost econòmic i consum energètic mínim, sense una pèrdua significativa d'eficiència, flexibilitat i rapidesa de desplegament i amb un disseny modular dels seus components que faciliti el manteniment.

El component principal de la xarxa és el node, sigui en versió recol·lector de dades, *gateway*, o tots dos. Com ja hem explicitat està format per una sèrie de components tant de maquinari com programari, i un protocol de comunicació:

- Mota COU24 (versió COU_1_2)
  - Microcontrolador: **ATZB-24-A2** d' **ATMEL**, que integra el microcontrolador **Atmega1281** i els transceptor **AT86RF230** per permetre el funcionament del node i la comunicació inhalàmbrica
  - Sensors integrats al node:
    - Temperatura: **MCP9700** de **Microchip**
    - Lluminositat: **PDV-P9003-1** de **Advanced Photonix**
    - Hall: **BU52001GUL** de **Rohm Semiconductor**
    - Humitat relativa: **HIH-5031** de **Honeywell**
    - Sensor de tensió d'alimentació (pont divisor de tensió)
  - Alimentació elèctrica: 2 piles AA, o directament pel port USB (3v)
  - Comunicació sèrie (USART) pel port USB
- Programari TinyOS
- Protocol de comunicació inhalàmbrica ZigBee 2,4GHz, IEEE 802.15.4

2.1.1 Mota COU24 (versió COU_1_2)

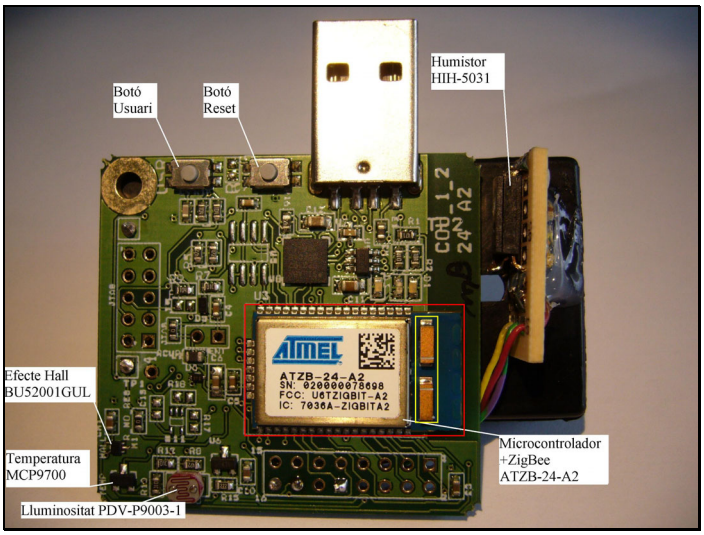


Figura 2.2. Mota o node COU_1_2 modificada amb humistor HIH-5031

El component principal és el microcontrolador, ATZB-24-A2 que integra un microcontrolador ATMEL ATmega1281 i un mòdul ràdio AT86RF230 que utilitza el protocol ZigBee 2,4GHz IEEE 802.15.4 (tots dos integrats dins d'un aïllament metàl·lic, blindatge, per evitar distorsions per efecte de la radiofreqüència), a més de les antenes (a la fotografia encerclat pel rectangle vermell):

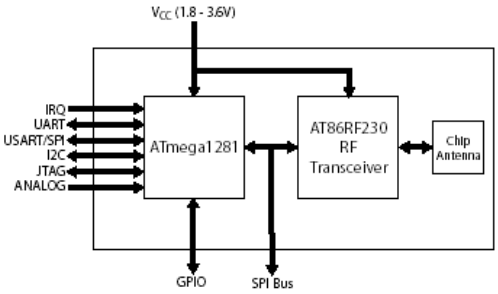


Figura 2.3. Diagrama de blocs de l'ATZB-24-A2

El microcontrolador ATmega1281 s'encarrega de la gestió d'entrades-sortides i de la lògica programada del node (el seu funcionament). El bus SPI connecta el microcontrolador amb el transceptor ràdio AT86RF230, que gràcies al mòdul d'antenes integrat (rectangle groc a la fotografia del node) permet l'emissió i recepció de missatges via ràdio, i per tant la comunicació inhalàmbrica del node. La gràcia del disseny és que en una única placa s'integra tot el maquinari de control i transmissió. El detall dels pins de connexió d'aquest mòdul és el següent:

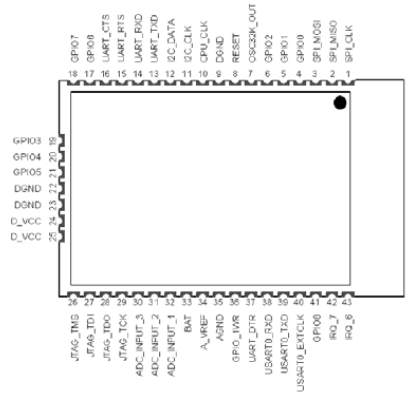


Figura 2.4. Connexions del mòdul ATZB-24-A2

La resta de l'electrònica del node serveix per adaptar els sensors a l'entrades ADC, botons, alimentar de forma estable el mòdul ATZB-24-A2 (3v), i facilitar la connexió sèrie per USB entre el node i un PC.

El ATmega1281 és un microcontrolador de 8 bits amb 128 kB de memòria flash, 4kB de memòria EEPROM, 8 kB de memòria RAM, 2 ports USART serial i 8 canals ADC. El seu diagrama de blocs és el següent:

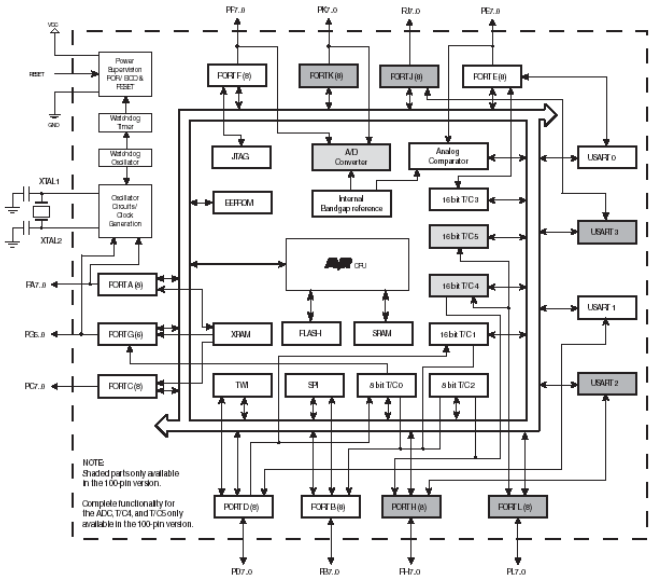


Figura 2.5. Diagrama de blocs ATmega1281

El **AT86RF230** és un transceptor ràdio de baixa potència dissenyat pel protocol ZigBee 2,4 GHz IEEE 802.15.4. Les característiques de consum energètic en transmissió/recepció són:

- *SLEEP* (inactiu): 20 nA
- Recepció (RX): 15,5 mA
- Transmissió (TX): 16,5 mA (a la màxima potència de transmissió 3 dBm)

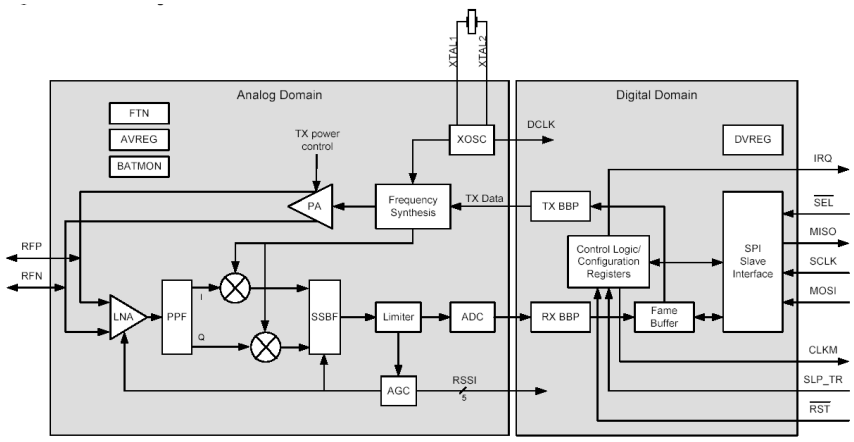


Figura 2.6. Diagrama de blocs de l'AT86RF230

A l'esquerra es veuen els pins RFP i RFN són els encarregats de l'emissió i recepció ràdio. A la dreta el bloc SPI (*slave interface*) que es connectaria a l'**ATmega1281** per rebre el missatge a transmetre. Els missatges tant de transmissió com de recepció s'emmagatzemen en un *buffer* intern de 128 kB.
Una simplificació de la part de transmissió/recepció és la següent:

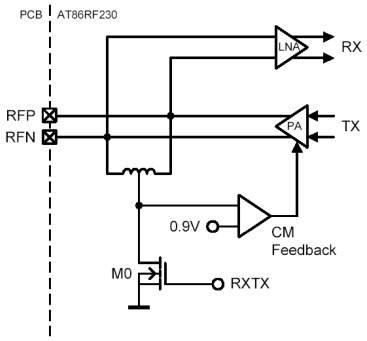


Figura 2.7. Recepció/transmissió AT86RF230

Es mostra ara un exemple de com estarien cablejats el microcontrolador ATmega1281 i el transceptor AT86RF230 utilitzant la comunicació mitjançant el bus SPI, al que no tenim accés directament per estar integrat a la placa ATZB-24-A2:

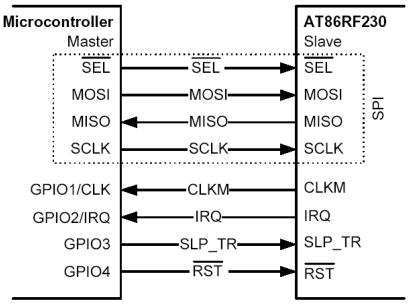


Figura 2.8. Connexions ATmega1281-AT86RF230

Finalment quan el node no està connectat al port USB, i per tant no es fa la comunicació via serial (USART), l'alimentació elèctrica la subministren 2 piles AA en sèrie. De forma que l'alimentació total és de 3v. En el cas de connexió al port USB, com que aquest té una tensió d'alimentació possible de 5v és necessari utilitzar un convertidor de tensió elèctrica, i passar de 5v a 3v. La comunicació sèrie s'aconsegueix amb el xip CP2102 que fa la conversió USART-USB.

2.1.2 Sistema operatiu TinyOS

El sistema operatiu que facilita la programació del node o mota és TinyOS, en aquest cas la versió 2.1.1. Aquest sistema operatiu va ser desenvolupat inicialment per la universitat de California Berkeley sota llicència BSD, tot i que ara és un projecte obert a la comunitat de desenvolupadors. El seu objectiu principal és minimitzar el cost en memòria, a causa de les limitacions de maquinari a les motes, sense que això impliqui un baix rendiment del sistema, i també el seu consum energètic. És per tant un sistema operatiu dissenyat específicament per plataformes que finalment, pels seus requeriments tècnics, poden ser integrades fàcilment en xarxes WSN. La seva arquitectura està basada en el disseny de components, utilitzant-se el llenguatge *nesC* per a la seva programació i té llibreries amb protocols de xarxa, serveis distribuïts, drivers de sensors i eines d'adquisició de dades. Aquest sistema és molt portable i existeixen moltes plataformes que el suporten.
El que subministra a alt nivell:

- Model de components reutilitzable
- Model d'execució concurrent (execució sincrònica i asincrònica per interrupcions)
- API, serveis i components per escriure aplicacions ràpidament

2.1.3 nesC

El llenguatge de programació per desenvolupar components que s'executin a TinyOS és nesC. Aquest llenguatge és molt similar a C o C++, però és orientat a components. Les diferències principals amb ells, apart d'algunes diferències en paraules reservades, són:

- Defineix components enlloc de classes (C++) o fitxers de programa (C)
- Els fitxers de capçalera (C) o les declaracions de classe (C++), ara són declaracions de components
- Les classes abstractes (C++) són interfícies (força similar)
- Els punters i similars són substituïts per un concepte nou, el *wiring* o cablejat entre components

L'aplicació no és totalment seqüencial, sinó que està basada en diferents abstraccions:

- *event* que forçaran l'execució de la part de codi relacionada a aquests quan es produeixen
- *command* execució de components amb l'ús de crida *call*
- *task* que és una crida que s'executa quan no hi una execució d'un *event* en marxa . La seva execució, amb crida *post*, es per ordre d'arribada

2.1.4 Protocol de comunicació ZigBee, IEEE 802.15.4

Per comunicar via ràdio les motes o nodes utilitzen el protocol ZigBee a 2,4 GHz, utilitzant la banda ISM (*Industrial, Scientific and Medical*) que no necessita llicència per transmetre. És un protocol específicament dissenyat pel seu ús en comunicacions de sistemes on sigui necessari un baix cost energètic i capacitats autoorganitzatives de la xarxa, tot i que la taxa de transmissió de dades ha de ser baixa. Per aquests motius aquest protocol és ideal per al seu ús en una xarxa WSN, on necessitem una alta durabilitat de bateries, i els valors de dades dels sensors no necessiten d'una gran taxa de transmissió.

2.2 Estudi de mercat

Les possibilitats comercials d'un sistema de control ambiental dedicat a la producció, i basat en xarxes de sensors inhalàmbrics, sembla no necessitar de grans arguments de venda. Pràcticament totes les seves característiques fan que sigui una opció molt interessant i cada cop més present en processos industrials, de domòtica i d'inmòtica. Repetim com a recordatori algunes d'elles, ja explicades en altres punts d'aquesta memòria:

- Sistemes flexibles d'instal·lació i escalables
- Alta resistència a fallades de la xarxa WSN
- Taxa de transmissió de dades suficient per a la majoria de processos
- Moltes llibreries *open source* i desenvolupament de baix cost, o sense cost, de llicències
- Maquinari (motes) de baix cost i que es pot dissenyar a mida la captura dels paràmetres són d'interès
- Baix consum energètic
- Possibilitat de connexió a sistemes TIC per permetre control i monitorització remota
- ...

Com veiem les bondats d'aquests sistemes són molt grans. Així que no és difícil trobar múltiples fabricants de dispositius, desenvolupadors de projectes i també aplicacions totalment funcionals al mercat. De cara a trobar solucions o casos d'èxit reals, és molt més fàcil en l'entorn de la climatització d'edificis o domicilis, i sobretot d'oficines comercials, que en l'àmbit d'aquest projecte que gira sobre l'aplicació de solucions en el sector agrari, concretament en el control d'un hivernacle. Per aquest motiu la cerca s'ha centrat en trobar un cas d´aplicació similar. L'**Instituto Tecnológico de Galicia (ITG)** ha desenvolupat el projecte **SWAP** (*Sistema de telemetría por redes WSN para Agricultura de Precisión*) amb l'objecte d'incrementar la productivitat i fer un ús extensiu del control precís a les produccions agrícoles. Per aquest motiu han editat la següent presentació de les diferents solucions actives en aquest moment:

> **Casos d'èxit amb xarxes WSN aplicats a centres de producció agrícola a Galícia**
> *http://www.itg.es/ITG/CasosExito_AgriculturaPrecision%20_RedesWSN.pdf*
>
> **Projecte SWAP**
> *http://www.swapproject.org/*

I dels quals en formen part:

- *Bodega **Condes de Albarei***. Situada a Cambados produeix 2 milions d'ampolles de vi amb denominació d'origen *Rias Baixas*.Vídeo d'aplicació de xarxes WSN pel control en temps real de vinya (està en gallec)
  http://www.youtube.com/watch?v=6JsUHWwpWks
- *Bodega **Guimaro*** ubicada en Sober, Lugo, amb denominació d'origen *Ribeira Sacra*
- *Cooperativa **HORSAL***. La cooperativa hortícola més gran de Galícia, situada en la comarca de Salnés
- *Viñedos **Camalie***, on s'ha millorat la irrigació augmentant la quantitat i qualitat de la uva, gràcies a la monitorització constant dels valors d'humitat i temperatura del terra, pressió d'aigua i llum
- *Projecte **LOFAR*** dedicat al control de malalties provocades pel fong Phytophthora a les plantacions de patata. Es va veure afavorit pel control constant dels valors mediambientals que afavoreixen la proliferació: temperatura, humitat ambiental i del terra, i il·luminació

A més també es mostra una informació apareguda al diari ***El Progreso*** sobre la implantació des de l'Institut Tecnològic de Galicia (ITG) de xarxes WSN a la producció agrícola:

http://elprogreso.galiciae.com/nova/78564.html
El Progreso
Galiciae
Los viticultores conocerán en tiempo real las necesidades de sus viñedos

---

Etiquetas: *XX.AA., viñedos, necesidades, viticultores*
X.L.Q. / El Progreso (Chantada).
El Instituto Tecnológico de Galicia (ITG), en colaboración con Xóvenes Agricultores (XX.AA.), implantan en la Ribeira Sacra la denominada viticultura de precisión, que se aplica a partir de sensores inalámbricos ubicados en zonas estratégicas. El proyecto, que nació hace más de un año, se dará a conocer mañana, a partir de las 20.15 horas, en el telecentro de Chantada.
Los responsables del ITG y de la agrupación XX.AA. destacaron que gracias a la implantación del sistema de recogida de datos a través de sensores, conocidos como ***'wireless sensor network'****, se* ***podrán predecir plagas y agentes atmosféricos*** *que puedan afectar a los viñedos en tiempo real gracias a la existencia de pequeñas estaciones ubicadas en distintas partes de la ribera.*
El responsable de Xóvenes Agricultores en la Ribeira Sacra, el chantadino Antonio Paz, explicó que gracias al proyecto, «cada xestor ou viticultor poderá acceder en tempo real e por internet á información sobre parámetros críticos do cultivo, tales como humedade, luz, radiación solar, temperatura do chan, ambiental e contido de auga do terreo, entre outras particularidades».
De esta forma, con cualquier soporte que pueda conectarse a la red, tal como un ordenador, ya sea de sobremesa o portátil, así como a través de una PDA o de un teléfono móvil de última generación, se podrá conocer el estado preciso de los viñedos que estén bajo control. Así, «poderán tomarse decisións no intre preciso ao tempo que redúcense os custos de explotación e mellora da eficiencia do cultivo. Todo isto cun custo de implementación moi reducido en comparación coas solucións de cable tradicionais ou outras tecnoloxías sen fíos como o GPRS», remarcó Antonio Paz.
innovación
El programa, conocido ***por Swap, esto es, Sistema de telemetría por redes WSN para Agricultura de Precisión,*** *fue planteado en sus orígenes ante la necesidad del sector agrícola gallego de innovar en los procesos productivos e incrementar las prácticas agrícolas basadas en métodos precisos con la incorporación de tecnología funcional. Por ello se eligieron los sensores inalámbricos.*
Para desarrollar la iniciativa, Xóvenes Agricultores comenzó hace más de un año a recopilar datos en puntos estratégicos de la Ribeira Sacra e introducirlos en una base de datos, mientras desde el Instituto Tecnológico de Galicia y desde la empresa Wireless Galicia, se inició el desarrollo técnico gracias al respaldo económico de la Secretaría Xeral de Modernización e Innovación Tecnológica de la Xunta a través de la línea Empresa Dixital 2010.
medio ambiente
Antonio Paz estimó que la llegada de las nuevas tecnologías a la Ribeira Sacra también facilitará el trabajo de cara «á consecución dunha agricultura máis respectuosa co medio ambiente».
El representante de XX.AA. basa su explicación en que el sistema Swap «realiza as aplicacións de insumos, tales como fertilizantes, fitosanitarios, auga e outros, segundo as necesidades do cultivo en cada intre». Eso también da lugar a una reducción de los gastos al optimizar los productos.
Objetivo: superar la falta de medios y las barreras orográficas
El Instituto Tecnológico de Galicia también resalta la implantación del nuevo sistema «por permitir un avance en una zona donde las distintas barreras a la innovación que marcaron tradicionalmente el sector no dieron paso a avances en los sistemas de producción». En ese sentido se considera desde el centro tecnológico que en muchos casos se debió a la falta de medios para afrontar nuevos retos o simplemente por las dificultades que impone el medio. El proyecto, dado el sistema empleado para la transmisión de datos,salva cualquier barrera orográfica y dificultades económicas de antaño gracias al respaldo de las administraciones públicas.
En marcha
El sistema, conocido también como de monitorización de cultivos de forma remota y sin cables, ya está en marcha en Rías Baixas, en concreto en la cooperativa Condes de Albarei, y también en la Ribeira Sacra a partir de los viñedos de la bodega Guímaro, en Amandi (Sober). A mayores, la aplicación cuenta con una fase de desarrollo en los invernaderos Horsal, en Sisán (Pontevedra).
Adhesiones
Xóvenes Agricultores fomentará el empleo de la nueva tecnología desde su oficina central de Chantada para toda la Ribeira Sacra.

Per tant existeixen solucions totalment funcionals i dedicades a la producció del sector agrícola, des d'hivernacles a bodegues, o ús fitosanitari. Aquestes solucions estan en funcionament, o a prop del seu desplegament, i han afavorit la producció objecte de control. Els punts comuns de totes elles són la necessitat d'un control precís de les condicions mediambientals, i per tant de la seva monitorització constant, i el seu manteniment automàtic. Aquests projectes han demostrat que són objectius que poden ser abordats amb una solució de nodes inhalàmbrics desplegats en el lloc a controlar, per llegir els valors mediambientals, i treballant amb un sistema de control automàtic per mantenir les condicions òptimes desitjades. Tots els punts anteriors avalen finalment la solució desenvolupada en aquest projecte, i per tant les seves possibilitats comercial reals i d'implantació.

3 Descripció funcional

3.1 Sistema total

A continuació es descriu el disseny de blocs del sistema, i la interacció entre els diferents objectes que el componen. S'ha dividit en:

- Diagrama de blocs del sistema
- Descripció de l'estructura de la xarxa de objectes
- Descripció de la interacció entre els diferents components

3.1.1 Diagrama de blocs del sistema

El sistema consta del següent diagrama simplificat:

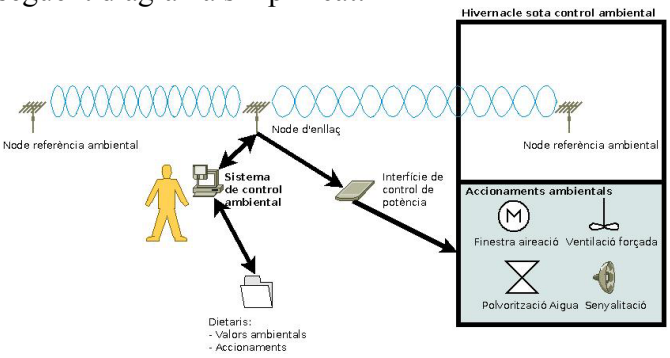


Figura 3.1. Diagrama de blocs sistema SCAFIA

Descripció:

- **Node referència ambiental**. La seva funció es obtenir lectures dels seus sensors:
  - Lluminositat
  - Efecte Hall
  - Temperatura
  - Tensió d'alimentació
  - Humitat relativa

  I transmetre-la de forma inhalàmbrica al node d'enllaç, que fa de pont en la comunicació xarxa de sensors - PC de control. Existeixen 2 diferenciacions per ubicació:
  - Instal·lació a dins de l'hivernacle. La funció és obtenir les dades ambientals del lloc a controlar, en una instal·lació real poden ser molts, tants com zones a controlar
  - Instal·lació fora de l'hivernacle. Es redueix a un únic node l'objectiu del qual és obtenir els valors ambientals externs, de cara a possibles estadístiques, valors de referència en auditories o bé millores dels processos de regulació ambiental

- **Node d'enllaç**. Té 2 funcions principals:
  - Fer de pont bidireccional entre el sistema de control ambiental, instal·lat al PC, i la xarxa de nodes de referència ambiental. L'objectiu és rebre les dades recollides pels sensors dels nodes de referència ambiental, i també transmetre comandes, des del sistema instal·lat al PC, cap la xarxa de nodes
  - Mitjançant les comandes rebudes des del sistema de control, accionar els diferents controls ambientals, mitjançant un control indirecte exercit per la interfície de potència, per mantenir els valors climàtics

- **Sistema de control ambiental**. Està format per un programari instal·lat a un PC, que està permanentment connectat al node d'enllaç de la xarxa. Les seves funcions són:
    - Recopilar el valors ambientals que rep dels diferents nodes, i també les maniobres ordenades sobre els accionaments, i gravar-los a un fitxer de dietari
    - Emetre comandes de consulta als nodes de referència ambiental
    - Emetre comandes als accionaments, mitjançant el node d'enllaç
    - Variar les condicions ambientals al variar els valors operatius

- **Interfície de control de potència**. Permet fer el pont necessari per al control de potència que no permeten les sortides del microcontrolador del node d'enllaç. Aquests dispositius normalment toleren unes potències elèctriques a les seves sortides de l'ordre dels mW, i el control dels accionaments proposats fa que necessitem W, o fins i tot KW, així que és obligatòria una adaptació. Aquesta interfície permet aïllar el senyal de control, que prové del microcontrolador i és de mW, de la potència real a controlar, de l'ordre de W/KW. Per fer-ho s'utilitzen per exemple adaptacions amb transistors MOSFET de potència, relés/commutadors de potència,...

- **Accionaments ambientals**. Mitjançant la interfície de potència, connectada al node enllaç, és possible accionar els diferents mecanismes que permeten variar les condicions ambientals:
    - Finestra d'aireació motoritzada
    - Ventilació forçada per electroventiladors
    - Control d'electrovàlvula que permet polvorització d'aigua
    - Activar senyals lluminoses i/o acústiques

3.1.2 Descripció de l'estructura de la xarxa d'objectes

Les limitacions de components fa que alguns dels rols anteriors no pugin tenir una definició tan exacta, i s'hagin de compartir o bé simular. De forma que el diagrama de blocs inicials ha patit alguna variació, així tenim:

- **Node referència ambiental**. Estan implementades totes les funcionalitats descrites a nivell de captació de valors dels seus sensors. Respecte a la diferenciació entre node a dins de l'hivernacle i node de referència a fora el projecte s'ha implementat de la següent forma:
    - un node amb configuració *mS* que correspondria a un node situat a dins de la zona a controlar
    - no existeix un node amb configuració mS que captaria valors ambientals exteriors a l'hivernacle

- **Node d'enllaç**. Continua amb les funcionalitats descrites de pont bidireccional entre el sistema i la xarxa de nodes, i la recepció de comandes de control per a la gestió dels sistemes de variació ambiental. A més s'ha afegit la configuració *mS* modificada per a transmetre el s valors captats pels seus sensors via sèrie al sistema, i així simular el node de referència ambiental

- **Sistema de control ambiental**. Implementa les funcionalitats descrites de control automàtic i registre d'auditoria

- **Interfície de control de potència**. Serà simulada amb els LED de la placa del node enllaç

- **Accionaments ambientals**. Es farà la presentació teòrica de com vincular un control de potència des d'una sortida de baixa potència, com és la que pot suportar un microcontrolador

D'aquesta forma el diagrama de blocs presentat quedarà de la següent forma que recull la realitat del projecte:

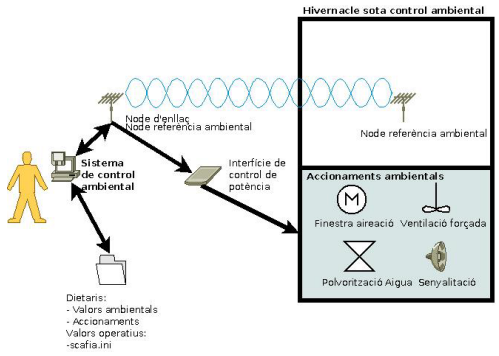


Figura 3.2. Diagrama de blocs real del sistema SCAFIA

La part d'interfície de potència i els accionaments ambientals es tracten de forma simulada.

3.1.3 Interacció entre els diferents objectes del sistema

La posició i comunicació real entre els components de la xarxa és la següent:

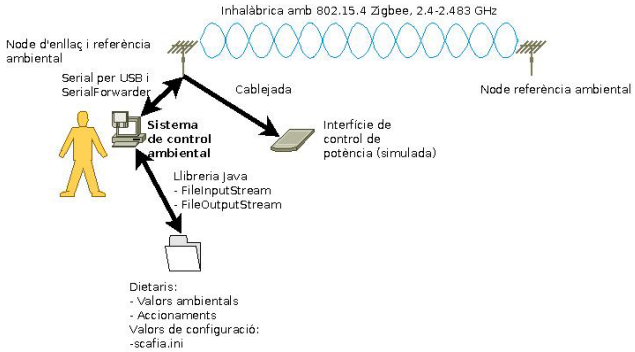


Figura 3.3. Interacció d'objectes del sistema SCAFIA

En aquest cas tenim les següents comunicacions:

- Inhalàmbriques entre el node sensor remot que estaria a dins de la zona a controlar climàticament
- Comunicació sèrie mitjançant el port USB entre la consola de control i el node d'enllaç
- Cablejada entre el node d'enllaç i la interfície de control de potència
- Utilitzant el propi sistema operatiu i les llibreries de Java per escriure els valors d'auditoria i control estadístic, així com per llegir la configuració del sistema, als fitxers corresponents

3.1.3.1 Telemetria de valors climàtics

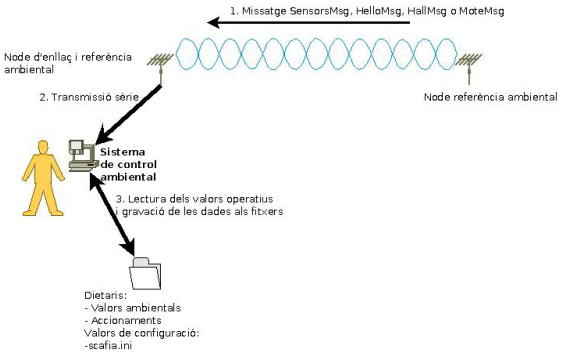


Figura 3.4. Interacció d'objectes del sistema per obtenir telemetria

El node sensor segons el temps programat de lectura llegeix els valors dels sensors i els envia al node d'enllaç en un missatge. El node enllaç, per comunicació sèrie amb un port USB, envia el missatge a la consola fent servir l'aplicació *SerialForwarder* que permet lligar el port de comunicació entre l'aplicació ScafiaCon i el node d'enllaç. Aquest missatge serà processat per la lògica programada a ScafiaCon i gravada als fitxers de dietari i estadístics.
El node enllaç simula també la funció de node de referència, es a dir llegeix també dades dels sensors. El seu funcionament és igual al descrit anteriorment, només que la transmissió es fa directament sobre la connexió sèrie implementada sobre el port USB.
La mateixa comunicació s'aplica al missatge d'activació d'un dels nodes a la xarxa.

3.1.3.2 Control d'accionaments

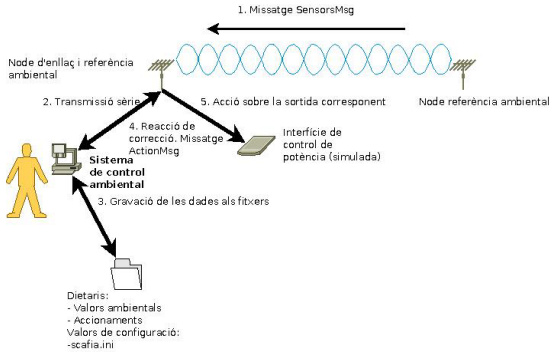


Figura 3.5. Interacció d'objectes del sistema per al control d'accionaments

La consola ScafiaCon pot automàticament variar l'estat dels accionaments, per corregir els valors climàtics reals als programats. També es pot fer manualment variacions a tots els accionaments. En tots dos casos ordre de control queda encapsulada en un missatge ActionMsg i per via del SerialForwarder, i la comunicació sèrie, arriba al node d'enllaç. Aquest interpreta el missatge i activa o desactiva la sortida corresponent del seu microcontrolador. L'acció queda enregistrada al fitxer d'auditoria.

3.1.3.3 Variació del temps de lectura de sensors

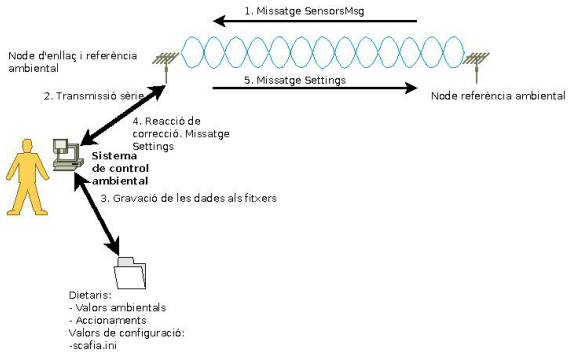


Figura 3.6. Interacció d'objectes del sistema per variar el temps d'un node remot

En aquest cas el missatge Settings, generat per la consola, es fa de forma automàtica, quan detecta que un dels nodes no respecta el termini fixat entre lectures. El valor de referència està al fitxer scafia.ini i es comparat amb el valor rebut a cada missatge SensorsMsg, que inclou aquesta dada. El missatge viatja fins al node infractor utilitzant l'aplicació **SerialForwarder**, la comunicació sèrie i si s'escau la inhalàmbrica, ja que pot afectar al node enllaç o al remot, ja que tots dos fan lectures dels seus sensors.
Al fitxers d'auditoria es guarden tots els valors de telemetria rebuts.

3.2 Disseny de la interfície d'usuari ScafiaCon

L'usuari de l'aplicació utilitza un programa desenvolupat en Java que permet:

- Activar les tasques d'auditoria
- Visualitzar en temps real que està passant al sistema
- Comunicar ordres de control als accionaments
- Variar diferents paràmetres operatius

Una descripció completa del funcionament, i la interacció amb la consola del sistema, està descrita a l'annex d'aquesta memòria. En aquest apartat només es farà una descripció funcional del seu funcionament.

La consola o interfície d'usuari consta de l'execució en línea de comandes del sistema de la següent instrucció:

java files.ScafiaCon [-opcions]

Les diferents opcions poden ser visualitzades si no es fa constar cap o bé és errònia, ja que l'aplicació inclou una petita ajuda a l'estil *man d'UNIX*. De forma breu direm que poden ser de control (verificar una mota si està encesa, activar un accionament,...) o de configuració de valors operatius (fixar la temperatura límit, llindars d'humitat relativa, ...). El diagrama de blocs i funcionament és el següent:

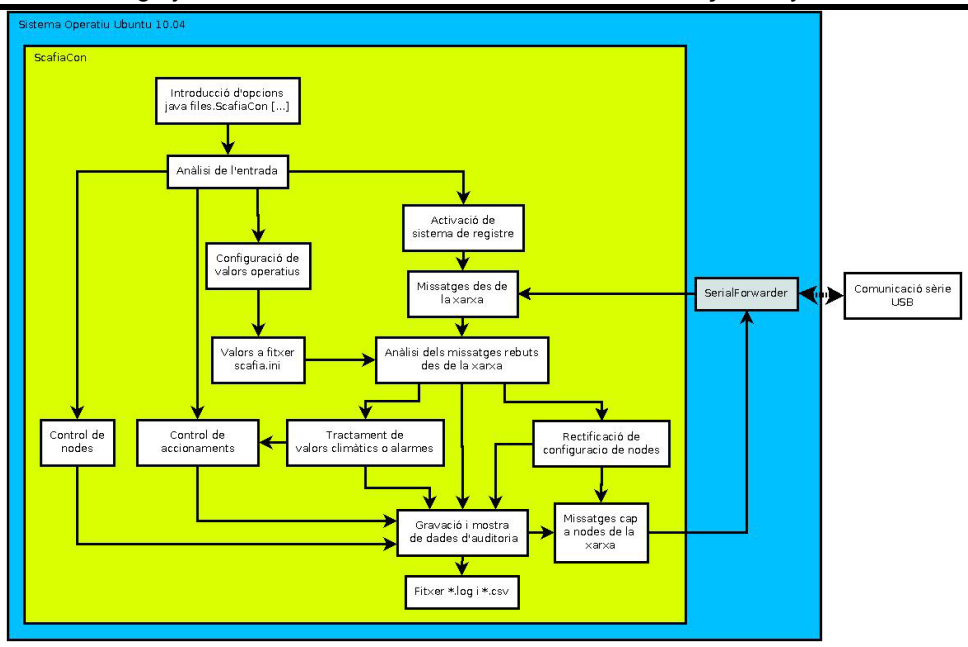


Figura 3.7. Diagrama de blocs de la consola del sistema SCAFIA

El sistema està instal·lat sobre el sistema operatiu Ubuntu versió 10.04, i aprofita les llibreries de Java 1.6 i també la connexió de port USB amb l'aplicació SerialForwarder que està disponible al sistema TinyOS 2.1.1. Una vegada iniciada la consola del sistema, s'analitzen les opcions indicades a la línia de comandes. Aquestes poden ser:

- Control d'accionaments
- Configuració de paràmetres operatius
- Inici del sistema de registre dels valors ambientals i també del comandament automàtic dels accionaments per mantenir-los
- Una combinació de les anteriors

Una vegada introduïda la línea de comandes, s'analitza si les opcions són vàlides, i si aquestes ho son llavors es passa a executar les instruccions corresponents. L'activació del sistema de registre fa que s'analitzin els missatges que es van reben des de la xarxa de nodes, mitjançant la comunicació sèrie amb el node enllaç. Així podem saber si els valors climàtics operatius es mantenen en els llindars correctes, gravar-los al fitxer d'auditoria o rectificar algun període de lectura d'algun node (configuració del temps cíclic de lectura). També si és necessari activar algun sistema climàtic per modificar un valor fora de llindar, per exemple la temperatura.
Les comandes de control de nodes i activació d'accionaments són també registrades als fitxers d'auditoria, i enviades al node corresponent. Així si una vegada iniciada la gravació de dades des de la xarxa, i el manteniment automàtic dels valors operatius, s'inicia una altra consola de sistema, en qualsevol moment variar un valor de configuració o fer una acció de control sobre algun node de la xarxa, i sempre quedarà registrada.

3.3 Disseny del node o mota d'enllaç (mB)

El node o mota d'enllaç té com a funcions:

- Fer d'enllaç entre la xarxa de nodes i el sistema ScafiaCon mitjançant la ràdio
- Llegir els seus sensors i transmetre-los via sèrie al sistema
- Activar o desactivar accionaments segons l'ordre que rebi

Aquestes funcions s'han dividit en els següents components que són carregats a la configuració del node/mota:

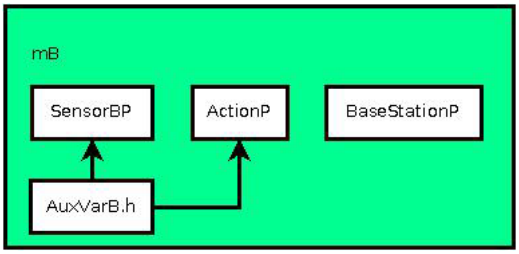


Figura 3.8. Diagrama de blocs del node d'enllaç (mB)

El fitxer **AuxVarB.h** conté els esquemes dels missatges que seran transmesos, entre el sistema i la mota, pels components ActionP i SensorBP.

Component **BaseStationP**

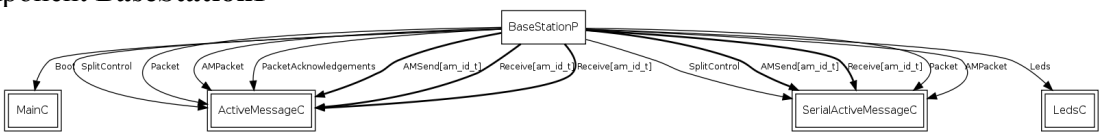


Figura 3.9. Interfícies i relacions del component BaseStationP

Fa l'enllaç sèrie i ràdio. La raó dels 2 tipus és què la mota fa de pont entre la xarxa de sensors de forma inhalàmbrica i la comunicació sèrie pel port USB.

Component **ActionP**

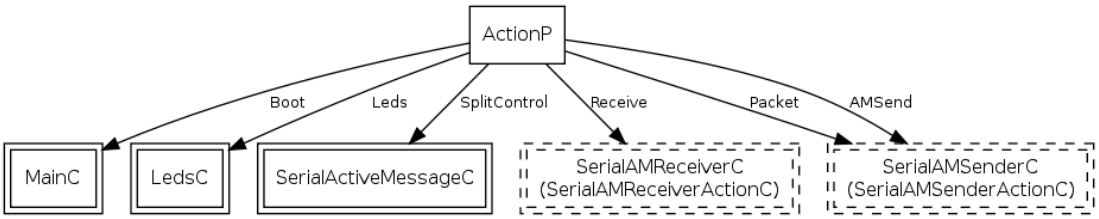


Figura 3.10. Interfícies i relacions del component ActionP

Aquest component seria l'encarregat de controlar l'activació/desactivació d'accionaments. Per aquest motiu rep la comunicació per via sèrie i farà el que li ordena la consola del sistema. També torna el resultat d'una activació/desactivació per avisar al sistema. El desenvolupament només és una simulació, ja que no s'ha implementat un placa de control de potència, i serà visualitzat el comportament sobre el LED 0 (vermell) del propi node.

Component **SensorBP**

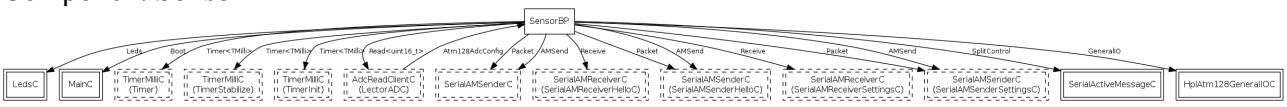


Figura 3.11. Interfícies i relacions del component SensorBP

Aquest component fa la lectura cíclica dels valors dels seus components i els transmet via sèrie per a què siguin recollits pel sistema. Per aquest motiu implementa totes les interfícies de lectura de convertidors ADC, temporització, gestió LED i comunicació sèrie.

3.4 Disseny del node o mota remot (mS)

El node o mota remot té com a funció recollir els valors captats de forma regular pels seus sensors i transmetre'ls al node d'enllaç per al seu tractament pel sistema. Per aquest motiu els components principals estan dedicats a la recollida de dades, verificació de funcionament, i la seva transmissió.

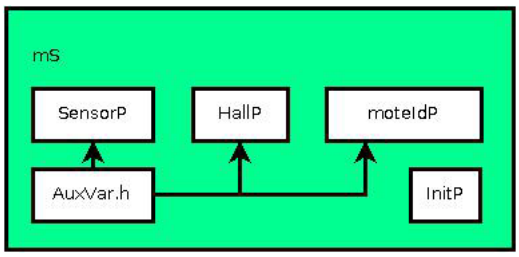


Figura 3.12. Diagrama de blocs del node remot (mS)

El fitxer **AuxVar.h** conté els esquemes dels missatges que seran transmesos, entre el sistema i la mota, pels components HallP, moteIdP i SensorP.

Component **InitP**

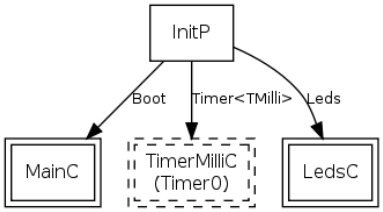


Figura 3.13. Interfícies i relacions del component InitP

Aquest component només serveix per advertir amb senyalització lluminosa que el node s'ha iniciat. Ho fa fent un cicle ràpid dels seus LEDS Verd->Verd+Groc->Verd+Groc+Vermell, després Verd i Vermell i finalment Verd, que indicaria que el node està iniciat. Per aquest motiu només fa servir les interfícies de temporització i control de LED.

Component **MoteIdP**

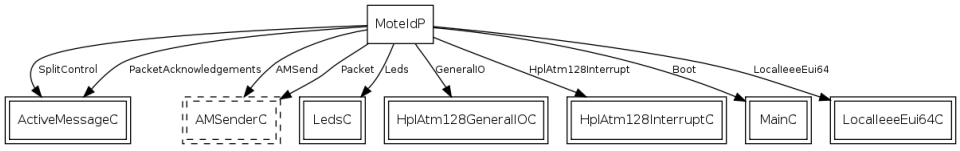


Figura 3.14. Interfícies i relacions del component MoteIdP

Aquest component envia el valor de TOS_NODE_ID, o valor d'identificació del node dins de la xarxa, i la seva adreça MAC. Quan pressionem el botó d'usuari que porta integrat la mota. Així que utilitza les interfícies de comunicació via ràdio, LED, identificació del node i gestió d'interrupcions (es produeix una interrupció al polsar el botó).

Component **HallP**

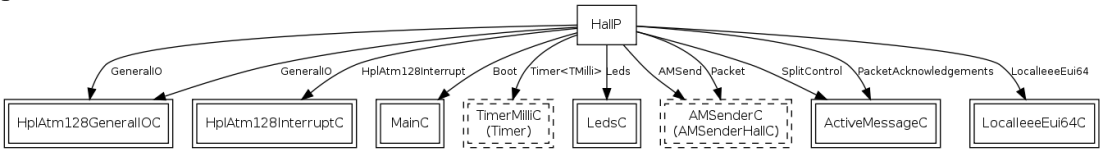


Figura 3.15. Interfícies i relacions del component HallP

La detecció de l'activació del sensor Hall es fa amb aquest component. Igual que en cas anterior es fa per detecció d'interrupció (flanc de pujada). Les interfícies que s'utilitzen són les gestió d'enviament per enllaç ràdio, gestió d'interrupció, temporitzador i identificació del node.

Component **SensorP**

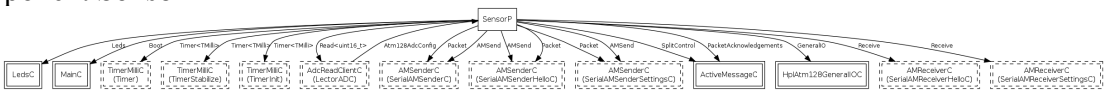


Figura 3.16. Interfícies i relacions del component SensorP

Aquest component realitza la lectura dels valors dels sensors que té integrats: tensió de bateria, lluminositat, temperatura i humitat; cada cert període de temps que té programat. Per fer-ho fa servir els convertidors ADC que té integrats el microcontrolador, la transmissió es realitza via ràdio. Les interfícies utilitzades són les de gestió de LED, temporitzador, convertidor ADC i gestió de comunicació ràdio.

4. Descripció tècnica del sistema

A continuació es presenten quins són els fonaments tècnics que s'han fet servir per donar compliment al capítol 3.

4.1 Sistema total

El sistema contempla els següents casos d'ús segons l'estudi de l'anàlisi funcional:

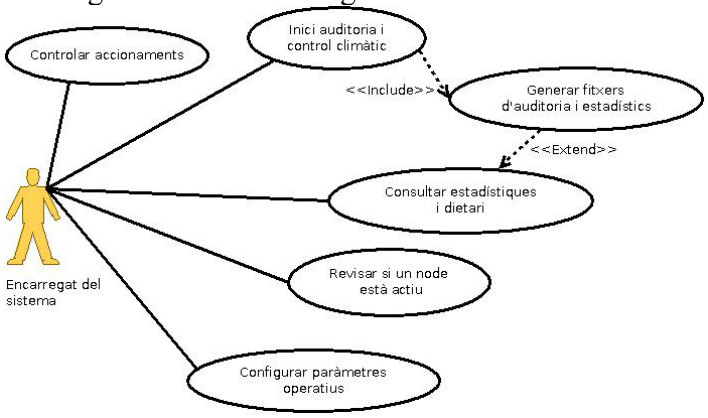


Figura 4.1. Casos d'ús de la consola del sistema SCAFIA

L'encarregat del sistema pot optar per alguna de les diferents opcions en qualsevol moment, ja que la consola pot obrir-se tot i que s'estigui fent una tasca de registre d'auditoria del sistema. Així podem controlar accionaments, revisar els nodes o configurar paràmetres operatius en qualsevol moment. Només s'ha d'introduir a la consola de sistema la línea de comanda apropiada. La consulta del dietari i els valors estadístics es realitza sobre els fitxers de .log i .csv, que el sistema genera o ha generat, però no s'estableix cap mecanisme des de la consola. Al ser dos fitxers amb un format universal es deixa a l'encarregat el seu accés, per exemple amb un editor de text o bé un full de càlcul.

4.2 Consola de sistema ScafiaCon

L'aplicació està detallada de forma inicial tant a la proposta com al pla de treball, que es va desenvolupar per portar a terme la seva realització. De forma resumida ha de complir les següents fites:

- Obtenir de forma contínua les dades dels sensors que incorpora cada node de la xarxa. Els sensors són: tensió de bateria, lluminositat, temperatura i humitat relativa

- Analitzar les dades rebudes fent servir uns llindars preprogramats
- Segons l'anàlisi anterior activar alarmes i dispositius externs que permetin mantenir les condicions climàtiques desitjades. També desactivar les alarmes i accionaments quan ja no siguin necessaris. Si es rebassen els límits màxims del sistema la correcció correspon a l'encarregat humà de la instal·lació
- Activar i desactivar a voluntat els accionaments/dispositius externs. En aquest cas s'ha programat un component, que facilita aquest control, sobre el node d'enllaç entre la xarxa de sensors i el PC de sistema. No s'ha fet el desenvolupament total en maquinari, només la part de programari
- Activar alarmes si detecta que no existeix comunicació entre la xarxa de nodes remots i el sistema
- Tant les dades com les accions que es porten a terme, de forma automàtica o manual, han de quedar registrades en un fitxer d'auditoria (log). Les dades climàtiques a més quedaran registrades en un tipus de fitxer que permeti posteriorment el seu anàlisis estadístic

La consola de sistema té el següent diagrama de flux:

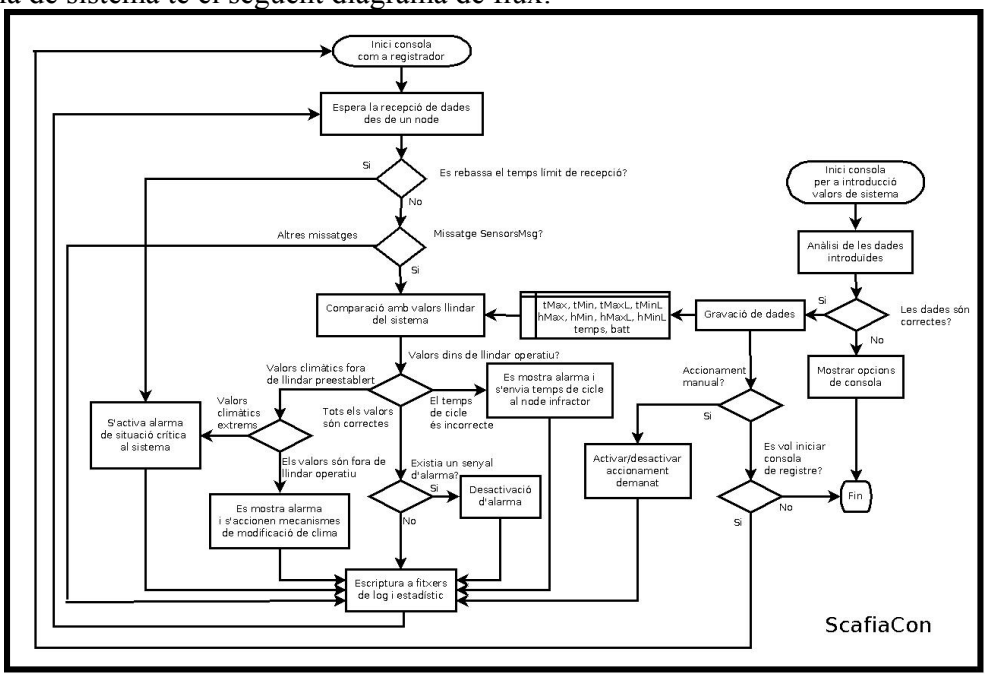


Figura 4.2. Diagrama de flux de la consola de sistema

El funcionament, com a registrador i control climàtic, és el següent:

- El sistema té configurat uns valors climàtics determinats, i que són l'objectiu a mantenir. També té uns valors límit. i que són considerats crítics (tenen la identificació xL). i que teòricament mai s'haurien de rebassar. En cas de superar-se s'activarà una alarma general del sistema i de forma manual l'encarregat humà d'aquest l'haurà de retornar a la normalitat
- Els nodes envien el seus valors climàtics, captats als sensors que porten, de forma regular segons el temps programat. Es poden produir dos situacions:
    - Que un node enviï els seus valors fora del temps predeterminat. Serà detectat i automàticament rep un missatge per què canviï la periodicitat a la programada
    - Que no existeixi comunicació des de la xarxa de nodes. S'ha establert que si com a mínim en el doble de temps de periodicitat de lectures de valors, no es rep una

comunicació del node de xarxa llavors s'activarà una alarma general del sistema per a que l'encarregat humà verifiqui que està passant. Per exemple si cada 10 segons hem de rebre dades d'algun dels nodes de xarxa, si passats almenys 20 segons no rebem comunicació llavors s'activarà l'alarma general del sistema

- Les dades climàtiques que va rebent de cada sensor són analitzades contra els valors climàtics predeterminats. Es poden produir 2 situacions de control automàtic:
  - Es detecta una infracció de valor climàtic entre els valors operatius. En aquest cas s'activa un senyal d'alarma, i també l'accionament que ha de restar l'efecte climàtic. Un cop recuperat els valors adequats el senyal d'alarma i els accionaments activats per aconseguir-ho desapareixen
  - Es detecta una infracció de valor climàtic que supera els límits del sistema. En aquest cas s'activa l'alarma general del sistema, i es deixa a l'operador humà la correcció climàtica
- Les dades climàtiques queden registrades a un fitxer .csv que permet fàcilment la seva importació per sistemes d'anàlisi estadístic. El nom del fitxer té la següent configuració:
  - scafia_aaaammdd.csv (scafia_anymesdia.csv)
- Totes les accions anteriors, manuals o automàtiques, així com les manipulacions de consola, queden registrades a un fitxer de dietari (log) accessible amb qualsevol editor de text, amb el següent nom
  - scafia_aaaammdd.log (scafia_anymesdia.log)

Els fitxers de classes que contenen la programació de la consola de sistema **ScafiaCon** són:

ScafiaCon.java (classe principal)
Utilitats.java (classe de suport a la classe principal)

A més dels generats automàticament per l'aplicació *mig* fent servir la instrucció:

```
mig –target=cou24 –java-classname=[classe] [fitxer d'estructures] [estructura] –o [fitxer java de sortida]
```

Per exemple la generació de la classe ActionMsg.java és la següent instrucció:

```
mig –target=cou24 –java-classname=ActionMsg AuxVarB.h ActionMsg –o ActionMsg.java
```

Els resultats són tants com estructures de dades emprades al sistema:

ActionMsg.java
HallMsg.java
HelloMsg.java
MoteMsg.java
SensorsMsg.java
Settings.java

4.3 Node remot (*mS*)

Aquest node té les següents funcions:

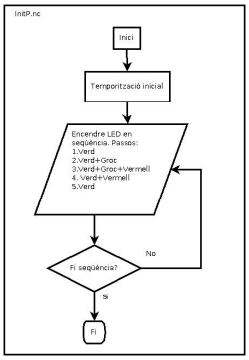
- Lectura dels valors dels seus sensors de forma cíclica, i dins d'un temps definit
- Contestar missatges de comprovació de funcionament
- Indicar si s'ha activat el seu sensor Hall
- Enviar un missatge de comprovació quan es polsat el seu botó d'usuari

Els components que composen la programació d'aquest node tenen els següents noms:

- AuxVar.h: estructura dels missatges
- InitP.nc i InitC.nc: inici del node, visualitzat pels seus LED

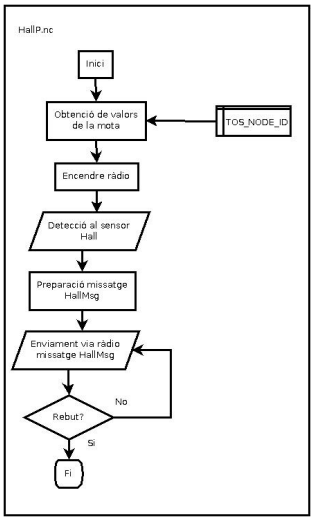
- HallP.nc i HallC.nc: gestió del sensor Hall
- moteIdP.nc i moteIdC.nc: missatge d'activació del node
- SensorP.nc i SensorC.nc: lectura de sensors i missatges d'activació, i el seu enviament

Es descriuen ara els diagrames de flux dels diferents components.



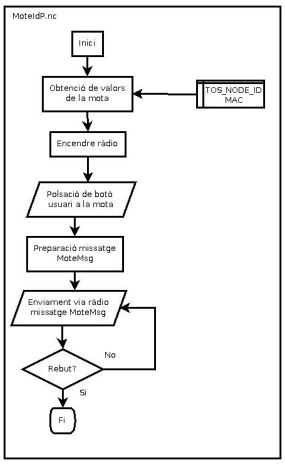
El diagrama de flux del component **InitP** es mostra a continuació. Només es fa des de l'inici una representació visual amb els LED que integra la placa del node. S'estableix un temporitzador i cada cop que aquest finalitza el temps va variant els LED encesos, fins que al final només queda el de color verd encés, moment en que es considera que el node s'ha iniciat correctament.

Figura 4.3. Diagrama de flux component InitP



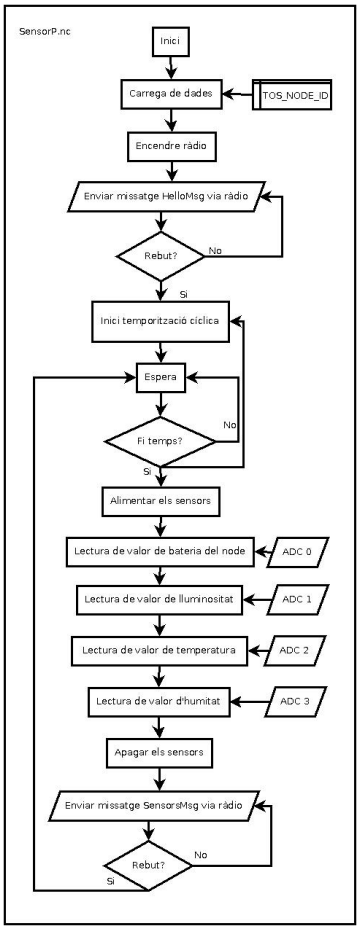
El següent component és **HallP**, i la seva funció és la d'enviar al sistema la detecció d'un accionament per exemple amb el sensor Hall integrat a la placa del node. En aquest cas una vegada s'inicia el component, primer obté els valors d'identificació del node per així transmetre'ls al sistema via ràdio quan el seu sensor Hall sigui activat (flanc de pujada). El missatge té comprovació d'enviament amb l'interfície *PacketAcknowledgements*, així que si no té rebut de lliurament el tornarà a enviar.

Figura 4.4. Diagrama de flux component HallP



El component **MoteIdP** envia via ràdio els valors d'identificació del node quan es pren el botó d'usuari del node. Aquesta acció manual genera un missatge *MoteMsg* que s'envia al sistema. Així es pot comprovar si un node està actiu i si arriba efectivament al node d'enllaç del sistema.

Figura 4.5. Diagrama de flux component MoteIdP



El següent component, **SensorsP**, és el més complex de tots. I té diverses parts com són la lectura i transmissió dels valors dels seus sensors, i també la variació del valor del cicle de lectura, que no es representa en diagrama de flux ja que només es tracta de la recepció d'un missatge *Settings* que forçarà un canvi de variable. La part complexa és la que es representa en el següent diagrama:
Existeix una primera part quan s'activa aquest component, que és la de càrrega del valor d'identificació del node i després d'encendre la ràdio l'envia per avisar al sistema que el node ja té aquest component operatiu (*HelloMsg*). Igual que en tots els enviaments serà obligatori rebre el rebut de transmissió correcta per que es deixi de transmetre el missatge. Una vegada lliurat el missatge s'inicia un temporitzador que de forma cíclica engegarà l'alimentació dels sensors, i un a un llegirà el seu valor gràcies al convertidor ADC relacionat. Una vegada llegits tots els sensors s'apagarà la seva alimentació, per a estalvi energètic, i s'enviarà un missatge *SensorsMsg* amb aquests valors al sistema. El cicle de lectura de sensors continuarà de forma ininterrompuda fins que es desconnecti el node o mota.

Figura 4.6. Diagrama de flux component SensorsP

L'estructura dels missatges que es fan servir, i que estan inclosos al fitxer *AuxVar.h* són els següents:

- Registre o estructura del missatge de valors de sensors del node

```
typedef nx_struct SensorsMsg {
  nx_uint16_t moteId;     //Identificació del node
  nx_uint16_t countsTemp;  //Valor de temperatura
  nx_uint16_t countsPhoto; //Valor de lluminositat
  nx_uint16_t countsBat;//Valor de bateria interna
  nx_uint16_t countsHum;   //Valor d'humitat
  nx_uint16_t vrefmilivolts; //Tensió de referència de la bateria
  nx_uint32_t militimer;  //Valor del cicle de lectura de sensors
  nx_uint32_t counter;        //Comptador o seqüència interna
} SensorsMsg;
```

- Registre o estructura del missatge d'inici del node

```
typedef nx_struct HelloMsg{
  nx_uint16_t moteId;        //Identificació del node
  nx_uint32_t seqNum;     //Identificador del missatge
  nx_uint32_t militimer;   //Temps de cicle de lectura de sensors programat
} HelloMsg;
```

- Registre o estructura del missatge d'activació del sensor Hall del node

```
typedef nx_struct HallMsg{
  nx_uint16_t moteId;        //Identificació del node
} HallMsg;
```

- Registre o estructura del missatge amb id del node i la seva MAC que s'envia al polsar el botó d'usuari

```
typedef nx_struct MoteMsg{
  nx_uint16_t moteId;        //Identificació del node
  nx_uint8_t macAddr[8];   //Identificació del node MAC_address
}MoteMsg;
```

- Registre o estructura del missatge de configuració del temps de cicle de lectura

```
typedef nx_struct Settings{
  nx_uint16_t moteId;        //Identificació del node
  nx_uint32_t setmilitimer;  //Temps de cicle de lectura de sensors
  nx_uint32_t seqNum;   //Comptador o seqüència interna
}Settings;
```

4.4 Node d'enllaç (*mB*)

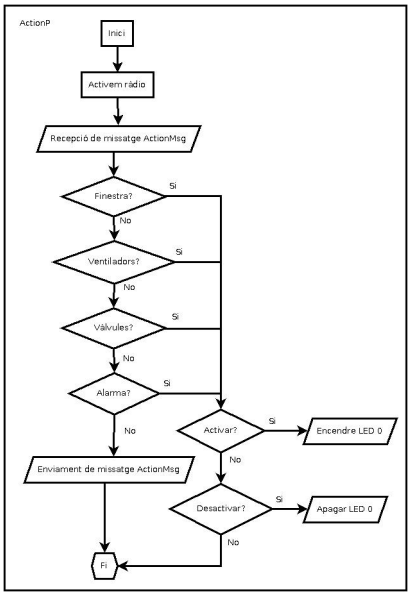
Aquest node té les següents funcions:

- Enllaç de comunicació entre la xarxa i el sistema (ScafiaCon)
- Fer de node de referència climàtica
- Accionar mecanismes per mantenir les condicions climàtiques, o avisar d'alarmes

Els components que composen la programació d'aquest node tenen els següents noms:

- AuxVarB.h: estructura dels missatges
- ActionP.nc i ActionC.nc: control d'accionaments
- SensorBP.nc i SensorBC.nc: lectura de sensors i missatges d'activació
- BaseStationP.nc i BaseStationC.nc: enllaç de comunicació sèrie/ràdio

Els diagrames de flux només fan referència al component ActionP.nc, ja que SensorBP.nc té el mateix funcionament que el seu homòleg en la configuració del node *mS*, excepte que en aquest cas la comunicació utilitzada és serial i no ràdio.



En aquest cas després d'activar la ràdio, quan es rep un missatge ActionMsg s'analitza per saber a quin dels accionaments fa referència, i també quina acció s'hauria d'executar (activar/desactivar). Una vegada realitzada l'acció demanada, es torna un missatge al sistema per a que tingui la conformitat de que aquesta s'ha realitzat.

Figura 4.7. Diagrama de flux component ActionMsg

L'estructura dels missatges que es fan servir, i que estan inclosos al fitxer *AuxVarB.h* són els següents:

- Registre o estructura del missatge de valors de sensors del node

```
typedef nx_struct SensorsMsg {
  nx_uint16_t moteId;     //Identificació del node
  nx_uint16_t countsTemp; //Valor de temperatura
  nx_uint16_t countsPhoto; //Valor de lluminositat
  nx_uint16_t countsBat;//Valor de bateria interna
  nx_uint16_t countsHum;  //Valor d'humitat
  nx_uint16_t vrefmilivolts; //Tensió de referència de la bateria
  nx_uint32_t militimer; //Valor del cicle de lectura de sensors
  nx_uint32_t counter;        //Comptador o seqüència interna
} SensorsMsg;
```

- Registre o estructura del missatge d'inici del node

```
typedef nx_struct HelloMsg{
  nx_uint16_t moteId;       //Identificació del node
  nx_uint32_t seqNum;   //Identificador del missatge
  nx_uint32_t militimer; //Temps de cicle de lectura de sensors programat
} HelloMsg;
```

- Registre o estructura del missatge de configuració del temps de cicle de lectura

```
typedef nx_struct Settings{
  nx_uint16_t moteId;         //Identificació del node
  nx_uint32_t setmilitimer;  //Temps de cicle de lectura de sensors
  nx_uint32_t seqNum;   //Comptador o seqüència interna
```

```
}Settings;
```

- Registre o estructura del missatge d'activació/desactivació d'accionaments

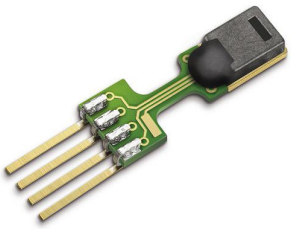
```
typedef nx_struct ActionMsg{
  nx_uint16_t moteId;     //Identificació del node
  nx_uint8_t action;    //Accionament que volem activar/desactivar
  nx_uint8_t maniobra; //1 = activar; 0 = desactivar
  nx_uint32_t seqNum; //Comptador o seqüència interna
}ActionMsg;
```

5. Adaptació dels nodes als requeriments del projecte

El projecte tenia dos fites importants, i que no estaven recollides en el maquinari dels nodes. Per un costat es pretenia recollir dades sobre la humitat relativa i així completar els valors climàtics registrats pel node. Per l'altre es volia controlar també des del microcontrolador del node un accionament de potència, això fa que necessariament s'hagi de fer una adaptació que permeti watts o kilowatts de control, i no els miliwatts (mW) de potència de sortida de l'electrònica del node. Malauradament aquesta última fita només es podrà indicar de forma teòrica.
Els següents punts tracten sobre aquestes adaptacions.

5.1 Adició d'un sensor d'humitat als nodes

La cerca de components imposava límits clars: la seva disponibilitat, el seu preu i l'adaptació a l'electrònica del node. Per tant es tractava de trobar un component electrònic de baix cost i que fos fàcilment afegible al node, per des de el convertidor ADC lliure llegir els valors que detectava i enviar-los al sistema. No va ser una tasca fàcil, primer per què existeixen components d'alta qualitat, però precisament per això feien que l'adaptació a l'electrònica del node no fos fàcil.



Un exemple el trobem al component SH71 de Sensirion, que amb un relatiu baix cost, aproximadament 30€, permet la lectura de la humitat relativa a una tensió nominal de 3.3V, tot i que funciona en un rang de 2.4 a 5.5 volts. El principal problema és que la lectura s'ha de fer amb una connexió de dades sèrie des del controlador, I^2C, rellotge inclòs, i aquesta connexió complica la seva integració.

Figura 5.1.Sensor Sensirion SH71

La solució triada finalment ha estat un sensor dels anomenats humistors, que té un funcionament similars als components que ja té instal·lat el node: rep alimentació elèctrica i varia el seu valor proporcionalment a com varia la humitat. En aquest cas vaig trobar que el fabricant Honeywell ven el model HIH-5031 que compleix totes les expectatives de disseny: costa 11€, la disponibilitat era quasi immediata (compra a la web de l'empresa *Farnell*) i es podia adaptar perfectament a la nostra placa del node, tant per la seva electrònica, similar a la dels altres sensors encastats, com per la tensió d'alimentació (2.7 a 5.5 volts).

Figura 5.2. Sensor HIH-5031

Les condicions operatives d'aquest sensor són les següents:

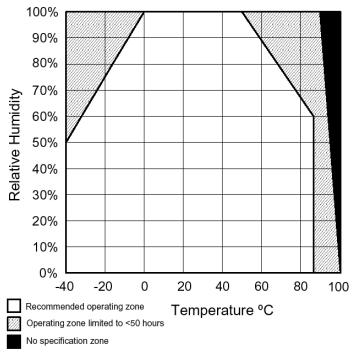


Figura 5.3. Condicions operatives HIH-5031

On veiem que s'adapta a les condicions climàtiques d'un hivernacle, ja que poden variar entre els 20 i els 35 graus centígrads típicament.

El circuit d'aplicació típic és el següent:

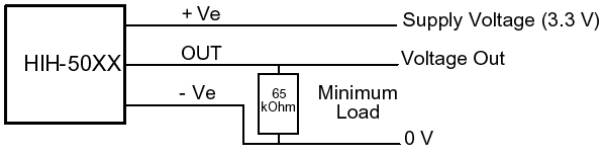


Figura 5.4. Connexió elèctrica HIH-5031

Només hem de connectar les patilles:

+Ve a la tensió d'alimentació
OUT a l'entrada del convertidor ADC 3
-Ve a la connexió *sensor sink*

Que segons la informació sobre el maquinari correspon a les connexions del port d'expansió de la placa electrònica següent:

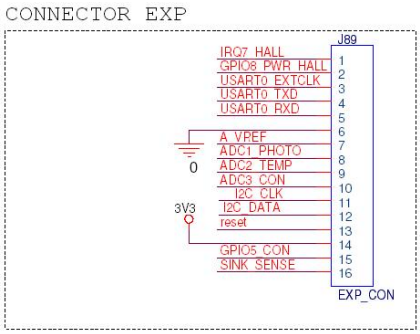


Figura 5.5. Detall del port d'expansió disponible a COU_1_2

Per tant s'han de fer les següents connexions:

+Ve a la tensió d'alimentació, que la tenim a la patilla 14
OUT a l'entrada del convertidor ADC 3, que està connectat a la patilla 9
-Ve a la connexió *sensor sink*, que està en la patilla 16

Un canvi acceptat pel fabricant és la variació de la resistència de sortida del sensor. Enlloc d'instal·lar una d'un valor de 65 kOhm, s'ha instal·lat una de 100 kOhm, que no ha d'implicar cap problema ja que els 65 kOhm és un valor de càrrega mínim (a més reduïm el consum elèctric a l'augmentar aquesta resistència).

Els components utilitzats són, apart d'uns cables i una petita *protoboard*, el sensor d'humitat i la resistència de 100 kOhm:

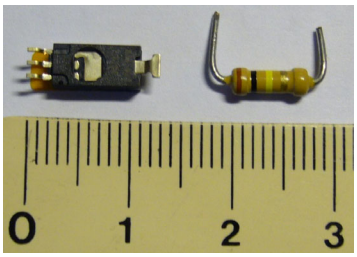


Figura 5.6. HIH-5031 i resistència de 100 kOhms

El resultat final és el següent:

Figura 5.7. Imatges d'una de les motes amb el sensor d'humitat HIH-5031 d'Honeywell integrat

A l'annex es descriuen les formules de conversió, segons la corba del fabricant i les seves recomanacions, per mostrar els valors llegits d'humitat correctament a la consola del sistema.

5.2 Adaptació de la sortida del microcontrolador

Segons el fabricant, la sortida del microcontrolador és de mW, clarament insuficient per activar un dispositiu elèctric de potència. Així que s'imposa l'adaptació mitjançant una interfície que permeti fer de pont entre les potències en joc. Es tracta doncs de fer una cosa molt freqüent en l'electrònica de potència, on des de circuits de control de baixa potència, microcontroladors o autòmats programables, s'han de controlar processos industrials de gran potència(per exemple una locomotora elèctrica). Com que es tracta de fer un circuit simple de demostració, i no coneixem el consum final de la instal·lació, el que se suggereix és:

> La sortida del microcontrolador serà connectada a un transistors de tipus MOSFET. El motiu d'utilitzar aquesta tecnologia és que s'activen per tensió, i la seva resistència entre graduador i sortidor (GS) és tant alta que el corrent elèctric que pot passar és menyspreable.
> El transistor actuarà com a interruptor, es a dir en tall o saturació només, permeten la conducció a través de la seva estructura i per tant activant o no la càrrega. La saturació es produeix quan la sortida del microcontrolador està en estat alt (simulat amb el LED quan aquest està encès), fet que al posar una tensió entre GS fa conduir al transistor
> La càrrega estarà formada per un relé que a l'activar-se pot fins i tot controlar càrregues de baixa potència elèctrica (electrovàlvules, alarmes acústiques o lluminoses, ...)

Les característiques anteriors constituirien el següent circuit

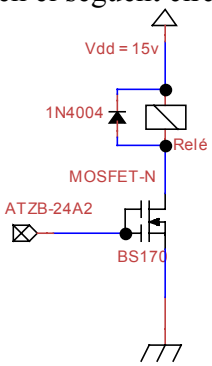


Figura 5.8. Circuit de control de relé

On tenim:

- **Vdd**. Tensió d'alimentació, de valor suficient per afavorir la commutació del MOSFET
- **Diode 1N4004**. Es tracta d'un diode de protecció davant de la descàrrega de la bobina del relé, que si no és controlada podria danyar els circuits electrònics. La descàrrega es produeix quan deixa de conduir-se un corrent elèctric per la bobina del relè
- **Relé**. És un component que permet accionar càrregues amb un consum de potència diferent al del circuit del seu control. Funciona amb una bobina que al passar corrent elèctric per ella acciona un actuador (solenoide). Aquest actuador controla uns contactes de forma que commuta la seva posició (que pot estar normalment tancat NC, o normalment obert NO), i així es permet que s'alimenti el sistema de potència. Podem dir que és un interruptor comandat elèctricament
- **MOSFET BS170**. Aquest és de tipus enriquiment, que és el típic en circuits on necessitem un commutador normalment obert, i aquesta serà la seva característica de funcionament: fer de commutador electrònic. Quan al graduador existeix un nivell de tensió superior a la llindar $V_{GS}(th)$, típicament la corresponent a un nivell 1 o alt a tecnologia TTL, el transistor comença a conduir entre drenador i sortidor, i per tant activant el relé i la càrrega controlada per aquest. Quan la tensió passa V_{GS} a valor 0, el transistor deixa de conduir i per tant el relé commuta a la seva posició de repòs, i se desactiva l'accionament lligat a ell. La tensió d'activació prové d'una sortida del microcontrolador ATZB-24A2, connectada directament al graduador (G) del MOSFET. El seu sortidor (S) està connectat a massa, i el drenador (D) a l'alimentació elèctrica Vdd mitjançant la bobina del relé

El circuit final controlant els 4 accionaments podria ser el de la següent figura:

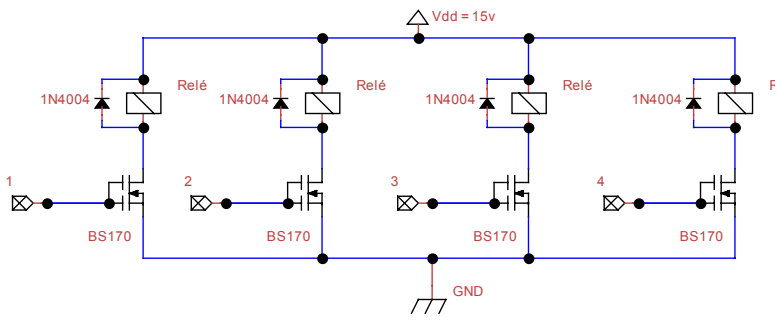


Figura 5.9. Interfície de control de potència

On les entrades correspondrien a les sortides de l'ATZB-24A2 amb el significat:

1: sortida per al control de finestres
2: sortida per al control dels electroventiladors
3: sortida per al control de les electrovàlvules
4: sortida per al control de les alarmes crítiques del sistema

La potència total controlada dependria del valor nominal de cada relé, i l'alimentació elèctrica de cada accionament quedaria totalment separada del circuit de la mota, o node d'enllaç, gràcies també al relé.

6. Viabilitat tècnica

Una pregunta obligada és si el sistema desenvolupat en aquest projecte, és comercialment viable. La resposta crec que sincerament és sí, però limitada a explotacions agrícoles de baix nivell de complexitat o requeriments, bàsicament per què només es guarden uns pocs paràmetres ambientals, i s'apliquen unes mesures correctores que segurament són molt millorables, ja que només actuen tres accionaments correctors. A més la consulta de les dades es fa sobre un fitxer pla, cosa que tampoc ajuda si es vol una gestió ràpida de les dades i que pugui ser importada per altres sistemes.
La viabilitat tècnica s'ha demostrat al programar uns nodes/sensors i una aplicació Java que, tot i que de forma limitada, monitoritza en temps real la situació climàtica i la intenta mantenir dins d'uns paràmetres configurats. Com després es presenta a l'apart de proposta de millores, podem extreure molt més de sistemes d'aquest tipus. Només hem de combinar la seva flexibilitat d'instal·lació, baix consum i adaptabilitat, demostrada a l'instal·lar un sensor als existents, i que són els seus veritables punts forts, amb un disseny de sistema distribuït que inclogui una gestió de les dades més potent, per exemple amb una base de dades relacional. Per contra s'haurà de treballar en crear un node/mota sensor que mecànicament sigui més resistent a les condicions climàtiques (temperatura i estanquitat), i al que es puguin adaptar fonts d'energia que no restin flexibilitat (plaques solars).
La suma dels conceptes anteriors permeten la construcció de sistemes molt atractius tècnicament i amb veritable sortida comercial.

7. Valoració econòmica

Qualsevol projecte s'ha de poder expressar en termes de costos, de forma que es pugui valorar la conveniència de la seva realització, i el resultat. Per un costat s'ha de valorar l'esforç en hores de disseny, desenvolupament i proves, i també el cost d'instal·lació, tant de formació com de lliurar un sistema 'clau en ma'. En aquest projecte s'intenta fer una valoració econòmica fent una divisió de tasques, com si fos un projecte desenvolupat per un equip. Així trobem la figura del cap de projecte, analistes, analista-programador, ... Tot i que a la realitat el projecte només l'ha desenvolupar una persona, amb les dificultats que això implica, sobre tot en la part de control del projecte i en l'assegurament de la qualitat.
En el nostre cas el cos de desplegament és negligible, ja que es basa en la instal·lació d'un PC amb sistema operatiu Ubuntu 10.04, i el necessari per fer funcionar el sistema TinyOS 2.1.1 i les seves aplicacions de suport. Totes elles gratuïtes. La instal·lació de les motes a lloc de desplegament és molt senzilla, i permet fer-ho només alimentant la que funciona remotament amb unes piles, i connectant la que fa d'enllaç al PC per un port USB. Els costos derivats de la instal·lació són molt baixos (desplaçar-se, desplegar motes remotes i instal·lar el programari).
La formació sobre el sistema és molt bàsica, ja que és força simple i amb unes breus explicacions sobre com instal·lar les motes i executar la consola d'usuari, crec que és més que suficient.

On s'apliquen els costos principals és en el desenvolupament pròpiament dit del sistema des de zero, que s'hauria de repercutir directament al client si aquest fos sobre una comanda única. També s'ha d'afegir el cost dels materials i adaptacions que s'han realitzat.

7.1 Cost desenvolupament del sistema

Les diferents activats de desenvolupament i el seu cost estan reflectits en la següent taula:

Codi de l'activitat	Nom de l'activitat	Estimació (hores)	Recurs	Cost
01	Inici del projecte	-	-	-
03.01	Estudi d'oportunitat	10	Cap de projecte	150€
03.02	Anàlisis	40	Analista	440€
03.03	Disseny	40	Analista programador	400€
03.04	Programació i proves unitàries	100	Analista programador	1.000€
03.05	Proves finals	20	Analista	220€
04	Formació d'usuaris	1	Analista	11€
06	Modificació plaques	5	Tècnic de sistemes electrònics	45€
07	Documentació	50	Analista	550€
08	Fi de projecte	-	-	
Total		**270**		**2.816€**

Les tarifes aplicades són les següents des del punt de vista d'empresa desenvolupadora i sense marge comercial:

Recurs	Cost/Hora
Cap de projecte	15 €
Analista	11 €
Analista programador	10 €
Tècnic de sistemes electrònics	9 €

7.2 Cost dels materials

Els materials que es descriuen a continuació conformen la instal·lació del sistema sencer a un PC, que tindrà un sistema d'alimentació ininterrompuda (SAI) de salvaguarda. També inclou les adaptacions de les motes per controlar accionaments i la integració d'humistors.

Sistema	Descripció	Preu unitari	Proveidor	Quantitat	Total
Mota	Mota COU_1_2	24€	UOC	2	48€
PC de control	Lenovo ThinkCentre A70 7099 - Core 2 Duo E7500 2.93 GHz	475,53€	PCGreen	1	475,53€
	APC Back-UPS RS 1200 LCD - UPS - 720 vatios - 1200 VA	318€	PCGreen	1	318€
Adaptació mota	Humistor HIH 5031	11€	Farnell	2	22€
	Resistència 100kOhms	1c	Farnell	2	2c
	Protoboard	1c	Farnell	2	2c
Placa de potència	BS170	0,39€	Farnell	4	1,56€
	1N4004	0,1€	Farnell	4	0,4€
	Relé RTD14012 - RELÉ, PCB, SPCO, 12VCC	3,52€	Farnell	4	14,08€
	Protoboard ROTH ELEKTRONIK RE210-S1	4,99€	Farnell	1	4,99€

Materials, en total **884,60€**

La valoració econòmica final del projecte seria:

Cost de desenvolupament: 2.816€

Cost dels materials: 884,60€

Total 3.700,60€

8. Conclusions

8.1 Llista d'objectius i estat final

Al pla de treball inicial es van exposar una sèrie de tasques i subtasques, que necessitaven fixar uns objectius que s'han de complir. Que es mostren a continuació:

1. *Node enllaç*
    c. *Pont entre aplicació de control i xarxa de nodes sensors*
        i. *Recepció de missatges des de la xarxa de sensors*
        ii. *Transmissió d'ordres des de l'aplicació de control cap a un node sensor concret*
    d. *Control de sistemes d'accionament mitjançant les seves sortides (veure punt 5)*
        i. *Sortida 1: ventiladors i la seva senyalització lluminosa*
        ii. *Sortida 2: polvorització d'aigua i la seva senyalització lluminosa*
        iii. *Sortida 3: obertura de finestres i la seva senyalització lluminosa*
        iv. *Sortida 4: alarma, i la seva senyalització lluminosa i acústica*

2. *Node sensor*
    a. *Lectura periòdica dels valors dels seus sensors*
    b. *Transmissió immediata dels valors dels seus sensors*
    c. *Emmagatzematge dels valors davant d'una caiguda de la xarxa de nodes*

3. *Node sensor de referència*
    a. *Lectura periòdica dels valors dels seus sensors*
    b. *Transmissió immediata dels valors dels seus sensors*
    c. *Emmagatzematge dels valors davant d'una caiguda de la xarxa de nodes*

4. *Aplicació de la consola de control*
    a. *Control de la xarxa de nodes*
        i. *Validació dels nodes actius actuals*
        ii. *Mostrar els nodes amb problemes d'alimentació elèctrica (baixa bateria)*
        iii. *Enviament de comandes de verificació cap a un node de la xarxa, o tots*
        iv. *Caiguda de la xarxa de nodes. Avís a la consola de control del sistema*
    b. *Configuració dels valors límits ambientals*
    c. *Control dels accionaments (veure punt 5)*
        i. *Mostrar l'estat actual dels accionaments*
        ii. *Enviament manual d'una ordre per accionar un o varis dels subsistemes de control ambiental (ventiladors, polvorització o obertura de finestres)*
    d. *Escriptura al fitxer de dietari de les dades ambientals rebudes pels nodes sensors*
    e. *Mostrar els valors de les dades dels nodes sensors a la consola de control*
    f. *Programar la reacció del sistema davant de les dades referents a:*
        i. *Valor límit temperatura*
        ii. *Valor límit humitat (veure punt 5)*
        iii. *Valor límit de tensió d'alimentació*

   g. *Escriptura d'un fitxer de dietari separat per recollir les decisions preses de forma automàtica i manual pel sistema (accionaments) i també de les caigudes de la comunicació*

5. *Estudis teòrics addicionals. La seva realització pràctica dependrà de la disponibilitat material*
   a. *Sensor d'humitat*
      i. *Model recomanat*
      ii. *Programació de la interfície*
      iii. *Implementació en un node sensor*
      iv. *Verificació del funcionament en el sistema*
   b. *Placa de control de potència*
      i. *Model recomanat*
      ii. *Programació de la interfície*
      iii. *Implementació en el node d'accionaments*

Tots aquest objectius s'han aconseguit, i per tant s'han finalitzat tots els objectius planificats per al projecte. Tot i que s'ha de fer una esmena respecte a la realització de la placa de control de potència. Per motius de temps, i materials, no s'ha pogut realitzar una placa electrònica i connectar-la a la mota *mB* com era desitjable. Aquest component només es simula i es deixa la seva realització pràctica seguint l'exemple que es presenta a l'annex.

8.2 Descartes fets al llarg del projecte

Al pla inicial del projecte es van prendre decisions de disseny que a posteriori s'han abandonat, bàsicament per qüestions de temps de realització. També algunes de les sugerències que el consultor m'ha fet arribar per afegir més qualitat. Són les següents:

- No es farà servir cap interfície gràfica. Es considera suficient la sortida per consola en format text de les informacions que es van produint

- Els nodes/motes no registraran les dades si no les poden transmetre per caiguda de la xarxa. El sistema està pensat per funcionar 'en temps real', si no funciona la xarxa això ja es un problema crític en si mateix. L'objectiu del sistema és tant mantenir unes condicions climàtiques com auditar les que s'estan produint en tot moment, si falla la comunicació és símptoma d'un error greu ja que no disposem les dades en el moment que s'estan produint

- La recerca a les dades emmagatzemades als fitxers de log queda fora de l'objectiu del projecte. La raó és que els 2 formats triats permeten el treball de cerca amb aplicacions ja disponibles. Pel fitxer de log serveix un editor de text qualsevol, i per l'estadístic qualsevol que permeti la lectura de fitxers .csv amb separador de dades per caràcter ';'.

- El consultor m'ha suggerit un disseny més modular de l'aplicació, de forma que es poden fer components per a cada sensor. La falta de temps m'ha impedit fer aquesta implementació de codi, però és una molt bona idea ja que permet independitzar el tipus de sensor, i el seu funcionament particular, de la lògica principal del node

- El consultor també m'ha suggerit que els llindars d'alarma també fossin controlats des del node. He descartat aquest funcionament per què volia estalviar el màxim d'energia. L'estratègia que he seguit:

- Els sensors són desactivats els sensors entre lectures, forçant al màxim l'estalvi
- Si el node ha de fer algun càlcul, això també resta energia. Per evitar-ho es podria emmagatzemar el valor de la variable en la seva codificació ADC, per exemple si el valor límit de temperatura és 2,71VDC i això equival a 900, només guardaríem 900 al node, i seria el valor de comparació directe amb el valor que torna l'ADC. El problema és que hauríem de tenir sempre activats els sensors per fer comparacions continues entre els valors límits i les lectures, i així avisar d'una situació límit, contradient el punt anterior
- La radio només transmet, en treball normal, quan finalitza el cicle de temps entre lectures. El seu valor es pot programar entre 5 i 7200 segons. La resta del temps està només escoltant
- El projecte està pensat per controlar el clima en una instal·lació relativament estàtica climàticament, com ho és un hivernacle. Pensem que el volum d'aire no sol variar dramàticament, i com a pràctica de disseny de la instal·lació hauríem de partir de les següents consideracions:
  - La situació dels nodes sensors, que haurien d'estar situats en llocs estratègics que recullin el millor possible la mitjana climàtica de l'espai a controlar (per exemple no seria una bona pràctica situar-los al costat de la porta d'entrada, o ventiladors)
  - El volum d'aire i el temps que triga per exemple en variar 1 ºC de temperatura. Si obtenim quan triga la variació climàtica del volum d'aire per fer el canvi +/-1ºC i +/- 1% d'humitat relativa, i l'apliquem al temps de cicle de lectura podrem tenir una salvaguarda suficient de la condició climàtica i a més estalviar energia. Per exemple si per les variacions anteriors tenim que es triga 120 segons, el temps de cicle de lectura es podria fixar per sota d'aquest valor i així estar segurs que detectarem la variació amb temps suficient (per exemple amb un cicle de lectura de 60 segons). D'aquesta forma no estarem lligats a cicles massa curts i mantindrem les condicions climàtiques estalviant energia de la mota/node

- El consultor també m'ha suggerit que es podia fer un desenvolupament de major estalvi si la ràdio només transmet amb la potència necessària per arribar al node d'enllaç. Aquest és un desenvolupament interessant, que es pot començar aplicant la mínima potència de transmissió i si no es té resposta, anar incrementant-la progressivament fins establir contacte. Per manca de temps he de descartar aquest desenvolupament

8.3 Proposta de millores

Un sistema comercial real hauria de considerar, apart dels descartes que ja s'han tractat i les millores que ha suggerit el consultor al punt anteriors, almenys les següents que afegirien molta més flexibilitat, seguretat i estalvi de recursos, a més de construir un sistema comercialment més potent:

- Utilització d'una base de dades relacional. L'objectiu és el de mantenir l'estructura de dades del sistema de forma coherent i segura al llarg del temps. Aquesta estructura no sols s'hauria de pensar per mantenir les dades estadístiques climàtiques generals i el registre de log, que és el que fa la nostra aplicació de forma 'plana', sinó que es podria utilitzar per:
  - Portar un control de quants nodes/motes de registre climàtic té el sistema, les seves dades personals (data adquisició, model, sensors que utilitza, temps de

cicle), incidències que s'hagin registrat en una d'elles, les que actualment estan en servei, ...
  - Poder establir zones de valors climàtics diferenciats, i les motes/nodes que intervenen, i registrar tots els valors d'aquestes zones de forma particular
  - Portar un registre d'alarmes activades, la seva duració, els valors climàtics que es van produir abans i durant la incidència
  - Fer càlculs estadístics directes en temps real, mostrar gràfiques de comportament
  - Poder imprimir informes sobre qualsevol consulta predefinida, o permetre la creació a mida
  - Permetre una estructura més segura d'emmagatzematge de dades, que a més pot formar part d'un sistema distribuït més gran
  - Registrar el tipus de cultiu que s'està produint per després poder establir comparacions amb d'altres campanyes. També el valor final de la collita (pes) i així comparar entre campanyes
  - ...

  Bàsicament és tot un desenvolupament a nivell de dades que permeti el registre de qualsevol valor del sistema, fixar paràmetres d'operació, la seva consulta, facilitar procediments de mineria sobre les dades, i garantir la seva seguretat

- L'escenari d'instal·lació, un hivernacle, permet utilitzar fonts d'energia addicionals a les piles, sobretot pel baix consum dels dispositius, i permetent que no es limiti la flexibilitat que disposem gràcies a les connexions inhalàmbriques. Per exemple amb una petita adaptació es podria alimentar el node/mota amb l'energia provinent d'una placa solar. A la nit una petita bateria a cada node/mota seria suficient per mantenir la seva activitat, i de nou al matí passaria a alimentar-se de la placa solar. Els canvis d'alimentació podrien ser gestionats per la pròpia mota gràcies al sensor de tensió de bateria. Aquest canvi permetria d'altres com variar el funcionament de la mota i establir llindars límit que forçarien missatges d'alarma abans del cicle de lectura programat, ja que sempre tindríem els sensors encesos al no dependre tant de l'estalvi energètic

- Afegir altres sensors interessants en el sector de producció agrícola. Per exemple varis sensors d'humitat del terra, a diferents profunditats, per saber el moment òptim de rec. O també analitzar la quantitat de CO_2 a l'aire de la instal·lació, per si s'ha d'elevar la concentració i així afavorir la fotosíntesi de les plantes

- El node/mota utilitzat no és viable per instal·lar-lo tal i com l'hem utilitzat. No és operatiu. Les condicions climàtiques a les que estarà sotmès fan necessari que sigui més robust i estanc. Si es volgués desenvolupar com a projecte comercial seria relativament fàcil redissenyar la placa electrònica i les connexions al nostre desig operatiu, i encapsular-lo tot en una caixa adient a les condicions climàtiques a suportar. La personalització d'aquest contenidor pot permetre l'adició també d'alimentació per placa solar, antenes per una transmissió/recepció ràdio més eficient, i la personalització comercial que desitgem entre d'altres

- Ampliar l'escenari de control d'accionaments a qualsevol mota remota, sempre que tinguéssim una font d'energia garantida que accionés la part de control. D'aquesta forma podríem fer petits clústers de control d'accionaments que dependrien directament del sensor remot més proper. Així s'actuaria de forma local dins de l'hivernacle. Per exemple si un node/mota instal·lat en una cantonada de l'hivernacle detecta un valor climàtic

anòmal, baixada de la humitat, i envia aquest valor al sistema de control centralitzat, aquest li enviarà la comanda d'engegar la polvorització d'aigua fins regular el valor. Aquest efecte seria local a la zona del sensor i la resta de l'hivernacle no tindria cap efecte directe, amb el consegüent estalvi de recursos

- Fer un control proporcional del temps d'accionament. El sistema ara activa una contramesura per variar un valor climàtic, i la desactiva quan es supera el valor de llindar. Un funcionament més correcte seria el de considerar el volum d'aire del lloc a controlar, el valor actual i futur del paràmetre a modificar, la resta de valors climàtics, i llavors calcular quan de temps s'ha d'actuar. Les lectures constants dels valors permeten anar recalculant el temps final, per ajustar-nos al màxim. D'aquesta forma evitem consums energètics innecessaris, i no correm el risc d'iniciar una carrera de controls per què hem desestabilitzat el sistema

- En el cas de pensar en controls remots, ja per sobre del control local d'un hivernacle, es pot establir almenys un nivell més de control distribuït de forma que des d'un únic control centralitzat de més nivell podem controlar tota una àrea d'instal·lacions separades, i aprofitant les TIC actuals podrien estar fins i tot molt separades geogràficament. Aquest sistema és perfectament escalable, i es poden mesclar estructures ambientals diferents: hivernacles, magatzems de queviures, granges, ... Ja que cada estructura tindria les seves particularitats ambientals, i la seva distribució de sensors i accionaments, i a més remotament es podrien variar molts dels valors d'interès (com el cicle de lectura de sensors o els valors límit ambientals)

Amb les millores plantejades poden tenir un escenari de control d'alta flexibilitat, molt adaptable a qualsevol estructura, no només a la seva instal·lació a un hivernacle, i per tant amb una possible sortida comercial més segura.

8.4 Autoavaluació

Una de les fases més importants, o potser la més important, d'iniciar un projecte qualsevol a la vida és la aprendre dels errors propis. Aquests poden ser tant comesos per nosaltres mateixos com de la resta de companys o contractistes que intervinguin. La premissa bàsica és que mai, sobretot si és la primera vegada que ens enfrontem a un projecte de certa entitat, tot sortirà segons el previst. L'habilitat que hem de desenvolupar és la de treure profit d'aquests errors, analitzant els seus motius i les solucions aplicades, i així anar augmentant la nostra experiència personal. Només d'aquesta forma podem evitar repetir alguns aspectes desagradables, i quan es produeix algun error tenir almenys algunes possibles alternatives. L'autoavaluació sobre el projecte he decidit dividir-la en 2 visions, una sobre una avaluació personal al treball desenvolupat, i una altra als problemes tècnics que m'han sortit i les solucions, si les he trobades, que s'han aplicat.

En el primer cas, l'autoavaluació sobre el treball desenvolupat, he de dir que tot i haver complit els objectius marcats a l'inici del projecte, trobo que hi han coses a millorar tot i haver aplicat l'experiència, que ja acumulo dels meus anys voltant per diverses empreses. En aquest cas no era conscient de la soledat que m'acompanyaria al llarg del projecte, tot i ser 4 companys a l'aula i l'ajuda del consultor. El projecte era una tasca a abordar, organitzar i realitzar per una sola persona. Cosa que dificulta la qualitat final, ja que no és recomanable fer de director, analista, programador i provador... A més aquesta "múltiple personalitat" tampoc permet seguir fil per randa una certa planificació. De forma sobtada trobava que un algorisme podia ser més eficient, i això m'ha fet anar endavant i enrere en algunes fases del projecte. Si hagués tingut una direcció

ferma això no hagués passat, ja que algunes decisions les hagués pres un tercer, amb un anàlisi més objectiu i pragmàtic segurament.

La falta d'experiència amb el sistema operatiu TinyOS i el llenguatge nesC també han fet que abordés algunes tasques potser amb massa optimisme. Després, a les primeres dificultats no trobava la suficient informació, o bé em costava d'assimilar. Amb els conseqüents nervis i preses. En un projecte amb poc mes de 3 mesos és difícil avançar uniformement des de 0. Potser al principi del projecte era el moment de no posar fites massa elevades i lliurar un sistema més compacte i senzill. De fet és la tècnica que sol realitzar-se: treure un producte que interessi, però amb camp per créixer si té èxit. Per tant una cosa que he fallat és la gestió del temps, al final he pogut enllestir els objectius, però a força de treure totes les hores que no dedicava a treballar o dormir, i amb la sensació de que eren millorables.

Com a conclusions importants d'aquesta autoavaluació personal puc treure que primer hem de fixar uns objectius realistes i assolibles a la nostra experiència personal sobre el sistema base que farem servir. En cas de falta d'experiència, tot i definir unes idees inicials i generals del que volem, el més recomanable és dedicar força temps al principi a l'estudi de materials i exemples, i consultar a fonts que tinguin experiència, que després aplicarem per fer una planificació que serà més abordable, i per tant amb més garanties d'èxit. En cas de no fer-ho segurament podrem lliurar un projecte que complirà amb la idea general de disseny, però que segurament tindrà ressentida la qualitat o el cost, o bé tots dos, fent que la seva evolució posterior i el seu manteniment siguin més difícils. Una situació per tant inacceptable.

A nivell tècnic les coses són una mica diferents. Consultant a diferents fonts, exemples i fòrums quasi tot té solució. Els problemes més greus que he trobat al desenvolupar són els següents:

- Error en configuració del sistema. Quan vaig instal·lar el sistema vaig utilitzar una còpia Ubuntu amb tot preinstal·lat, llavors vaig baixar-me del repositori una còpia del fitxer del Tinyos amb la cou24, però no vaig substituir-ho bé, i als valors d'entorn al fitxer tinyos.sh es mantenia un valor a un sistema que no existia. El resultat és que no podia comunicar-me amb les motes. Després de revisar tot el sistema vaig descobrir l'error:

  TOSROOT="/opt/tinyos-2.x"

  Quan hauria de posar:

  TOSROOT="/opt/tinyos-2.1.1"

  Vaig canviar-ho, recompilar-ho tot i va funcionar perfectament

- No recepció de missatges. El node remot enviava missatges que eren recollits pel node enllaç, i transmesos correctament al programari Java. El problema el tenia quan volia fer el pas invers, ja que mai m'arribaven els missatges al node remot. Vaig llençar la pregunta al fòrum i un company em va contestar si havia revisat la funció Java per enviar missatges, ja que per defecte l'enviava al node amb TOS_NODE_ID amb valor 10, i el meu era 1. Efectivament el problema era aquest

- *Broadcast*. Quan volia enviar un missatge a totes els nodes de la xarxa, aquest no arribava tot i fer servir el valor de broadcast (65535) al identificador de mote/node on volia enviar-ho. El problema és que feia servir un tipus de dada per recollir l'identificador al missatge, que no era l'adient: feia servir uint8_t quan havia de ser uint16_t. Vaig llençar la pregunta al fòrum, però tot i que el consultor em va dir la resposta correcta, ja l'havia trobat fent proves de formats

- La compilació del programari de les motes donava un *warning* quan volia assignar un valor superior a 255 a una variable uint16_t. Tot i que 16 bits són suficients per emmagatzemar el valor. El problema és que valors superiors a 255 s'ha d'indicar explícitament afegint L a la dada, ja que sinó només es fa servir 1 byte. Per exemple
  uint16_t test = 65535; -> warning
  uint16_t test = 65535L; -> solucionat

El dubte el vaig escriure al fòrum i vaig tenir resposta del consultor

- Entendre el concepte de *wiring*. Per solucionar-ho he consultat els manuals disponibles al web de *www.tinyos.net*, el llibre de *Tinyos Programming* i comparar codi dels exemples disponibles a l'entorn *tinyos 2.1.1* instal·lat. De moment he aconseguit el nivell suficient per programar de forma bàsica els components necessaris

- Executar la generació de classes Java amb l'eina *mig*. No trobava informació massa aclaridora fins que vaig trobar un exemple als *mailing list* de la web *www.tinyos.net*. El format que he fet servir, i no m'ha donat problemes és el següent:
  - mig java
    - –target=cou24
    - –java-clasname=<classe que generem>
    - <fitxer de capçalera amb els structs>
    - <struct d'interès>
    - -o <fitxer de sortida>

  Un exemple amb el meu struct MoteMsg i fitxer de capçalera AuxVar.h:
  mig java –target=cou24 –java-classname=MoteMsg AuxVar.h MoteMsg –o MoteMsg.java

- No he pogut resoldre el *warning* indicat a la part de *bugs*, tot i consultar molta literatura sobre el problema de tenir massa connexions en una interfície. No he trobat cap exemple clar. He provat de desactivar totes les connexions a la interfície *Receive*, i quan ho feia en només un component, per exemple *BaseStation*, aquest funcionava correctament. Però tot i fer-ho en components separats si per exemple a *SensorBP* activava només una connexió a *Receive* en tornava a sortir al compilar. No he trobat cap solució i curiosament només trobo el problema al compartir una connexió sèrie, la part ràdio no dóna cap *warning*

- L'enviament segur de *packets* per ràdio l'he pogut millorar afegint *PacketAcknowledgements*, i fora d'alguna col·lisió ara he pogut observar que no es perden gairebé mai els paquets de per exemple canvi de temps de cicle de lectura de sensors

- Tot i provar de capturar les excepcions en Java respecte a la caiguda o no activació del *SerialForwarder* al final no he pogut trobar la solució, provant totes les excepcions que conec. Al final si activem la consola de gravació de dades o volem comunicar amb una mota, i aquest programa no fa el pont entre les aplicacions, el programa finalitza sense el control que m'hauria agradat implementar

- Vaig tenir sort en trobar ràpidament un component força econòmic per afegir la mesura d'humitat relativa, i que a més és fàcil d'interpretar i implementar ja que segueix la lògica dels components que ja porta el node preinstal·lats. Després de llegir força sobre sensors d'humitat vaig trobar que el **Honeywell HIH-5031** era el que millor s'adaptava, per

funcionar a una tensió de fins 2.7Vdc, i a més a meitat de preu que d'altres (per exemple el **Sensirion SHT7x**)

- El desenvolupament de la placa de control de potència, tot i que volia enllestir-la, només la podré presentar teòricament a la memòria. La falta de temps no em permet anar més enllà, i el disseny tot i electrònicament no és difícil, un MOSFET polaritzat que controli un relé, no puc acabar la instal·lació a las sortides del microcontrolador per què tampoc no disposo de les eines adequades i no vull fer malbé l'electrònica de la placa (el meu soldador no és per l'ús amb tecnologia SMD...)

- A les motes/nodes canviava per codi el seu valor *TOS_NODE_ID*, ja que tot i fer *make cou24 install.0* no quedava fixat el valor d'Id del node a 0. El problema es trobava en que carregava el fitxer incorrecte *main.srec* enlloc de *main.srec-out0*, una vegada detectat vaig poder eliminar el codi i fer-ho en temps de compilació

- El consultor em va comunicar que no feia un control del port sèrie quan l'activava, de forma que si un altre component en feia ús no el podria fer servir. Vaig veure que existeixen diferents missatges que torna el sistema davant de l'activació i si aquesta no era completa (SUCCESS) llavors el recomanable era activar de nou l'activació fent servir una tasca

- Tot i comprendre el mecanisme de *watchdog* no vaig entendre exactament com trencar la barrera de manipular directament els *flags* i registres del microcontrolador, i comunicar-ho als components. El dia 29 de maig he vist que al fòrum el consultor ens indica una forma de fer-ho, però ja no disposo de temps material per a més proves, així que no el puc incorporar al projecte

9. Glossari

ADC (*Analog-to-Digital Converter)*
En català Convertidor Analògic Digital (CAD). Consisteix en la transformació de senyals analògiques en senyals digitals per facilitar el seu processament, i fer el senyal menys sensible al soroll i altres interferències

API (*Application Programming Interface*)
Conjunt de regles i especificacions que els programes han de seguir per comunicar amb un altre, serveix d'interfície entre programes per així facilitar la seva interacció

Buffer
Referit a la memòria utilitzada com a magatzem temporal de dades d'entrada o sortida

Cicle de vida en cascada
Metodologia de desenvolupament de programari en la que es defineixen unes etapes ordenades i tancades, de forma que per iniciar una nova etapa ha d'estar finalitzada completament l'anterior

Diagrama de Gantt
Eina gràfica que permet mostra el temps de dedicació previs per a diferents tasques o activitats al llarg d'un temps total de terminat

Sensor d'efecte Hall
Sensor que s'utilitza per mesurar camps magnètics, corrents o determinar posicions gràcies a l'aplicació de l'efecte Hall (afectació de camps magnètics perpendiculars)

I^2C (*Inter-Integrated Circuit*)
Bus de comunicacions dissenyat per Phillips, amb velocitat de 100 Kbits/s a 3,4 Mbit/s, utilitzat sobre tot per comunicar microcontroladors i perifèrics. Utilitza 2 línies: una pel rellotge i l'altra per les dades.

SPI (*Serial Peripheral Interface*
Estandard de comunicacions utilitzat per transferència d'informació entre circuits integrats. funciona amb transferència d'informació controlada per rellotge. Té 4 línies de transmissió: data in (MISO), data out (MOSI), rellotge (SCLK) i selecció del xip (SS)

LED (Ligth Emitter Diode)
Al polaritzar un diode semiconductor convencional de forma directa, es produeixen fotons. Un diode LED està dissenyat per aprofitar al màxim aquest efecte lumínic, i utilitzar-se per petites il·luminacions, com les necessàries en instrumental

MAC (*Media Access Control*)
Identificador únic d'un dispositiu en xarxa, que correspon a la seva adreça física, i que en principi és única a nivell mundial

Memòria EEPROM (*Electrically-Erasable Programmable Read-Only Memory)*
Memòria de només lectura que pot ser esborrada elèctricament i tornada a programar

Microcontrolador
A diferència d'un microprocessador, un microcontrolador integra les tres unitats funcionals d'una computadora:UCP, memòria i perifèrics

Protoboard
Placa de muntatge de circuits electrònics, que facilita passar del disseny teòric al circuit, ja que disposa de pistes conductores i/o espais on fixar els components, i així muntar un primer prototip viable

Socket
Abstracció de programari que permet la comunicació entre processos

TIC (*Tecnologies de la Informació i la Comunicació*)
Conjunt d'elements i tècniques utilitzades pel tractament i transmissió d'informació

Transceptor
Dispositiu que fa funcions tant de transmissió com de recepció d'informació, aprofitant per fer-ho de circuits comuns als dos processos

TTL (*Transistor Transistor Logic*)
Família lògica o tecnologia de construcció de circuits electrònics digitals on els seus components d'entrada i sortida són transistors bipolars

USART (*Universal Asynchronous Receiver-Transmitter*)
Xip que controla els ports i dispositius sèrie. Les seves funcions principals són el control de les interrupcions dels dispositius connectats al port sèrie i convertir les dades al format paral·lel, del bus de sistema, a dades en format sèrie per així ser transmesos pel port sèrie i viceversa

10. Bibliografia

1. Web ***http://www.tinyos.net***
Accés a la documentació en línia (principalment manuals i el *search engine*).

Principals TEP que he consultat:
TEP101: *Analog-to-Digital Converters (ADCs)*
TEP102: *Timers*
TEP106: *Schedulers and Tasks*
TEP107: *TinyOS 2.x Boot Sequence*
TEP109: *Sensors and Sensor Boards*
TEP111: *message_t*
TEP112: *Microcontroller Power Management*
TEP113: *Serial Communication*
TEP114: *SIDs: Source and Sink Independent Drivers*
TEP116: *Packet Protocols*
TEP117: *Low-Level I/O*
TEP118: *Dissemination of Small Values*
TEP126: *CC2420 Radio Stack*

2. Web ***http://cv.uoc.edu/app/mediawiki14/wiki/Inici24***
Recursos de programari disponibles a la pàgina:

ExampleMoteProgramForCOU.tar.gz (BlinkCOU)
Per l'estudi de les lectures a sensors dintre d'un node, i la composició d'un missatge d'enviament via ràdio, així com la recepció de missatges i el seu tractament.

COUTester.zip
Per l'estudi de recepció i enviament de missatges entre un programari JAVA instal·lat en un PC i la xarxa de nodes.

3. Recursos de programari disponibles al sistema **Tinyos 2.1.1**

BaseStation
Programa que permet l'enviament de missatges rebuts de forma sèrie, a la resta de nodes utilitzant la ràdio. I també enviar al PC els missatges que rep per la ràdio des de la resta de nodes.
Autors: Phil Buonadonna, Gilman Tolle, David Gay i Philip Levis

SerialForwarder
Programa que permet obrir un canal sèrie entre un node connectat a un port USB i el programari JAVA.

4. Llibre ***TinyOS programming***
Autors: Philip Levis i David Gay
Editorial: Cambridge University Press
ISBN: 978-0-521-89606-1

5. Llibre **Principios de electrónica** (capítol 13, *El transistor MOSFET*)
Autor: Albert Paul Malvino
Editorial: McGRAW-Hill
ISBN: 968-451-721-1

6. Sensors d'humitat
Sensirion SH71
http://www.sensirion.com/en/01_humidity_sensors/05_humidity_sensor_sht71.htm
http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT7x.pdf

HoneyWell HIH-5031
http://sensing.honeywell.com/index.cfm?ci_id=140301&la_id=1&pr_id=156098
http://sensing.honeywell.com/index.cfm/ci_id/155943/la_id/1/document/1/re_id/0
http://sensing.honeywell.com/index.cfm/ci_id/147516/la_id/1/document/1/re_id/0
http://sensing.honeywell.com/index.cfm/ci_id/156871/la_id/1/document/1/re_id/0

7. Altres sensors del sistema
- Temperatura: **MCP9700** de **Microchip**
  http://ww1.microchip.com/downloads/en/DeviceDoc/21942e.pdf
- Lluminositat: **PDV-P9003-1** de **Advanced Photonix**
  http://www.advancedphotonix.com/ap_products/pdfs/PDV-P9003-1.pdf
- Hall: **BU52001GUL** de **Rohm Semiconductor**
  http://www.rohm.com/products/databook/sensor/pdf/bu52001gul-e.pdf

8. Mòdul ATMEL Zigbit 2,4 GHz **ATZB-24-A2**
http://www.atmel.com/dyn/resources/prod_documents/doc8226.pdf

9. Microcontrolador **ATmega1281**
http://www.atmel.com/dyn/products/product_docs.asp?category_id=163&family_id=607&subfamily_id=760&part_id=3630

10. Transceptor **AT86RF230** 2,4 GHz Zigbee
http://www.atmel.com/dyn/resources/prod_documents/doc5131.pdf

11. Empresa subministradora de components electrònics
http://es.farnell.com

12. Portal informatiu d'aplicacions **ZigBee**
http://www.zigbee.org/Home.aspx

13. Casos d'èxit amb xarxes WSN aplicats a centres de producció agrícola a Galicia
http://www.itg.es/ITG/CasosExito_AgriculturaPrecision%20_RedesWSN.pdf

14. **Projecte SWAP** (*Sistema de telemetría por redes WSN para Agricultura de Precisión*)
http://www.swapproject.org/

15. Redes de sensores inalámbricos. Nuevas soluciones de interconexión para la automatización industrial
Autors: Niels Aakvaag, Jan-Erik Frey
http://library.abb.com/GLOBAL/SCOT/scot271.nsf/VerityDisplay/A019E9833DCF2819C1257199004E5DD2/$File/39-42%202M631_SPA72dpi.pdf

16. Redes **WSN**
http://zone.ni.com/devzone/cda/tut/p/id/9028

17. Mòdul UOC ***Gestió i desenvolupament de projectes***
Autor: Alfons Bataller Díaz
Referència: P08/19018/00444

18. Mòdul UOC ***UML (II): el model dinàmic i d'implementació***
Assignatura: Enginyeria del programari
Autor: Benet Campderrich Falgueres; Recerca Informàtica S.L.
Referència: XP03/05060/02078

19. Wikipèdia
http://ca.wikipedia.org/wiki/Portada
http://es.wikipedia.org/wiki/Wikipedia:Portada

Annexos

Annex A. Manual d'usuari

L'aplicació està dissenyada per fer-se servir en sistema operatiu Ubuntu 10.04 amb instal·lació correcta d'entorn Java versió1.6. A més ha de tenir instal·lat correctament el sistema Tinyos 2.1.1 amb la versió de mota/node cou24, i el classpath ha de recollir correctament on es troben les llibreries. Per carregar els programes als nodes/motes és necessari el programari meshprog, versió 0.1.2, de Meshnetics.

L'aplicació està dividida en diferents programaris:

- Un programari per la mota/node remot, escrit en nesC. Aquest permet el següent:
  - Té un inici amb codi de LEDs per informar que s'està iniciant correctament. El codi és:
    - LED verd ->LED verd i LED groc -> LED verd i LED groc i LED vermell
    - LED verd i LED vermell (inici sensor HALL)
    - LED verd (tot iniciat)
  - Captura i envia els valors dels sensors de temperatura, llum, bateria, temps de cicle i també el número de seqüència actual. Envia aquestes dades de forma periòdica. El cicle està definit per defecte a 30 segons en el codi de la mote/node. L'enviament es pot veure per l'activitat del LED groc. Aquest enviament es fa de forma ràdio
  - Captura i envia la detecció d'activitat en el sensor HALL que integra. Aquest enviament és totalment asíncron i la dada que s'envia és TOS_NODE_ID. L'enviament es pot veure per l'activitat del LED groc. Aquest enviament es fa de forma ràdio
  - Captura i envia la detecció d'activitat en botó d'usuari que integra. Aquest enviament és totalment asíncron i les dades que s'envien és TOS_NODE_ID i la seva MAC. L'enviament es pot veure per l'activitat del LED groc. Aquest enviament es fa de forma ràdio

- Un programari per la mota/node enllaç, escrit en nesC. Aquest permet el següent:
  - A l'iniciar-se correctament mostra el LED verd encès, el LED groc serveix per avisar que s'envien paquets via ràdio i sèrie
  - Captura i envia els valors dels sensors de temperatura, llum, bateria, temps de cicle i també el número de seqüència actual, ja que aquest node fa de sensor de referència fora de l'espai sota control climàtic. Envia aquestes dades de forma periòdica. El cicle està definit per defecte a 30 segons en el codi de la mote/node. L'enviament es pot veure per l'activitat del LED groc. Aquest enviament es fa de forma sèrie
  - Comunicació sèrie i ràdio per permetre l'enllaç entre la xarxa de nodes i l'aplicació de control instal·lada al PC
  - Activació i desactivació d'accionaments. De forma simulada s'ha establert que si el LED 0 (vermell) està encès indica que s'ha activat un accionament, altrament que estan apagats o el canvi d'encès a apagat indica que en aquell moment s'ha desactivat un accionament

- Un programari, realitzat en Java, que integra tota la 'intel·ligència' del sistema i fa de registre de les condicions climàtiques i pren decisions per mantenir el clima als valors desitjats. Les funcions que fa resumidament, ja explicitades en un altre apartat, són:
  - Permetre configurar el sistema als paràmetre climàtics i d'altres
  - Enviar missatges a les motes/nodes
  - Accionar manualment mitjançant la consola del sistema (línia de comandes) accionaments de potència (de forma simulada en aquest desenvolupament)
  - Registrar les dades de telemetria en un fitxer per a estudis estadístics
  - Prendre decisions sobre les dades rebudes i mostrar alarmes i forçar accionaments si és necessari per mantenir-les

- Un programari que permeti la comunicació entre el node enllaç, instal·lat en un port USB, i l'aplicació Java desenvolupada. Aquest programari s'utilitza tal i com és, i es tracta del *SerialForwarder*

L'aplicació s'executa des de la línia de comandes del sistema operatiu Ubuntu. Per fer-ho es recomana primer llegir la part de l'annex que tracta de com compilar i executar el sistema abans d'abordar els següents passos.

Una vegada tot ja instal·lat i compilat correctament, iniciarem una sessió de consola d'usuari. Si introduïm la instrucció

java files.ScafiaCon

Obtindrem una petita ajuda on s'especifica el sentit de cada opció configurable al sistema:

```
SCAFIA CON 30-05-2011 15:23:02 Inici de sessió de consola de control del sistema. Paràmetres introduïts:
SCAFIA - Error en la introducció de comanda en línia

# Format de comanda:    java scafia [opció 1] [opció 2] ...

# Opcions de control:
                [-info] [-iniciLog] [-h [iD]] [-obrir on/off] [-vent on/off] [-valve on/off] [-alarm on/off]

# Opcions de configuració del sistema:
                [-t <segons>] [-tMax <temp.>] [-tMin <temp.>] [-hMax <humitat>] [-hMin <humitat>]
                [-tMaxL <temp.>] [-tMinL <temp.>] [-hMaxL <humitat>] [-hMinL <humitat>]
                [-batt <V>] [-llum <nivell>] [-folderLog <nom>] [-folderStats <nom>] [-source <sf@IP:port>]

      Atenció!
      Les opcions de configuració del sistema fan necessari el reinici de la consola de missatges per carregar els nous valors.

# On:
      [-info] -> Mostra els valors actuals del fitxer de configuració del sistema. Es mostren més valors dels que es poden
            modificar amb la línia de comandes que es presenta, i que corresponen a configuracions de seguretat o inicials del sistema.

      [-iniciLog] -> Mostra en consola els valors actuals de lectures als sensors del nodes, alarmes i altres informacions del sistema.

      [-h [iD]] -> Envia un missatge al node especificat per comprovar si està en línia.
            [iD] ha de contenir el node d'interès (de 0 a 65534), si el seu valor es deixa en blanc s'envia a tots els nodes de la xarxa.

      [-obrir on/off]] -> Envia un missatge al node de control d'accionaments per aplicar l'acció descrita sobre les finestres.
            on: obre les finestres       off: tanca les finestres.

      [-vent on/off]] -> Envia un missatge al node de control d'accionaments per aplicar l'acció demanada sobre els electroventiladors.
            on: activa els electroventiladors      off: desactiva els electroventiladors.

      [-valve on/off]] -> Envia un missatge al node de control d'accionaments per aplicar l'acció demanada sobre les electrovàlvules de pas d'aigua
             al sistema de polvorització.
            on: activa les electrovàlvules  off: desactiva les electrovàlvules.

      [-alarm on/off]] -> Envia un missatge al node de control d'accionaments per aplicar l'acció demanada sobre els sistema d'alarma del sistema.
            on: activa l'alarma   off: desactiva l'alarma.

# Utilització de les opcions de configuració del sistema
```

```
            El següents valors es poden canviar en qualsevol moment, i es faran efectius immediatament, si es fa un reset de la consola de missatges o bé en la següent
            recepció d'un missatge de telemetria.
            Es recomana fixar-los abans d'iniciar l'operació del sistema, per exemple afegint després l'opció -iniciLog al final de la línia de comandes, si es volen
            canviar i iniciar de nou la gravació de valors.

      [-t <segons>] -> Canvia la periodicitat, de lectures de sensors, als segons especificats.
            Per disseny el valor de lectura per defecte està fixat a 30 segons. El màxim són 7200 segons (2 hores) i el mínim 5 segons.

      [-tMax <temp.>] -> Fixa el valor màxim en ºC de la temperatura operativa del sistema.
            El seu valor dependrà del límit fixat al sistema (valor tMaxL).

      [-tMin <temp.>] -> Fixa el valor mínim en ºC de la temperatura operativa del sistema.
            El seu valor dependrà del límit fixat al sistema (valor tMinL).

      [-hMax <humitat>] -> Fixa el valor màxim en % de l'humitat relativa operativa del sistema.
            El seu valor dependrà del límit fixat al sistema (valor hMaxL).

      [-hMin <humitat>] -> Fixa el valor mínim en % de l'humitat relativa operativa del sistema.
            El seu valor dependrà del límit fixat al sistema (valor hMinL).

      [-tMaxL <temp.>] -> Fixa el valor màxim en ºC de la temperatura del sistema.
            El seu valor ha de ser el més gran dels valors de temperatura fixats al sistema.

      [-tMinL <temp.>] -> Fixa el valor mínim en ºC de la temperatura del sistema.
            El seu valor ha de ser el més petit dels valors de temperatura fixats al sistema.

      [-hMaxL <humitat>] -> Fixa el valor màxim en % de l'humitat relativa del sistema.
            El seu valor ha de ser el més gran dels valors d'humitat fixats al sistema.

      [-hMinL <humitat>] -> Fixa el valor mínim en % de l'humitat relativa del sistema.
            El seu valor ha de ser el més petit dels valors d'humitat fixats al sistema.

      [-batt <V>] -> Fixa el nivell de tensió d'alarma als nodes sensors. Per defecte el seu valor és 2.7V
            El seu valor ha d'estar entre 2.7 i 3.

      [-llum <V>] -> Fixa el nivell de llum que indica si és dia o nit. Per defecte el seu valor és 50.
            El seu valor ha d'estar entre 0 i 200

      [-folderLog <nom>] -> Fixa el nom de la carpeta on s'emmagatzemen els fitxers diari de sistema (log).
            El seu valor només accepta caràters alfabètics del sistema anglès.

      [-folderStats <nom>] -> Fixa el nom de la carpeta on s'emmagatzemen els fitxers per a estadístiques del sistema.
            El seu valor només accepta caràters alfabètics del sistema anglès.

      [-source <sf@IP:port>] -> Fixa la ruta de comunicació entre el sistema SCAFIA i la xarxa de nodes.
            El seu valor ha de seguir el format sf@IP:port. Per defecte el valor és sf@LocalHost:9002. Els ports vàlids són entre 1024 i 65535

# Exemples:
      java scafia -h 1      Envia un missatge de comprovació al node remot 1.
      java scafia -h        Envia un missatge de comprovació a tots els nodes de la xarxa.
      java scafia -vent on Activa els electroventiladors de la instal·lació.
      java scafia -t 30     Modifica el cicle de lectures a 30 segons.
      java scafia -tMax 45      Modifica la temperatura màxima operativa a 45ºC.

# Fi de l'ajuda de consola SCAFIA
SCAFIA CON 30-05-2011 15:23:02 Finalitza sessió de consola de control del sistema.
```

De les opcions anteriors hi han els següents tipus:

- De control
  - –info: mostra els valors operatius configurats al sistema
  - –iniciLog: inici del sistema com a registrador climàtic
  - -h: enviar un missatge a un node concret per comprovar si està actiu
  - -obrir: activar/desactivar l'accionament que controla finestres
  - -vent: activar/desactivar l'accionament que controla els ventiladors
  - –valve: activar/desactivar l'accionament que controla les electrovàlvules de polvorització d'aigua
  - –alarm: activar/desactivar l'alarma general del sistema
- De configuració del sistema
  - Valors climàtics
    - tMax, tMin, hMax i hMin
  - Valors climàtics límit
    - tMaxL, tMinL, hMaxL i hMinL

- Altres valors
  - batt
  - folderLog
  - folderStats
  - source

Per veure els valors operatius del sistema, que actualment estan configurats, farem servir la instrucció

java files.ScafiaCon –info

SCAFIA CON 30-05-2011 15:27:03 S'ha demanat l'estat actual de les variables de configuració del sistema (-info)
==========
SCAFIA - Valors de configuració actuals
==========
30-05-2011 15:27:03

[Valors operatius]
Temperatura màxima (tMax): 31ºC
Temperatura mínima (tMin): 25ºC
Humitat màxima (hMax): 60%
Humitat mínima (hMin): 25%
Temps de lectura de sensors (t): 5 segons

[Valors límit del sistema]
Temperatura màxima (tMaxL): 32ºC
Temperatura mínima (tMinL): 10ºC
Humitat màxima (hMaxL): 60%
Humitat mínima (hMinL): 10%
Lluminositat canvi nit/dia (llum): 50
Bateria límit en node (batt): 2.71Vcc

[Altres valors del sistema]
Carpeta de logs del sistema: Logs
Carpeta d'estadístiques: Stats
Ruta de comunicació sistema-xarxa de nodes: sf@localhost:9002

SCAFIA CON 30-05-2011 15:27:03 Finalitza sessió de consola de control

Els paràmetres anteriors defineixen entre d'altres uns llindars operatius i uns llindars límits. La diferència entre ells és que els primers serveixen per fixar entre que valors s'han de mantenir automàticament el sistema, i els segons marquen una anomalia que posa al sistema en una situació indesitjable, activant una alarma del sistema, i que s'haurà de corregir de forma manual.

Valors operatius climàtics

El valors operatius de temperatura estan fixats per tMax i hMin
El valors operatius d'humitat estan fixats per hMax i hMin

Valors límit climàtics

Els valors límit de temperatura estan fixats per tMaxL i tMinL
Els valors límit d'humitat estan fixats per hMaxL i hMinL

El valor operatiu (t) controla cada quant es pren una mostra climàtica del sistema. Estant limitat entre 5 segons i 2 hores. El valor límit de bateria al node està fixat per (batt), quan es mesura un valor inferior es mostra una alarma per forçar el canvi de piles al node corresponent.

Un altre valor límit, tot i que en el sistema no té cap implementació pràctica, és el relatiu al canvi dia/nit. No té cap conseqüència ni activa cap alarma.

Els paràmetres de control sobre accionaments o l'estat d'un node es poden executar en qualsevol moment, per exemple des d'una altra consola. Les accions sempre quedaran registrades al fitxer de log corresponent. El mateix passa amb la resta de paràmetres, no és necessari reiniciar el sistema per què els nous paràmetres s'executin. Començaran a tenir-se en compte en la següent lectura de telemetria que rebi el sistema central.

Una vegada modificats al nostre desig els paràmetres anteriors, sobretot els relatius a les condicions ambientals, podem iniciar la tasca del sistema com a auditor o registre de condicions ambientals. Per fer-ho s'ha d'introduir la següent instrucció com a mínim:

java files.ScafiaCon –iniciLog

A partir d'aquest moment es mostraran els missatges corresponents a valors de telemetria, alarmes i accionaments per corregir les situacions climàtiques.

Els missatges sortiran per pantalla, i es gravaran al fitxer de dietari (log) són de diferents tipus. De forma general hi ha un cert esquema:

- Indicar la data i hora que es genera o es rep
- Poden portar algun dels següents indicadors:
    - **A!**: situació d'alarma activa al sistema
    - **-** : indica normalitat, cap alarma al sistema o la solució d'una d'elles
    - **ACC**: activació/desactivació d'algun accionament
    - **ALARM**: detall d'alarma
    - **CON**: missatge des de consola d'usuari
    - **USR**: pulsació del botó d'usuari d'un node remot de la xarxa
    - **HAL**: detecció de l'activació del sensor Hall del node remot de xarxa
    - **ON!**: un nou node s'activa a la xarxa
    - **TEL**: conté dades de telemetria
    - **TIM**: canvi de temps de cicle de lectura de sensors de la mota
    - **NET**: referit a la comunicació amb xarxa de motes (caiguda o recuperació)
- Poden ser:
    - D'entrada al sistema des de la xarxa de nodes (=>|)
    - De sortida del sistema cap a la xarxa de nodes (|=>)
- Finalment porten una descripció més o menys extensa que permet entendre el missatge

A continuació es mostren diferents tipus de missatge que podem trobar al fitxer de dietari (log) del sistema. Aquests es gravaran també en un fitxer de format text amb nom scafia_yyyymmdd.log (on yyyy indica l'any, mm el mes i dd el dia), dins de la carpeta definida a *–folderLog*.

Missatges per manipulació de la consola de comandes (indicador **CON)**

Missatges d'inici i fi de sessió de consola:
SCAFIA CON 23-05-2011 18:23:24 Inici de sessió de consola de control del sistema. Paràmetres introduïts: -tMin 25
SCAFIA CON 23-05-2011 21:39:00 Finalitza sessió de consola de control del sistema.

Al introduir valors de configuració per consola, o missatges a causa de la manipulació de la consola:
SCAFIA CON 23-05-2011 18:23:24 Canvi de valor a variable de control de temperatura mínima (tMin), valor = 25
SCAFIA CON 23-05-2011 18:23:20 Error: el valor de temperatura (-temp) no és vàlid. Valor introduït (20)

Al enviar un missatge manual a un sensor remot per veure si està actiu:
SCAFIA CON |=> 24/04/2011 09:00 [1] Petició d'estat a sensor remot amb identificació 1
SCAFIA - ON! =>| 23-05-2011 17:45:27 [0] Node 1 està operatiu. Temps de cicle de lectura de sensors configurat = 10 segons

Missatges del sistema a causa de la recepció de telemetria i la seva anàlisi
Telemetria normal des de un node:
SCAFIA – TEL =>| 24/04/2011 09:00 [1] Valors Node Remot 1
Bateria(V): [2.8] Lluminositat(Lux): [100] Temperatura(ºC):[21.5] Humitat (%):[34.07]

Telemetria amb alarma:
SCAFIA A! TEL =>| 23-05-2011 19:25:14 [1115] Valors Node Ref. 0
Bateria(V):[2.96] Lluminositat(Lux):[18.07] Temperatura(ºC):[29.50] Humitat(%):[34.07]

Activació d'un node. rebem el seu identificador i el temps de lectura de sensors que té programat:
SCAFIA – ON! =>| 24/04/2011 09:00 [1] Node 1 està operatiu. Temps de cicle de lectura de sensors configurat = 15 segons

Missatge que es rep quan es pulsa el botó d'usuari en un node remot, s'informa de la seva MAC:
SCAFIA - USR =>| 23-05-2011 19:22:07 El node remot 1 està operatiu i té la MAC 77:69:83:77:66:0:207:231

Missatge que es rep quan el sensor Hall d'un node remot detecta activitat:
SCAFIA – HAL =>| 24/04/2011 09:00 [1] El node 1 detecta activació de dispositiu extern.

Quan el sistema detecta que un node té un temps de cicle de lectura de sensors diferent del programat al sistema, s'envia cap al node un missatge de reprogramació, i es mostra de la següent forma:
SCAFIA – TIM |=> 24/04/2011 09:00 [1] Canvi automàtic de temps de cicle de lectura de sensors a node [1]. Valor de cicle que s'envia = 10 segons

Quan el node remot reprograma el seu temps de lectura de sensors, llavors es mostra el següent missatge:
SCAFIA – TIM =>| 24/04/2011 09:00 [1] El node 1 informa d'un temps de cicle de lectura de sensors de 30 segons

Alarmes (ALARM)
Telemetria des d'un node i amb alarma activada:
SCAFIA A! TEL =>| 24/04/2011 09:00 [1] Valors Node Remot 1 Bateria(V): [2.8] Lluminositat(Lux): [100] Temperatura(ºC): [21.5] Humitat (%): [32.56]

La detecció pel sistema d'un baix nivell de tensió, en la bateria d'un node, es mostra de la següent forma:
SCAFIA A! ALARM 24/04/2011 09:00 La bateria del node 1 té un valor de tensió massa baix: 2.5 V [límit 2.6 V]

Recuperació del valor correcte de tensió d'alimentació en un node:
SCAFIA - ALARM 24/04/2011 09:00 La bateria del node 1 ha recuperat un valor de tensió correcte: 2.6 V

Recuperació del valor correcte de la temperatura al sistema:
SCAFIA – ALARM 24/04/2011 09:00 La temperatura, segons el node 1, ha recuperat els valors operatius: 22ºC

Caiguda de la xarxa de nodes
SCAFIA A! NET 23-05-2011 19:23:06 La xarxa de nodes està caiguda, o els nodes tenen un temps de lectura de sensors incorrecte.

Recuperació de la xarxa de nodes
SCAFIA - NET 23-05-2011 19:23:46 La xarxa de nodes s'ha recuperat, o els nodes tenen un temps de lectura de sensors correcte.

Activació d'accionaments (ACC)
Activació d'un accionament per corregir un valor climàtic fora de rang:
SCAFIA A! ACC 23-05-2011 19:23:06 S'activa el sistema d'alarma de la instal·lació.
SCAFIA A! ACC |=> 23-05-2011 19:23:06 [41] S'envia al node 0 que les alarmes generals de sistema s'han d'activar.
SCAFIA A! ACC =>| 23-05-2011 19:23:06 [41] El node 0 informa que les alarmes generals de sistema estan activats.

Fitxer estadístic

Per últim indicar que les dades de telemetria es gravaran també en un fitxer de format .csv (separador ';') amb nom scafia_yyyymmdd.csv (on yyyy indica l'any, mm el mes i dd el dia), dins de la carpeta definida a *–folderStats*, i amb la següent estructura:

Número de missatge;
data i hora;
node/mote;
bateria(Vdc);
lluminositat(Lux);
temperatura(ºC);
humitat relativa(%)

Per exemple:

1; 24/04/2011 09:00;1;2.5;126;21.5;41.55

Annex B. Execució i compilació

Per executar l'aplicació des de zero hem de fer els següents passos:

1. Copiar la carpeta **/Scafia** al nostre sistema. Té la següent estructura:

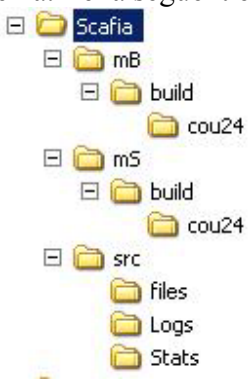


Figura B.1 Arbre de carpetes del sistema

2. Compilar codi font dels nodes/motes remotes, i carregar-lo. Només disposem d'un, que tindrà la identificació 1 seguint l'exemple, en cas de voler afegir més s'haurà de variar la instrucció b i e de forma que a install.*n* i main.srec.out-*n* on indica 'n' serà el valor a modificar amb els identificadors que farem servir. No es pot fer servir el valor 0
    a. Entrem a la carpeta ***/Scafia/mS***
    b. Executem la instrucció ***make cou24 install.1***. Es compilarà el codi font i el dipositarà a la carpeta ***mS/build/cou24***
    c. Entrem a la carpeta ***mS/build/cou24***
    d. Connectem el node al port USB disponible
    e. Executem la instrucció ***meshprog –t/dev/ttyUSB0 –f./main.srec.out-1***. Si no s'inicia s'ha de polsar el botó de reset del node fins carregar el programa al node corresponent
    f. Una vegada finalitzada la càrrega del programa a la mota, la desconnectem del port USB
3. Compilar codi font del node d'enllaç amb la xarxa de nodes, i carregar-lo (obligatòriament és el node amb identificador 0).
    a. Entrem a la carpeta ***/Scafia/mB***
    b. Executem la instrucció ***make cou24 install.0***. Es compilarà el codi font i el dipositarà a la carpeta ***mB/build/cou24***
    c. Entrem a la carpeta ***mB/build/cou24***
    d. Connectem el node al port USB disponible
    e. Executem la instrucció ***meshprog –t/dev/ttyUSB0 –f./main.srec.out-0***. Si no s'inicia s'ha de polsar el botó de reset del node fins carregar el programa al node corresponent
    f. Desconnectem el node/mota del port USB si no el volem fer servir immediatament
4. Compilar el codi font de l'aplicació Java
    a. Entrem a la carpeta ***/src***
    b. Executem la instrucció ***javac files/*.java***
5. Executar l'aplicació de consola de control ScafiaCon per revisar els valors de configuració
    a. Entrem a la carpeta ***/src*** si és necessari
    b. Executem la instrucció ***java files.ScafiaCon -info***

6. Executar l'aplicació de consola de control ScafiaCon per canviar els valors de configuració que ens interessin
    a. Entrem a la carpeta ***/src*** si és necessari
    b. Executem la instrucció ***java files.ScafiaCon*** **[opció-s que volem canviar]** La consola de control permet en aquest moment la introducció dels següents controls i valors (si volem una descripció detallada de cadascun s'ha d'entrar la comanda ***java files.ScafiaCon***):
        - Llindars operatius:
            - Control de bateria del node: -batt
            - Climàtics
                - Temperatura: -tMax, -tMin, -tMaxL, -tMinL
                - Humitat: -hMax, -hMin, -hMaxL, -hMinL
                - Lluminositat (nit/dia): -Llum
        - De sistema:
            - Carpetes de dades: -folderLog (dades del sistema) i –folderStats (estadístiques)
            - Connexió sèrie sistema-xarxa: -source
            - Control d'accionaments
                - Finestres elèctriques: -obrir
                - Electroventiladors: -vent
                - Electrovàlvules: -valve
                - Alarma general: -alarm
            - Control de nodes
                - Saber si un node està actiu: -h (només efectiu si tenim desplegada la xarxa de nodes. Seguir els passos 7 i següents)
            - Gravació de dades de registre: -iniciLog (només efectiu si tenim desplegada la xarxa de nodes. Seguir els passos 7 i següents)
7. Connectem el node/mota d'enllaç de xarxa a un port USB del PC (si no l'hem deixat ja connectat al pas 3.f)
8. Executar l'aplicació de comunicació SerialForwarder
    a. Entrem a la carpeta ***/support/net/tinyos/sf,*** que es troba a la part del sistema on està instal·lat el sistema TinyOS
    b. Executem la instrucció ***java net.tinyos.sf.SerialForwarder***
    c. S'obrirà una aplicació, hem de connectar-la al port 9002, per defecte, i a la velocitat adequada:
        i. Parar el servei clicant el botó ***stop server***
        ii. Revisar que el port sigui el 9002 (altrament el configurat en l'opció ***–source***)
        iii. Introduir la opció ***sf@/dev/ttyUSB0:19200*** on indica *mote communication*
        iv. Engegar de nou el servei clicant el botó ***start server***
        v. Hem de veure que sincronitza correctament. A la finestra de missatges ha de posar:
        Listening to serial/dev/ttyUSB0:19200
        Listening for clients connections on port 9002
        serial/dev/ttyUSB0:19200: resinchronysing
9. Executar l'aplicació de consola de control ScafiaCon com a registrador executant la instrucció ***java files.ScafiaCon –iniciLog*** des de la carpeta ***/Scafia***
10. Despleguem les sondes remotes i les alimentem elèctricament

11. La consola ha de començar a mostrar les dades de telemetria que rep des de la xarxa de nodes

A efectes de telemetria i enviament de missatges tenim que:

- TOS_NODE_IDE de la sonda d'enllaç de xarxa té sempre el valor 0
- TOS_NODE_IDE de la sonda remota té el valor 1 (i següents). El sistema només reacciona als valors climàtics que rebi de la sonda remota

Annex C. Càlcul de conversió bateria, llum, humitat i temperatura

Els nodes tenen quatre sensors connectats als conversors ADC del microcontrolador, que són els següents:

- Bateria
- Lluminositat
- Temperatura
- Humitat relativa

Per mostrar correctament els valors llegits pel conversor ADC s'han d'aplicar una sèrie de càlculs matemàtics, que formen part de l'algorisme de funcionament de la consola de sistema, i que ara es mostren en detall.
El primer que hem de conèixer és quina definició ens permet el conversor ADC. Aquest té una resolució de 10 bits, per tant permet 1024 nivells (2^{10}) diferents de resolució de tensió. La tensió de referència dels ADC és 2,56v (2560mV) per tant el valor de resolució del conversor és de 2,5 mV.
La forma d'obtenir una tensió analògica des de la conversió per nivells feta per l'ADC és la següent::

$$Tensió = \frac{Vref}{1024} Nivell\_ADC$$

Per exemple si el nivell que detecta en un moment donat l'ADC és 1000, llavors la conversió de tensió és:

$$\frac{2560mV}{1024} 1000 = 2500mV = 2{,}5v$$

Finalment la fórmula es pot simplificar per

$$Tensió = 2{,}5mV \cdot Nivell\_ADC$$

C.1 Valor de tensió de la bateria

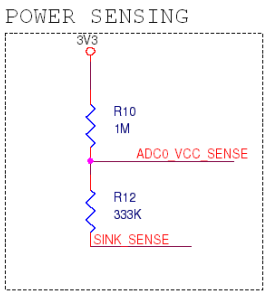


Figura C.1. Divisor de tensió del sensor de bateria

En aquest cas el valor que rep l'ADC 0 prové d'un divisor de tensió, tot i que els valors mostrats de resistència no són correctes. Els valors correctes corresponen a valors iguals de resistència, per exemple 1Mohm, i que fan que la tensió es reparteixi a cada resistència. Per tant la fórmula per obtenir el valor correcte de bateria és:

$$Bateria = 2{,}5mV \cdot Nivell\_ADC0 \cdot 2$$

C.2 Valor de lluminositat

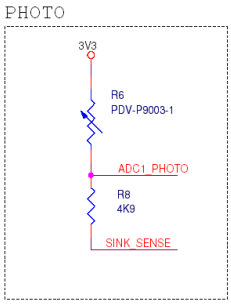


Figura C.2. Divisor de tensió del sensor de lluminositat

El sensor de lluminositat és una resistència variable a la llum, que disminueix el seu valor com més gran és la lluminositat que rep. Aquesta està en un divisor de tensió de forma que al disminuir la resistència per la llum, fa augmentar en la mateixa proporció la tensió en la resistència de càrrega de l'ADC1.
La corba de resistència versus lluminositat del sensor PDV-P9003-1 és la següent:

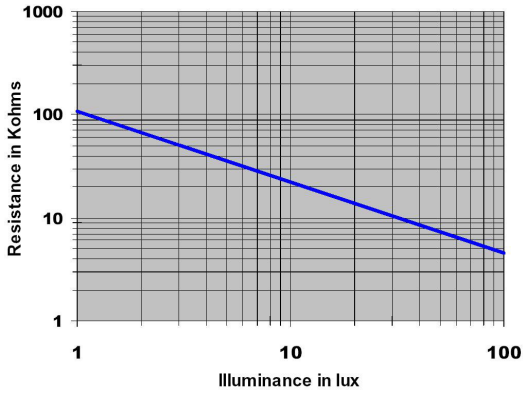


Figura C.3. Corba resistència vs. lux

El que veiem és una característica pràcticament lineal entre 100 kOhms i 4,5 kOhms aproximadament. El valor de la resistència fixa és de 4K9, per tant es pot concloure que quan la lluminositat sigui de 100 lux el valor de tensió al divisor estarà repartit al 50% aproximadament. Obtenim la següent relació:

$$\left.\begin{array}{l} \dfrac{Bateria}{2} \rightarrow 100Lux \\ 2{,}5mV.Nivell\_ADC1 \rightarrow X \end{array}\right\}$$

Que simplifiquem en la següent funció:

$$Lux = \frac{2{,}5 \cdot Nivell\_ADC1}{\dfrac{Bateria}{2}} \cdot 100$$

C.3 Valor de temperatura

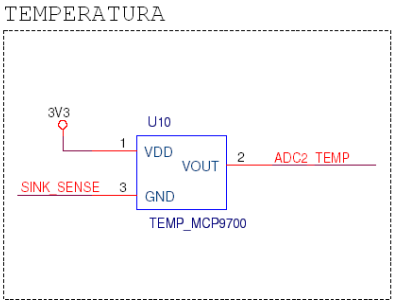


Figura C.4. Sensor de temperatura MCP9700

En aquest cas la tensió prové de la sortida d'un sensor alimentat a Vdd=3V3. Aquest sensor té la següent corba característica:

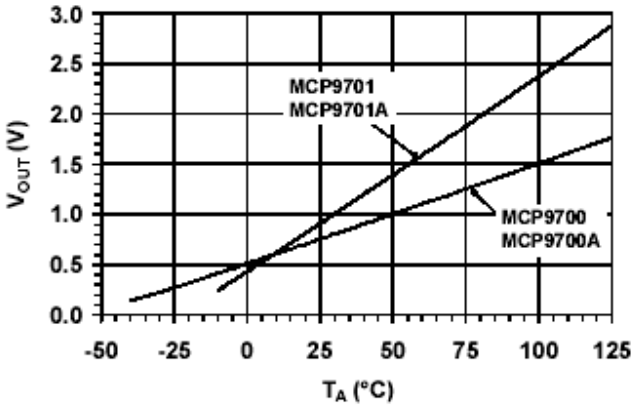


Figura C.5. MCP9700Vout vs. Temperatura

La corba que ens interessa és la del MCP9700, i aquesta varia linealment entre 0 i 100ºC almenys, el curiós és que a 0ºC tenim una tensió de 0,5v que haurem de tenir en compte, i a 100ºC tenim 1,5v. Com que és una relació lineal podem establir que per cada ºC tenim una variació Vout de 10 mV.

La funció de conversió és la següent a ºC:

$$Temperatura = \frac{2{,}5 \cdot Nivell\_ADC2 - 0{,}5}{10}$$

C.4 Valor d'humitat

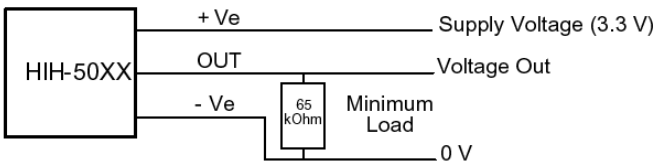


Figura C.6. Muntatge sensor d'humitat HIH-5031

En aquest cas la tensió de sortida es connecta a l'ADC3. La corba característica a 25ºC d'aquest sensor és la següent:

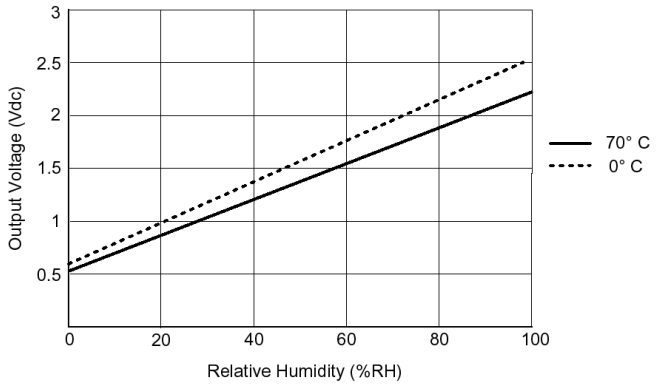


Figura C.7. Corba característica HIH-5031 a 70ºC i 0ºC

Igual que abans podem veure que existeix una linealitat de valors, que permet extreure aproximadament, per cada 1% d'humitat, una variació de tensió de sortida de 17 mV (fem una mitja entre les 2 corbes). Obtenim la següent fórmula de conversió:

$$Humitat = \frac{2{,}5 \cdot Nivell\_ADC3 - 0{,}5}{17}$$

El fabricant a més indica que s'han de fer correccions a causa de la temperatura. La fórmula, a partir de la humitat anterior, és la següent:

$$Humitat\_correccio = \frac{Humitat}{(1{,}0564 - 0{,}00216) \cdot Temperatura}$$

Annex D. Codi de les aplicacions

Només s'indica el codi Java no generat automàticament per l'aplicació *mig*, i el codi nesC de les motes

D.1 ScafiaCon.java

```
package files;
/**TFC ETIS 2n Semestre curs 2010-2011
 * Nom del Projecte: SCAFIA
 *
 * Consultor: Jordi BÃ©cares FerrÃ©s
 * @author Joan Carles MartÃnez RodrÃguez
 *
 */


import net.tinyos.message.Message;
import net.tinyos.message.MessageListener;
import net.tinyos.message.MoteIF;
import net.tinyos.packet.BuildSource;
import net.tinyos.packet.PhoenixSource;
import net.tinyos.util.PrintStreamMessenger;
import java.io.IOException;
import java.util.Timer;
import java.util.TimerTask;
import files.HelloMsg;
import files.MoteMsg;
import files.SensorsMsg;
import files.HallMsg;
import files.Settings;
import files.Utilitats;
import files.ActionMsg;



public class ScafiaCon implements MessageListener{
   public static ScafiaCon instance = null;

                static MoteIF mif;
   MoteIF mif2;
   static int segons=0;
   static int mote=-1;
   static boolean logging=false;
   static Utilitats utils = new Utilitats();
   static boolean netOk=true;
   static long comptador=0;


/**Procediment principal. Analitza la lÃnia d'arguments passada per consola:
 * - Si son correctes els executa, en cas contrari emet missatge d'error
 * - Si no s'inclouen parÃmetres es mostra una petita ajuda
 *
 * @param args
 * @throws Exception
 */
   public static void main (String args[]) throws Exception{
        if (checkIn(args)){
                instance = new ScafiaCon();
                PhoenixSource phoenix;
                phoenix = BuildSource.makePhoenix(PrintStreamMessenger.err);
                String source=utils.getSource();
                phoenix = BuildSource.makePhoenix(source, PrintStreamMessenger.err);

                instance.mif = new MoteIF(phoenix);
                instance.mif2 = new MoteIF(phoenix);

                instance.mif.registerListener(new SensorsMsg(), instance);
                instance.mif.registerListener(new HelloMsg(), instance);
                instance.mif.registerListener(new MoteMsg(), instance);
                instance.mif.registerListener(new HallMsg(), instance);
                instance.mif.registerListener(new Settings(), instance);
                instance.mif.registerListener(new ActionMsg(), instance);

                Timer netCheck = new Timer();
                netCheck.scheduleAtFixedRate(checkNet,0, Integer.parseInt(utils.getTemps())*2000);

                if (mote>=0){
                        sendMessage(mote);
                }
                controlManualAccionaments();
        }
        if (logging==false){
                utils.liniaLog(34,null);
                System.exit(0);
        }

   }


   /**S'analitza la correctesa dels arguments que s'introdueixen per la lÃnia de comandes:
    *  Opcions: [-info] [-iniciLog] [-h <iD>] [-t <segons>] [-tMax <temp.>] [-tMin <temp.>] [-hMax <humitat>] [-hMin <humitat>] [-folderLog <nom>] [-folderStats <nom>]
    * @param args
    * @return
    */
   private static boolean checkIn(String args[]){
        int p = 0;
        boolean check=false;
```

```
String linia="";
for(p=0;p<args.length;p++){linia=linia+" "+args[p];};
utils.liniaLog(33,linia);
if (0 == args.length){
        check=false;
        utils.mostrarOpcions();
}else{
        p=0;
        while (p < args.length){
          if (args[p].equals("-t")) {
                if (p < args.length-1){
                        p++;
                        if(utils.checkTiming(args[p])){
                                utils.gravarINI(5,args[p]);
                                utils.liniaLog(1,args[p]);
                               }
                }else{
                        //System.err.println("Error: falta el valor del temps a configurar");
                        utils.liniaLog(26,args[p]);
                }
 }else if (args[p].equals("-h")) {
  if (p < (args.length-1)){
                        p++;
                        mote=utils.checkHello(args[p]);
                        if(mote==65535){
                                p--;
                        }
                        check=true;
  }else{
          check=true;
          mote=65535;
  }
 }else if (args[p].equals("-tMax")){
  if (p < args.length-1){
                                p++;
                                if(utils.checkTMax(args[p])){
                                        utils.gravarINI(1,args[p]);
                                        utils.liniaLog(2,args[p]);
                                }
  }else{
                                //System.err.println("Error: falta el valor de temperatura mÃ¡xima a configurar");
          utils.liniaLog(27,args[p]);
  }
 }else if (args[p].equals("-tMin")){
  if (p < args.length-1){
                                p++;
                                if(utils.checkTMin(args[p])){
                                        utils.gravarINI(2,args[p]);
                                        utils.liniaLog(3,args[p]);
                                }
  }else{
                                //System.err.println("Error: falta el valor de temperatura mÃnima a configurar");
          utils.liniaLog(27,args[p]);
  }
 }else if (args[p].equals("-hMax")){
  if (p < args.length-1){
                                p++;
                                if(utils.checkHMax(args[p])){
                                        utils.gravarINI(3,args[p]);
                                        utils.liniaLog(4,args[p]);
                                }
  }else{
                                //System.err.println("Error: falta el valor d'humitat mÃ xima a configurar");
          utils.liniaLog(27,args[p]);
  }
 }else if (args[p].equals("-hMin")){
  if (p < args.length-1){
                                p++;
                                if(utils.checkHMin(args[p])){
                                        utils.gravarINI(4,args[p]);
                                        utils.liniaLog(5,args[p]);
                                }
  }else{
                                //System.err.println("Error: falta el valor d'humitat mÃnima a configurar");
          utils.liniaLog(28,args[p]);
  }
 }else if (args[p].equals("-tMaxL")){
  if (p < args.length-1){
                                p++;
                                if(utils.checkTMaxL(args[p])){
                                        utils.gravarINI(9,args[p]);
                                        utils.liniaLog(6,args[p]);
                                }
  }else{
                                //System.err.println("Error: falta el valor de temperatura mÃ¡xima lÃmit a configurar");
          utils.liniaLog(27,args[p]);
  }
 }else if (args[p].equals("-tMinL")){
  if (p < args.length-1){
                                p++;
                                if(utils.checkTMinL(args[p])){
                                        utils.gravarINI(10,args[p]);
                                        utils.liniaLog(7,args[p]);
                                }
  }else{
                                //System.err.println("Error: falta el valor de temperatura mÃnima lÃmit a configurar");
          utils.liniaLog(27,args[p]);
  }
 }else if (args[p].equals("-hMaxL")){
```

```
 if (p < args.length-1){
                                p++;
                                if(utils.checkHMaxL(args[p])){
                                        utils.gravarINI(11,args[p]);
                                        utils.liniaLog(8,args[p]);
                                }
 }else{
                                //System.err.println("Error: falta el valor d'humitat mÃ xima lÃmit a configurar");
        utils.liniaLog(28,args[p]);
 }
 }else if (args[p].equals("-hMinL")){
 if (p < args.length-1){
                                p++;
                                if(utils.checkHMinL(args[p])){
                                        utils.gravarINI(12,args[p]);
                                        utils.liniaLog(9,args[p]);
                                }
 }else{
                                //System.err.println("Error: falta el valor d'humitat mÃnima lÃmit a configurar");
        utils.liniaLog(28,args[p]);
 }
 }else if (args[p].equals("-batt")){
 if (p < args.length-1){
                                p++;
                                if(utils.checkBatt(args[p])){
                                        utils.gravarINI(13,args[p].replace(',', '.'));
                                        utils.liniaLog(10,args[p]);
                                }
 }else{
                                //System.err.println("Error: falta indicar un valor de tensiÃ³ vÃ lid.");
        utils.liniaLog(29,args[p]);
 }
 }else if (args[p].equals("-llum")){
 if (p < args.length-1){
                                p++;
                                if(utils.checkLlum(args[p])){
                                        utils.gravarINI(14,args[p]);
                                        utils.liniaLog(11,args[p]);
                                }
 }else{
                                //System.err.println("Error: falta indicar el valor de lluminositat.");
        utils.liniaLog(30,args[p]);
 }
 }else if (args[p].equals("-folderLog")){
 if (p < args.length-1){
                                p++;
                                if(utils.checkFolder(args[p])){
                                        utils.gravarINI(6,args[p]);
                                        utils.crearDirectori(args[p]);
                                        utils.liniaLog(12,args[p]);
                                }
 }else{
                                //System.err.println("Error: falta un nom de carpeta vÃ lid.");
        utils.liniaLog(31,args[p]);
 }
 }else if (args[p].equals("-folderStats")){
 if (p < args.length-1){
                                p++;
                                if(utils.checkFolder(args[p])){
                                        utils.gravarINI(7,args[p]);
                                        utils.crearDirectori(args[p]);
                                        utils.liniaLog(13,args[p]);
                                }
 }else{
        //System.err.println("Error: falta un nom de carpeta vÃ lid.");
        utils.liniaLog(31,args[p]);
 }
 }else if (args[p].equals("-source")){
 if (p < args.length-1){
                                p++;
                                if(utils.checkSource(args[p])){
                                        utils.gravarINI(8,args[p].replace(':', '_'));
                                        utils.liniaLog(14,args[p]);
                                }
 }else{
        //System.err.println("Error: falta una ruta d'accÃ¨s vÃ lida.");
        utils.liniaLog(32,args[p]);
 }
 }
 else if (args[p].equals("-info")){
  utils.liniaLog(15,args[p]);
  utils.printINIValues();
 }else if (args[p].equals("-iniciLog")){
  check=true;
  logging=true;
  utils.liniaLog(16,args[p]);
 }else if (args[p].equals("-obrir")){
  if (p < args.length-1){
                                p++;
                                if(args[p].equals("on")){
                                        utils.setAccFinestra(true);
                                        utils.setAccManual1(true);
                                        check=true;
                                }else if(args[p].equals("off")){
                                        utils.setAccFinestra(false);
                                        utils.setAccManual1(true);
                                        check=true;
                                }else{
                                        utils.liniaLog(35,args[p]);
```

```
                                utils.setAccManual1(false);
                        }
        }else{
                //System.err.println("Error: falta l'acciÃ³ a realitzar.");
                utils.setAccManual1(false);
                utils.liniaLog(35,args[p]);
        }
        }else if (args[p].equals("-vent")){
        if (p < args.length-1){
                        p++;
                        if(args[p].equals("on")){
                                utils.setAccVent(true);
                                utils.setAccManual2(true);
                                check=true;
                        }else if(args[p].equals("off")){
                                utils.setAccFinestra(false);
                                utils.setAccManual2(true);
                                check=true;
                        }else{
                                utils.liniaLog(35,args[p]);
                                utils.setAccManual2(false);
                        }

        }else{
                //System.err.println("Error: falta l'acciÃ³ a realitzar.");
                utils.liniaLog(35,args[p]);
                utils.setAccManual2(false);
        }
        }else if (args[p].equals("-valve")){
        if (p < args.length-1){
                        p++;
                        if(args[p].equals("on")){
                                utils.setAccValve(true);
                                utils.setAccManual3(true);
                                check=true;
                        }else if(args[p].equals("off")){
                                utils.setAccValve(false);
                                utils.setAccManual3(true);
                                check=true;
                        }else{
                                utils.liniaLog(35,args[p]);
                                utils.setAccManual3(false);
                        }

        }else{
                //System.err.println("Error: falta l'acciÃ³ a realitzar.");
                utils.liniaLog(35,args[p]);
                utils.setAccManual3(false);
        }
        }else if (args[p].equals("-alarm")){
        if (p < args.length-1){
                        p++;
                        if(args[p].equals("on")){
                                utils.setAccAlarm(true);
                                utils.setAccManual4(true);
                                check=true;
                        }else if(args[p].equals("off")){
                                utils.setAccAlarm(false);
                                utils.setAccManual4(true);
                                check=true;
                        }else{
                                utils.liniaLog(35,args[p]);
                                utils.setAccManual4(false);
                        }

        }else{
                //System.err.println("Error: falta l'acciÃ³ a realitzar.");
                utils.liniaLog(35,args[p]);
                utils.setAccManual4(false);
        }
        }else{
        utils.mostrarOpcions();
        utils.liniaLog(34,null);
        }
            p++;
            }
    }
        return(check);
}

/**Procediment que gestiona la recepciÃ³ de missatges des de les motes. Els missatges sÃ³n de diferents tipus:
 * - MsgHello: missatge amb identificador de node i temps programat (a l'inici del node)
 * - SensorsMsg: missatge amb els valors dels sensors (cÃclic)
 * - MoteMsg: missatge amb valor d'id del node i la seva Mac. Es genera al polsar el botÃ³ d'usuari
 * - HallMSg: missatge que avisa de detecciÃ³ al sensor Hall
 * - Settings: missatge que indica el valor de temps de lectura de sensors programat al node
 * - ActionMsg: missatge sobre l'estat de l'accionament a que es refereix
 * Es fa el tractament de cada missatge, segons el seu tipus, escrivint els assentaments als fitxers de logs
 * i analitzant si Ã©s necessari activar alarmes o canvis de temps.
 */

public void messageReceived(int to, Message m) {
        String stamp =utils.getStamp(1);
        String identitat;
        String linia = null;
        String stats = null;
        String flagA = utils.flagAlarm();
        if(m instanceof SensorsMsg){
                SensorsMsg sm = (SensorsMsg) m;
```

```
                double lecturaVoltsBat = utils.passarAVolts(sm.get_vrefmilivolts(),sm.get_countsBat())*2;    //2 es el factor d'atenuacio del divisor
                double lecturaVoltsTemp = utils.passarAVolts(sm.get_vrefmilivolts(),sm.get_countsTemp());
                double temperaturaCentigrads = (lecturaVoltsTemp-0.5)/0.01;
                double lecturaVoltsLlum = utils.passarAVolts(sm.get_vrefmilivolts(),sm.get_countsPhoto());
                double llumLux = (lecturaVoltsLlum*100/(lecturaVoltsBat/2));
                double lecturaVoltsHum = utils.passarAVolts(sm.get_vrefmilivolts(),sm.get_countsHum());
                double humitatRelativa = (lecturaVoltsHum-0.5)/0.017;//RH segons transferÃ¨ncia pel sensor HIH-5031
                humitatRelativa = humitatRelativa/(1.0564-0.00216*temperaturaCentigrads);//CompensaciÃ³ segons fabricant pel HIH-5031
                utils.carregaValors();
                if (sm.get_moteId()==0){
                        identitat="] Valors Node Ref. ";
                        linia=(flagA+"TEL    =>|    "    +stamp+"    ["+Long.toString(sm.get_counter())+identitat+sm.get_moteId()+"\tBateria(V):["+utils.formatejarDouble(lecturaVoltsBat)+"]
Lluminositat(Lux):["+utils.formatejarDouble(llumLux)+"]                                                                                    Temperatura(ÂºC):["+utils.formatejarDouble(temperaturaCentigrads)+"]
Humitat(%):["+utils.formatejarDouble(humitatRelativa)+"]");
                        utils.registrarMov(linia,1);
                        linia=null;
                }else{
                        netOk=true;
                        identitat="] Valors Node Remot ";
                        flagA=utils.flagAlarm();
                        linia=(flagA+"TEL    =>|    "    +stamp+"    ["+Long.toString(sm.get_counter())+identitat+sm.get_moteId()+"\tBateria(V):["+utils.formatejarDouble(lecturaVoltsBat)+"]
Lluminositat(Lux):["+utils.formatejarDouble(llumLux)+"]                                                                                    Temperatura(ÂºC):["+utils.formatejarDouble(temperaturaCentigrads)+"]
Humitat(%):["+utils.formatejarDouble(humitatRelativa)+"]");
                        utils.registrarMov(linia,1);
                        linia=null;
                        gestioClima(sm.get_moteId(),lecturaVoltsBat,temperaturaCentigrads,humitatRelativa);
                }
                if (sm.get_militimer()/1000!=Integer.parseInt(utils.getTemps())){
                        resetTemps(Integer.parseInt(utils.getTemps()),sm.get_moteId(),comptador);
                }
                stats=sm.get_counter()+";"+stamp+";"+sm.get_moteId()+";"+lecturaVoltsBat+";"+llumLux+";"+temperaturaCentigrads+";"+humitatRelativa;
                utils.registrarMov(stats,2);
        }
        else if(m instanceof HelloMsg){
                HelloMsg ham = (HelloMsg) m;
                linia=(flagA+"ON!  =>|  "+stamp+"  ["+ham.get_seqNum()+"] Node "+ham.get_moteId()+ " estÃ  operatiu. Temps de cicle de lectura de sensors configurat =
"+ham.get_militimer()/1000+" segons");
                if (ham.get_militimer()/1000!=Integer.parseInt(utils.getTemps())){
                        utils.registrarMov(linia,1);
                        linia=null;
                        resetTemps(Integer.parseInt(utils.getTemps()),ham.get_moteId(),ham.get_seqNum());
                }
                if(ham.get_moteId()>0){netOk=true;};
        }
        else if(m instanceof MoteMsg){
                MoteMsg mm = (MoteMsg) m;
                String macStr="";
                boolean first=true;
                String s="";
                for(int i=0;i<mm.get_macAddr().length;i++){
                        if(first){
                                first=false;
                                s="";
                        }
                        else{
                                s=":";
                        }
                        macStr=macStr+s+mm.getElement_macAddr(i);
                }
                if(mm.get_moteId()>0){netOk=true;};
                linia=(flagA+"USR =>| " +stamp+" El node remot " + mm.get_moteId()+ " estÃ  operatiu i tÃ© la MAC "+macStr);
        }else if(m instanceof HallMsg){
                HallMsg hm = (HallMsg) m;
                if(hm.get_moteId()>0){netOk=true;};
                linia=(flagA+"HAL =>| " +stamp+" El node " + hm.get_moteId()+" detecta activaciÃ³ de dispositiu extern.");
                //System.out.println(linia);
        }else if(m instanceof Settings){
                Settings st = (Settings) m;
                if(st.get_moteId()>0){netOk=true;};
                linia=(flagA+"TIM =>| " +stamp+" ["+st.get_seqNum()+ "] El node " + st.get_moteId()+" informa d'un temps de cicle de lectura de sensors de "+st.get_setmilitimer()/1000+"
segons");
        }else if(m instanceof ActionMsg){
                ActionMsg am = (ActionMsg) m;
                linia=(flagA+"ACC    =>|    "    +stamp+"    ["+am.get_seqNum()+    "]    El    node    "    +    am.get_moteId()+"    informa
"+accionamentActivat(am.get_action())+estatAccionament(am.get_maniobra(),1));
        }else{
                linia=(flagA+stamp+" Error: S'ha rebut un missatge inesperat des de la xarxa de nodes, el amType es "+m.amType());
                System.err.println(linia);
        }
        if(linia!=null){utils.registrarMov(linia,1);};
    }
    /**FunciÃ³ que torna el tipus d'accionament activat informaciÃ³ que serveix per completar la lÃnia
     * del fitxer d'auditoria.
     *
     * @param accionamentN
     * @return
     */

    private static String accionamentActivat(int accionamentN){
        String linia=null;
        switch (accionamentN){
                case 1: linia=" que els accionaments elÃ¨ctrics de finestres";break;
                case 2: linia=" que els electroventiladors";break;
                case 3: linia=" que les electrovÃ lvules de control de polvoritzaciÃ³";break;
                case 4: linia=" que les alarmes generals de sistema";break;
        }
        return linia;
    }
```

```
/**FunciÃ³ que torna l'estat dels accionaments, serveix per completar la linia d'informaciÃ³
 * del fitxer d'auditoria.
 * @param estat
 * @param opcio
 * @return
 */

private static String estatAccionament(int estat, int opcio){
	String linia=null;
	if (opcio==1){
		if (estat==0){
			linia=" estan desactivats.";
		}else{
			linia=" estan activats.";
		}
	}else if(opcio==2){
		if (estat==0){
			linia=" s'han de desactivar.";
		}else{
			linia=" s'han d'activar.";
		}
	}
	return linia;
}

/**FunciÃ³ que permet enviar a la mota indicada (per defecte Ã©s 0), quin accionament i quina
 * maniobra volem que faci (0: apagar, 1: activar).
 *
 * @param moteId
 * @param accionament
 * @param maniobra
 */
private static void accionaments(int moteId,int accionament, int maniobra){
	String linia;
	String flagA = utils.flagAlarm();
	boolean estat=false;
	if (maniobra==0){
		estat=false;
	}else{
		estat=true;

	}
	linia=(flagA+"ACC |=> " +utils.getStamp(1)+" ["+comptador+ "] S'envia al node " + moteId + accionamentActivat(accionament)+estatAccionament(maniobra,2));
	ActionMsg msg = new ActionMsg();
	msg.set_moteId(moteId);
	msg.set_seqNum(comptador);
	msg.set_action((short)accionament);
	msg.set_maniobra((short)maniobra);
	try {
		mif.send(moteId,msg);
		switch(accionament){
			case 1:{utils.setFinestres(estat);break;}
			case 2:{utils.setElectroVent(estat);break;}
			case 3:{utils.setElectroValve(estat);break;}
		}
	} catch (IOException e) {
		utils.registrarMov(linia,1);
		linia=("Error: No s'ha pogut enviar el missatge [" +comptador+"] de control d'accionaments al node ["+moteId+"]");
	}
	utils.registrarMov(linia,1);
	comptador++;
}

/**S'envia al node indicat per moteId, un nou valor de temps (mS) de lectura de sensors per reprogramar-lo al valor del sistema.
 *
 * @param moteTemps
 * @param moteId
 * @param seqNum
 */
private static void resetTemps(int moteTemps,int moteId,long seqNum){
	String linia;
	String flagA=utils.flagAlarm();
	linia=(flagA+"TIM |=> "+utils.getStamp(1)+"["+seqNum+"] Canvi automÃ tic de temps de cicle de lectura de sensors a node ["+moteId+"]. Valor de cicle que s'envia =
"+utils.getTemps()+" segons");
	Settings msg = new Settings();
	msg.set_moteId(moteId);
	msg.set_setmilitimer(moteTemps*1000);
	msg.set_seqNum(seqNum);
	try {
		mif.send(moteId,msg);
	} catch (IOException e) {
		utils.registrarMov(linia,1);
		linia=("Error: No s'ha pogut enviar el missatge [" +seqNum+"] de canvi de temps de cicle al node ["+moteId+"]");
		//System.err.println(linia);
	}
	utils.registrarMov(linia,1);
	comptador++;
}


/**Inici de gestiÃ³ d'alarmes del sistema, segons els valors rebuts de les lectures dels sensors
 *
 * @param sensor
 * @param bateria
 * @param llum
 * @param temp
 * @param humitat
 */
private void gestioClima(int sensor, double bateria, double temp, double humitat){
```

```
            gestioBat(sensor,bateria);
            gestioTemp(sensor,temp,humitat);
            gestioHum(sensor,temp,humitat);
        }

        /**GestiÃ³ del valor llegit de bateria en volts per detectar si Ã©s causa d'alarma. En aquest cas mostra un missatge
         * d'advertÃ¨ncia, i tornarÃ  a mostrar un missatge de normalitat quan es recuperi el valor.
         *
         * @param sensor
         * @param bateria
         */
        private void gestioBat(int sensor, double bateria){
            String linia;
            if (bateria<Double.parseDouble(utils.getBatt())){
                if(!utils.getAlarmBatt()){
                    utils.setAlarmBatt(true);
                    linia=("SCAFIA A! ALARM " +utils.getStamp(1)+" La bateria del node "+sensor+" tÃ© un valor de tensiÃ³ massa baix: "
                            +utils.formatejarDouble(bateria)+"V [lÃmit "+utils.getBatt() +"V]");
                    utils.registrarMov(linia,1);
                }
            }else{
                if(utils.getAlarmBatt()){
                    utils.setAlarmBatt(false);
                    linia=("SCAFIA - ALARM " +utils.getStamp(1)+" La bateria del node "+sensor+" ha recuperat un valor de tensiÃ³ correcte: "
                            +utils.formatejarDouble(bateria)+"V");
                    utils.registrarMov(linia,1);
                }
            }
        }

        /**GestiÃ³ del valor llegit de temperatura. Es tracta de detectar si Ã©s un valor anÃ²mal, i si Ã©s el cas activar
         * les alarmes i sistemes de control corresponents.
         *
         * @param sensor
         * @param temp
         * @param humitat
         */
        private void gestioTemp(int sensor,double temp,double humitat){
            String linia;
            if(temp>Double.parseDouble(utils.getTminl()) && temp<Double.parseDouble(utils.getTmaxl())){
                if (utils.getAlarmGTemp()){
                    utils.setAlarmGTemp(false);
                    linia=("SCAFIA - ALARM " +utils.getStamp(1)+" La temperatura, segons el node "+sensor+", ha recuperat els valors del sistema:
"+utils.formatejarDouble(temp)+"ÂºC");
                    utils.registrarMov(linia,1);
                    if (!utils.getFlagGAlarm()){accionaments(0,4,0);};
                }
                if(temp>Double.parseDouble(utils.getTmin()) && temp<Double.parseDouble(utils.getTmax())){
                    if(utils.getAlarmTemp()){
                        linia=("SCAFIA - ALARM " +utils.getStamp(1)+" La temperatura, segons el node "+sensor+", ha recuperat els valors operatius: "
                                +utils.formatejarDouble(temp)+"ÂºC");
                        utils.registrarMov(linia,1);
                        utils.setAlarmTemp(false);
                        accionamentsTemperatura(0);
                    }
                }else if(temp>Double.parseDouble(utils.getTmax())){
                    if(!utils.getAlarmTemp()){
                        utils.setAlarmTemp(true);
                        linia=("SCAFIA A! ALARM " +utils.getStamp(1)+" La temperatura, segons el node "+sensor+", tÃ© un valor superior al mÃ xim: "
                                +utils.formatejarDouble(temp)+"ÂºC [lÃmit "+utils.getTmax()+"ÂºC]");
                        utils.registrarMov(linia,1);
                        if(humitat<Double.parseDouble(utils.getHmax())){
                            linia=("SCAFIA A! ACC " +utils.getStamp(1)+" S'activa el sistema de polvoritzaciÃ³ d'aigua per baixar la temperatura.");
                            utils.registrarMov(linia,1);
                            accionaments(0,3,1);
                        }else{
                            linia=("SCAFIA A! ACC " +utils.getStamp(1)+" S'activa l'obertura de finestres i ventiladors per baixar la temperatura.");
                            utils.registrarMov(linia,1);
                            accionaments(0,1,1);
                            accionaments(0,2,1);
                        }
                    }
                }else if(temp<Double.parseDouble(utils.getTmin())){
                    if(!utils.getAlarmTemp()){
                        utils.setAlarmTemp(true);
                        linia=("SCAFIA A! ALARM " +utils.getStamp(1)+" La temperatura, segons el node "+sensor+", tÃ© un valor inferior al mÃnim: "
                                +utils.formatejarDouble(temp)+"ÂºC [lÃmit "+utils.getTmin() +"ÂºC]");
                        utils.registrarMov(linia,1);
                        linia=("SCAFIA A! ACC " +utils.getStamp(1)+" Es tanquen les finestres, s'atura la polvoritzaciÃ³ d'aigua, i els ventiladors per augmentar la temperatura.");
                        utils.registrarMov(linia,1);
                        accionamentsTemperatura(0);
                    }
                }
            }else{
                if (!utils.getAlarmGTemp()){
                    linia=("SCAFIA A! ALARM " +utils.getStamp(1)+" La temperatura, segons el node "+sensor+", estÃ  fora dels valors mÃ xims del sistema: "
                            +utils.formatejarDouble(temp)+"ÂºC [lÃmits: mÃ x. "+utils.getTmaxl() +"ÂºC, mÃn. "+utils.getTminl()+"ÂºC]");
                    utils.registrarMov(linia,1);
                    if (!utils.getFlagGAlarm()){
                        linia=("SCAFIA A! ACC " +utils.getStamp(1)+" S'activa el sistema d'alarma de la instalÂ·laciÃ³.");
                        utils.registrarMov(linia,1);
                        accionaments(0,4,1);
                    }
                    utils.setAlarmGTemp(true);
                }
            }
        }
```

```
/**Control dels accionaments. Si el valor d'opciÃ³ Ã©s 0 tots els accionaments actius s'apagaran
 *
 * @param opcio
 */
private void accionamentsTemperatura(int opcio){
	if(!utils.getAlarmGHum() && !utils.getAlarmHum()){
		if(utils.getFinestres()){
			accionaments(0,1,opcio);
		}
		if(utils.getElectroVent()){
			accionaments(0,2,opcio);
		}
		if(utils.getElectroValve()){
		accionaments(0,3,opcio);
		}
	}
}


/**GestiÃ³ del valor llegit d'humitat. Es tracta de detectar si Ã©s un valor anÃ²mal, i si Ã©s el cas activar
 * les alarmes i sistemes de control corresponents.
 *
 * @param sensor
 * @param temp
 * @param humitat
 */
private void gestioHum(int sensor,double temp,double humitat){
	String linia;
	if(humitat>Double.parseDouble(utils.getHminl()) && humitat<Double.parseDouble(utils.getHmaxl())){
		if (utils.getAlarmGHum()){
			utils.setAlarmGHum(false);
			linia=("SCAFIA - ALARM " +utils.getStamp(1)+" La humitat, segons el node "+sensor+", ha recuperat els valors del sistema: "+utils.formatejarDouble(humitat)+"%");
			utils.registrarMov(linia,1);
			if (!utils.getFlagGAlarm()){accionaments(0,4,0);};
		}
		if(humitat>Double.parseDouble(utils.getHmin()) && humitat<Double.parseDouble(utils.getHmax())){
			if(utils.getAlarmHum()){
				utils.setAlarmHum(false);
				linia=("SCAFIA - ALARM " +utils.getStamp(1)+" La humitat, segons el node "+sensor+", ha recuperat els valors operatius: "
					+utils.formatejarDouble(humitat)+"%");
				utils.registrarMov(linia,1);
				accionamentsHumitat(0);
			}
		}else if(humitat>Double.parseDouble(utils.getHmax())){
			if(!utils.getAlarmHum()){
				utils.setAlarmHum(true);
				linia=("SCAFIA A! ALARM " +utils.getStamp(1)+" La humitat, segons el node "+sensor+", tÃ© un valor superior al mÃ xim: "
					+utils.formatejarDouble(humitat)+"% [lÃmit "+utils.getHmax() +"%]");
				utils.registrarMov(linia,1);
				linia=("SCAFIA A! ACC " +utils.getStamp(1)+" Es tanca el sistema de polvoritzaciÃ³ amb aigua per baixar la humitat.");
				utils.registrarMov(linia,1);
				accionaments(0,3,0);
				if(temp>Double.parseDouble(utils.getTmin())){
					linia=("SCAFIA A! ACC " +utils.getStamp(1)+" S'activa l'obertura de finestres per baixar la humitat.");
					utils.registrarMov(linia,1);
					accionaments(0,1,1);
				}
			}
		}else if(humitat<Double.parseDouble(utils.getHmin())){
			if(!utils.getAlarmHum()){
				utils.setAlarmHum(true);
				linia=("SCAFIA A! ALARM " +utils.getStamp(1)+" La humitat, segons el node "+sensor+", tÃ© un valor inferior al mÃnim: "
					+utils.formatejarDouble(humitat)+"% [lÃmit "+utils.getHmin() +"%]");
				utils.registrarMov(linia,1);
				if(temp>Double.parseDouble(utils.getTmin())){
					linia=("SCAFIA A! ACC " +utils.getStamp(1)+" S'activa el sistema de polvoritzaciÃ³ per augmentar la humitat.");
					utils.registrarMov(linia,1);
					accionaments(0,3,1);
				}
			}
		}
	}else{
		if (!utils.getAlarmGHum()){
			linia=("SCAFIA A! ALARM " +utils.getStamp(1)+" La humitat, segons el node "+sensor+", estÃ  fora dels valors mÃ xims del sistema: "
				+utils.formatejarDouble(humitat)+"% [lÃmits: mÃ x. "+utils.getHmaxl() +"%, mÃn. "+utils.getHminl()+"%]");
			utils.registrarMov(linia,1);
			if (!utils.getFlagGAlarm()){
				linia=("SCAFIA A! ACC " +utils.getStamp(1)+" S'activa el sistema d'alarma de la instalÂ·laciÃ³.");
				utils.registrarMov(linia,1);
				accionaments(0,4,1);
			}
			utils.setAlarmGHum(true);
		}
	}
}

/**Control dels accionaments. Si el valor d'opciÃ³ Ã©s 0 tots els accionaments actius s'apagaran
 *
 * @param opcio
 */
private void accionamentsHumitat(int opcio){
	if(!utils.getAlarmGTemp() && !utils.getAlarmTemp()){
		if(utils.getFinestres()){
			accionaments(0,1,opcio);
		}
		if(utils.getElectroValve()){
			accionaments(0,3,opcio);
		}
	}
```

```
    }


    /**Envia un missatge tipus HelloMsg al node especificat pel valor mote.
     *
     * @param mote
     */
    public static void sendMessage(int mote) {
        String linia=null;
        HelloMsg msg = new HelloMsg();
        msg.set_moteId(mote);
        msg.set_seqNum(0);
        msg.set_militimer(Integer.parseInt(utils.getTemps()));
        linia=("SCAFIA CON |=> "+utils.getStamp(1)+" [0] PeticiÃ³ d'estat a sensor remot amb identificaciÃ³ "+mote);
        try {
            mif.send(mote,msg);
        } catch (IOException e) {
            utils.registrarMov(linia,1);
            linia=("Error: No s'ha pogut enviar el missatge [0] als nodes");
            System.err.println(linia);
        }

        utils.registrarMov(linia,1);
    }
    /**Si en la consola de control hem decidit activar manualment algun accionament, la segÃ¼ent funciÃ³
     * s'encarrega de fer-ho.
     *
     */
    private static void controlManualAccionaments(){
        if (utils.getAccManual1()){
            if (utils.getAccFinestra()){
                accionaments(0,1,1);
            }else{
                accionaments(0,1,0);
            }
        }
        if (utils.getAccManual2()){
            if (utils.getAccVent()){
                accionaments(0,2,1);
            }else{
                accionaments(0,2,0);
            }
        }
        if (utils.getAccManual3()){
            if (utils.getAccValve()){
                accionaments(0,3,1);
            }else{
                accionaments(0,3,0);
            }
        }
        if (utils.getAccManual4()){
            if (utils.getAccAlarm()){
                accionaments(0,4,1);
            }else{
                accionaments(0,4,0);
            }
        }
    }

    /**FunciÃ³ que permet controlar si la xarxa de nodes estÃ  caiguda. Quan finalitza el comptador
     * s'activa i si la variable netOk tÃ© valor false llavors indica que la xarxa estÃ  caiguda.
     * La variable netOk canvia el seu valor quan rep un missatge de telemetria d'un dels sensors
     * de la xarxa.
     *
     */
    static TimerTask checkNet = new TimerTask(){
        public void run(){
            String linia;
            if (!netOk){
                if (!utils.getAlarmNet()){
                    linia=("SCAFIA A! NET " +utils.getStamp(1)+" La xarxa de nodes estÃ  caiguda, o els nodes tenen un temps de lectura de sensors incorrecte.");
                    utils.registrarMov(linia,1);
                    if (!utils.getFlagGAlarm()){
                        linia=("SCAFIA A! ACC " +utils.getStamp(1)+" S'activa el sistema d'alarma de la instalÂ·laciÃ³.");
                        utils.registrarMov(linia,1);
                        accionaments(0,4,1);
                    }
                    utils.setAlarmNet(true);
                }
            }else{
                if (utils.getAlarmNet()){
                    utils.setAlarmNet(false);
                    linia=("SCAFIA - NET " +utils.getStamp(1)+" La xarxa de nodes s'ha recuperat, o els nodes tenen un temps de lectura de sensors correcte.");
                    utils.registrarMov(linia,1);
                    if (!utils.getFlagGAlarm()){
                        linia=("SCAFIA - ACC " +utils.getStamp(1)+" Es desactiva el sistema d'alarma de la instalÂ·laciÃ³.");
                        utils.registrarMov(linia,1);
                        accionaments(0,4,0);
                    }
                }
                netOk=false;
            }
        }
    };


}
```

D.2 Utilitats.java

```
package files;
/**TFC ETIS 2n Semestre curs 2010-2011
 * Nom del Projecte: SCAFIA
 *
 * Consultor: Jordi Bécares Ferrés
 * @author Joan Carles Martínez Rodríguez
 *
 */
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Properties;


/**Class que ajuda en la gestió de logs i consola del sistema.
 */
public class Utilitats {
   private static String tmax;
   private static String tmin;
   private static String hmax;
   private static String hmin;
   private static String temps;
   private static String tmaxl;
   private static String tminl;
   private static String hmaxl;
   private static String hminl;
   private static String llum;
   private static String batt;
   private static String folderLog;
   private static String folderStats;
   private static String source;
   private static boolean alarmTemp=false;
   private static boolean alarmHum=false;
   private static boolean alarmBatt=false;
   private static boolean alarmNet=false;
   private boolean finestres=false;
   private boolean electrovent=false;
   private boolean electrovalve=false;

   private boolean alarmGTemp=false;
   private boolean alarmGHum=false;

   private boolean accManual1=false;
   private boolean accManual2=false;
   private boolean accManual3=false;
   private boolean accManual4=false;

   private boolean accFinestra=false;
   private boolean accVent=false;
   private boolean accValve=false;
   private boolean accAlarm=false;


   public Properties misPropiedades;
/**Constructor de la classe, en combinació amb carregaValors()
 */
   public Utilitats() {
         carregaValors();
   }

   /**Llegeix els valors del fitxer scafia.ini i els carrega en les variables del sistema.
    */
   public void carregaValors(){
         String propertiesFilePath = ("./scafia.ini");
         Properties configuracio = new Properties();
         try {
               configuracio.load(new FileInputStream(propertiesFilePath));
               tmax = configuracio.getProperty("sistema.tmax");
               tmin = configuracio.getProperty("sistema.tmin");
               hmax = configuracio.getProperty("sistema.hmax");
               hmin = configuracio.getProperty("sistema.hmin");
               temps = configuracio.getProperty("sistema.temps");
               tmaxl = configuracio.getProperty("sistema.tmaxl");
               tminl = configuracio.getProperty("sistema.tminl");
               hmaxl = configuracio.getProperty("sistema.hmaxl");
               hminl = configuracio.getProperty("sistema.hminl");
               llum = configuracio.getProperty("sistema.llum");
               batt = configuracio.getProperty("sistema.batt");
               folderLog = configuracio.getProperty("sistema.folderLog");
               folderStats = configuracio.getProperty("sistema.folderStats");
               source = configuracio.getProperty("sistema.source").replace('_', ':');
         }catch (Exception e){
               System.out.println("Error: excepció a l'obrir el fitxer de configuració, no es troba o bé està protegit");
         }
   }

   /**Diferents getters() que tornen els valor operatius del sistema
    *
    * @return
    */
```

```
public static String getTmax(){return tmax;};
public static String getTmin(){return tmin;};
public static String getHmax(){return hmax;};
public static String getHmin(){return hmin;};
public static String getTemps(){return temps;};
public static String getTmaxl(){return tmaxl;};
public static String getTminl(){return tminl;};
public static String getHmaxl(){return hmaxl;};
public static String getHminl(){return hminl;};
public static String getLlum(){return llum;};
public static String getBatt(){return batt;};
public static String getfolderLog(){return folderLog;};
public static String getfolderStats(){return folderStats;};
public static String getSource(){return source;};

/**Grava les modificacions dels valors operatius del sistema que s'han passat per consola.
 *
 * @param opcio
 * @param valor
 * @throws FileNotFoundException
 */
public void gravarINI(int opcio, String valor)  {
        String propertiesFilePath = ("scafia.ini");
        Properties configuracio = new Properties();
        try{
                configuracio.load(new FileInputStream(propertiesFilePath));
                switch (opcio){
                        case 1:{
                                configuracio.setProperty("sistema.tmax",valor);
                                break;
                        }case 2:{
                                configuracio.setProperty("sistema.tmin",valor);
                                break;
                        }case 3:{
                                configuracio.setProperty("sistema.hmax",valor);
                                break;
                        }case 4:{
                                configuracio.setProperty("sistema.hmin",valor);
                                break;
                        }case 5:{
                                configuracio.setProperty("sistema.temps",valor);
                                break;
                        }case 6:{
                                configuracio.setProperty("sistema.folderLog",valor);
                                break;
                        }case 7:{
                                configuracio.setProperty("sistema.folderStats",valor);
                                break;
                        }case 8:{
                                configuracio.setProperty("sistema.source",valor);
                                break;
                        }case 9:{
                                configuracio.setProperty("sistema.tmaxl",valor);
                                break;
                        }case 10:{
                                configuracio.setProperty("sistema.tminl",valor);
                                break;
                        }case 11:{
                                configuracio.setProperty("sistema.hmaxl",valor);
                                break;
                        }case 12:{
                                configuracio.setProperty("sistema.hminl",valor);
                                break;
                        }case 13:{
                                configuracio.setProperty("sistema.batt",valor);
                                break;
                        }case 14:{
                                configuracio.setProperty("sistema.llum",valor);
                                break;
                        }
                }
                configuracio.store(new FileOutputStream(propertiesFilePath),null);
                carregaValors();
        }catch (Exception e){
                System.out.println("Error: excepció a l'obrir el fitxer de configuració, no es troba o bé està protegit");
        }

}
/**Torna un segell de data i hora, o només data, segons l'opció demanada, que servirà per marcar cada assentament
 * al fitxer de log
 *
 * @param opcio
 * @return
 */
public static String getStamp(int opcio){
   Date ahora = new Date();
   SimpleDateFormat formatData = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
   SimpleDateFormat formatData2 = new SimpleDateFormat("yyyyMMdd");
   if (opcio==1){
       return formatData.format(ahora);
   } else {
       return formatData2.format(ahora);
       }
}

/**Grava l'assentament que correspongui, sigui de log o bé estadístic, al fitxer que correspon.
 * L'assentament és el valor passat amb línia
 *
 * @param fitxer
```

```
 * @param linia
 * @throws IOException
 */
private static void gravarLog(File fitxer,String linia) throws IOException{
	try{
		PrintWriter fileOut=new PrintWriter(new FileWriter(fitxer,true));
		fileOut.println(linia);
		fileOut.close();
	}catch (IOException ex){
		System.err.println("No s'ha pogut fer el registre al fitxer de log");
	}
}

/**Gestor de gravació d'assentaments. Construeix el nom del fitxer segons l'opció desitjada: 1-log, 2-estadístiques i 3-accions. Línia conté
 * el valor de l'assentament.
 *
 * @param linia
 * @param opcio
 */
public static void registrarMov(String linia, int opcio){
	File fitxerLog=new File(folderLog,"scafia_"+getStamp(2)+".log");
	File fitxerStats=new File(folderStats,"scafia_"+getStamp(2)+".csv");
	try {
		if (opcio == 1){
			gravarLog(fitxerLog,linia);
			System.out.println(linia);
		}else{
			gravarLog(fitxerStats,linia);
		}
	} catch (IOException e) {
		e.printStackTrace();
		System.err.println("Error: no s'ha pogut registrar l'acció demanada");
	}

}

/**Crea el directori al disc passat per paràmetre.
 *
 * @param directori
 */
public void crearDirectori(String directori){
	File folder = new File(directori);
	folder.mkdir();
}


/**Diferents Getters sobre l'estat d'alarmes del sistema
 *
 * @return
 */
public boolean getAlarmTemp(){return alarmTemp;};
public boolean getAlarmHum(){return alarmHum;};
public boolean getAlarmBatt(){return alarmBatt;};
public boolean getAlarmNet(){return alarmNet;};
public boolean getAlarmGTemp(){return this.alarmGTemp;};
public boolean getAlarmGHum(){return this.alarmGHum;};


/**Diferents setters per modificar l'estat d'alarmes del sistema
 *
 * @param estat
 */
public void setAlarmTemp(boolean estat){alarmTemp=estat;};
public void setAlarmHum(boolean estat){alarmHum=estat;};
public void setAlarmBatt(boolean estat){alarmBatt=estat;};
public void setAlarmNet(boolean estat){alarmNet=estat;};
public void setAlarmGTemp(boolean estat){this.alarmGTemp=estat;};
public void setAlarmGHum(boolean estat){this.alarmGHum=estat;};


/**Torna cert si existeix alguna alarma al sistema
 *
 * @return
 */
public boolean getFlagAlarm(){
	boolean check = false;
	if (alarmTemp || alarmHum || alarmBatt || alarmNet || alarmGTemp || alarmGHum){
		check = true;
	}
	return check;
}


/**Diferents Getters sobre l'estat dels accionaments automàtics del sistema
 *
 * @return
 */
public boolean getFinestres(){return this.finestres;};
public boolean getElectroVent(){return this.electrovent;};
public boolean getElectroValve(){return this.electrovalve;};
public boolean getAccManual1(){return this.accManual1;};
public boolean getAccManual2(){return this.accManual2;};
public boolean getAccManual3(){return this.accManual3;};
public boolean getAccManual4(){return this.accManual4;};
public boolean getAccFinestra(){return this.accFinestra;};
public boolean getAccVent(){return this.accVent;};
public boolean getAccValve(){return this.accValve;};
public boolean getAccAlarm(){return this.accAlarm;};
```

```
    /**Diferents setters per modificar l'estat dels accionaments automàrics del sistema
     *
     * @param estat
     */
    public void setFinestres(boolean estat){this.finestres=estat;};
    public void setElectroVent(boolean estat){this.electrovent=estat;};
    public void setElectroValve(boolean estat){this.electrovalve=estat;};
    public void setAccManual1(boolean estat){this.accManual1=estat;};
    public void setAccManual2(boolean estat){this.accManual2=estat;};
    public void setAccManual3(boolean estat){this.accManual3=estat;};
    public void setAccManual4(boolean estat){this.accManual4=estat;};
    public void setAccFinestra(boolean estat){this.accFinestra=estat;};
    public void setAccVent(boolean estat){this.accVent=estat;};
    public void setAccValve(boolean estat){this.accValve=estat;};
    public void setAccAlarm(boolean estat){this.accAlarm=estat;};

    /**Es mostra una petita ajuda, a l'estil del 'man' del sistema UNIX, per informar de les opcions
     * disponibles des de consola.
     */
    public void mostrarOpcions(){
            System.out.println("SCAFIA - Error en la introducció de comanda en línia");
            System.out.println("\n# Format de comanda: \tjava scafia [opció 1] [opció 2] ...");
            System.out.println("\n# Opcions de control: \n\t\t\t[-info] [-iniciLog] [-h [iD]] [-obrir on/off] [-vent on/off] [-valve on/off] [-alarm on/off]");
            System.out.println("\n# Opcions de configuració del sistema:");
            System.out.println("\t\t\t[-t <segons>] [-tMax <temp.>] [-tMin <temp.>] [-hMax <humitat>] [-hMin <humitat>]");
            System.out.println("\t\t\t[-tMaxL <temp.>] [-tMinL <temp.>] [-hMaxL <humitat>] [-hMinL <humitat>] ");
            System.out.println("\t\t\t[-batt <V>] [-llum <nivell>] [-folderLog <nom>] [-folderStats <nom>] [-source <sf@IP:port>]");
            System.out.println("\n\tAtenció!\r\tLes opcions de configuració del sistema fan necessari el reinici de la consola de missatges per carregar els nous valors.");
            System.out.println("\n# On:");
            System.out.println("\t[-info] -> Mostra els valors actuals del fitxer de configuració del sistema. Es mostren més valors dels que es poden ");
            System.out.println("\t\tmodificar amb la línia de comandes que es presenta, i que corresponen a configuracions de seguretat o inicials del sistema.");
            System.out.println("\n\t[-iniciLog] -> Mostra en consola els valors actuals de lectures als sensors del nodes, alarmes i altres informacions del sistema.");
            System.out.println("\n\t[-h [iD]] -> Envia un missatge al node especificat per comprovar si està en línia.");
            System.out.println("\t\t[iD] ha de contenir el node d'interès (de 0 a 65534), si el seu valor es deixa en blanc s'envia a tots els nodes de la xarxa.");
            System.out.println("\n\t[-obrir on/off]] -> Envia un missatge al node de control d'accionaments per aplicar l'acció descrita sobre les finestres.");
            System.out.println("\t\ton: obre les finestres \toff: tanca les finestres.");
            System.out.println("\n\t[-vent on/off]] -> Envia un missatge al node de control d'accionaments per aplicar l'acció demanada sobre els electroventiladors.");
            System.out.println("\t\ton: activa els electroventiladors \toff: desactiva els electroventiladors.");
            System.out.println("\n\t[-valve on/off]] -> Envia un missatge al node de control d'accionaments per aplicar l'acció demanada sobre les electrovàlvules de pas d'aigua \n\t\t al sistema de
polvorització.");
            System.out.println("\t\ton: activa les electrovàlvules \toff: desactiva les electrovàlvules.");
            System.out.println("\n\t[-alarm on/off]] -> Envia un missatge al node de control d'accionaments per aplicar l'acció demanada sobre els sistema d'alarma del sistema.");
            System.out.println("\t\ton: activa l'alarma \toff: desactiva l'alarma.");
            System.out.println("\n# Utilització de les opcions de configuració del sistema");
            System.out.println("\tEl següents valors es poden canviar en qualsevol moment, i es faran efectius immediatament, si es fa un reset de la consola de missatges o bé en la següent
recepció d'un missatge de telemetria.");
            System.out.println("\tEs recomana fixar-los abans d'iniciar l'operació del sistema, per exemple afegint després l'opció -iniciLog al final de la línia de comandes, si es volen canviar i
iniciar de nou la gravació de valors.");
            System.out.println("\n\t[-t <segons>] -> Canvia la periodicitat, de lectures de sensors, als segons especificats.");
            System.out.println("\t\tPer disseny el valor de lectura per defecte està fixat a 30 segons. El màxim són 7200 segons (2 hores) i el mínim 5 segons.");
            System.out.println("\n\t[-tMax <temp.>] -> Fixa el valor màxim en ºC de la temperatura operativa del sistema.");
            System.out.println("\t\tEl seu valor dependrà del límit fixat al sistema (valor tMaxL).");
            System.out.println("\n\t[-tMin <temp.>] -> Fixa el valor mínim en ºC de la temperatura operativa del sistema.");
            System.out.println("\t\tEl seu valor dependrà del límit fixat al sistema (valor tMinL).");
            System.out.println("\n\t[-hMax <humitat>] -> Fixa el valor màxim en % de l'humitat relativa operativa del sistema.");
            System.out.println("\t\tEl seu valor dependrà del límit fixat al sistema (valor hMaxL).");
            System.out.println("\n\t[-hMin <humitat>] -> Fixa el valor mínim en % de l'humitat relativa operativa del sistema.");
            System.out.println("\t\tEl seu valor dependrà del límit fixat al sistema (valor hMinL).");
            System.out.println("\n\t[-tMaxL <temp.>] -> Fixa el valor màxim en ºC de la temperatura del sistema.");
            System.out.println("\t\tEl seu valor ha de ser el més gran dels valors de temperatura fixats al sistema.");
            System.out.println("\n\t[-tMinL <temp.>] -> Fixa el valor mínim en ºC de la temperatura del sistema.");
            System.out.println("\t\tEl seu valor ha de ser el més petit dels valors de temperatura fixats al sistema.");
            System.out.println("\n\t[-hMaxL <humitat>] -> Fixa el valor màxim en % de l'humitat relativa del sistema.");
            System.out.println("\t\tEl seu valor ha de ser el més gran dels valors d'humitat fixats al sistema.");
            System.out.println("\n\t[-hMinL <humitat>] -> Fixa el valor mínim en % de l'humitat relativa del sistema.");
            System.out.println("\t\tEl seu valor ha de ser el més petit dels valors d'humitat fixats al sistema.");
            System.out.println("\n\t[-batt <V>] -> Fixa el nivell de tensió d'alarma als nodes sensors. Per defecte el seu valor és 2.7V");
            System.out.println("\t\tEl seu valor ha d'estar entre 2.7 i 3.");
            System.out.println("\n\t[-llum <V>] -> Fixa el nivell de llum que indica si és dia o nit. Per defecte el seu valor és 50.");
            System.out.println("\t\tEl seu valor ha d'estar entre 0 i 200");
            System.out.println("\n\t[-folderLog <nom>] -> Fixa el nom de la carpeta on s'emmagatzemen els fitxers diari de sistema (log).");
            System.out.println("\t\tEl seu valor només accepta caràters alfabètics del sistema anglès.");
            System.out.println("\n\t[-folderStats <nom>] -> Fixa el nom de la carpeta on s'emmagatzemen els fitxers per a estadístiques del sistema.");
            System.out.println("\t\tEl seu valor només accepta caràters alfabètics del sistema anglès.");
            System.out.println("\n\t[-source <sf@IP:port>] -> Fixa la ruta de comunicació entre el sistema SCAFIA i la xarxa de nodes.");
            System.out.println("\t\tEl seu valor ha de seguir el format sf@IP:port. Per defecte el valor és sf@LocalHost:9002. Els ports vàlids són entre 1024 i 65535");
            System.out.println("\n# Exemples:");
            System.out.println("\tjava scafia -h 1 \tEnvia un missatge de comprovació al node remot 1.");
            System.out.println("\tjava scafia -h\t\tEnvia un missatge de comprovació a tots els nodes de la xarxa.");
            System.out.println("\tjava scafia -vent on\tActiva els electroventiladors de la instal·lació.");
            System.out.println("\tjava scafia -t 30 \tModifica el cicle de lectures a 30 segons.");
            System.out.println("\tjava scafia -tMax 45 \tModifica la temperatura màxima operativa a 45ºC.");
            System.out.println("\n# Fi de l'ajuda de consola SCAFIA");
    }
    /**Agrupació de missatges d'informació o d'error del sistema.
     *
     * @param opcio
     * @param valor
     */
    public void liniaLog(int opcio,String valor){
            String linia=null;
            switch (opcio){
            case 1: linia=" Canvi de valor a variable de control del temps de lectura a sensors (-t), valor = "+valor;break;
            case 2: linia=" Canvi de valor a variable de control de temperatura màxima (tMax), valor = "+valor;break;
            case 3: linia=" Canvi de valor a variable de control de temperatura mínima (tMin), valor = "+valor;break;
            case 4: linia=" Canvi de valor a variable de control d'humitat màxima (hMax), valor = "+valor;break;
            case 5: linia=" Canvi de valor a variable de control d'humitat mínima (hMin), valor = "+valor;break;
            case 6: linia=" Canvi de valor a variable de control de temperatura màxima límit (tMaxL), valor = "+valor;break;
            case 7: linia=" Canvi de valor a variable de control de temperatura mínima límit (tMinL), valor = "+valor;break;
```

```
        case 8: linia=" Canvi de valor a variable de control d'humitat màxima límit (hMaxL), valor = "+valor;break;
        case 9: linia=" Canvi de valor a variable de control d'humitat mínima límit (hMinL), valor = "+valor;break;
        case 10: linia=" Canvi de valor límit de tensió de bateria per avisar d'alarma (-batt), valor = "+valor;break;
        case 11: linia=" Canvi de valor límit de lluminositat que defineix el canvi dia/nit (-llum), valor = "+valor;break;
        case 12: linia=" Canvi de carpeta per emmagatzemar logs del sistema (-folderLog), valor = "+valor;break;
        case 13: linia=" Canvi de carpeta per emmagatzemar estadístiques del sistema (-folderStats), valor = "+valor;break;
        case 14: linia=" Canvi de comunicació entre sistema i xarxa de nodes (-source), valor = "+valor;break;
        case 15: linia=" S'ha demanat l'estat actual de les variables de configuració del sistema (-info)";break;
        case 16: linia=" S'ha demanat l'inici de registre de telemetria i accions del sistema (-iniciLog)";break;
        case 17: linia=" Error: nom de carpeta no vàlid. Al nom només s'accepten caràcters alfabètics en anglès. Valor introduït ("+valor+")";break;
        case 18: linia=" Error: el valor introduït de (-source) no és vàlid. Valor introduït ("+valor+")";break;
        case 19: linia=" Error: el valor de tensió introduït (-batt) no és valid. Valor introduït ("+valor+")";break;
        case 20: linia=" Error: el valor de control de lluminositat (-llum) no és valid. Valor introduït ("+valor+")";break;
        case 21: linia=" Error: el valor del cicle de lectura (-t) només pot estar entre 5 i 7200 segons (2 hores). Valor introduït ("+valor+")";break;
        case 22: linia=" S'envia el missatge de comprovació al node amb identificador (iD) "+valor;break;
        case 23: linia=" Error: el valor d'identificador (iD) no és vàlid. S'envia el missatge de comprovació a tots els nodes. Valor introduït ("+valor+")";break;
        case 24: linia=" Error: el valor de temperatura no és vàlid. Valor introduït ("+valor+")";break;
        case 25: linia=" Error: el valor d'humitat no és vàlid. Valor introduït ("+valor+")";break;
        case 26: linia=" Error: falta el valor del temps (-t) a configurar.";break;
        case 27: linia=" Error: falta el valor de temperatura a configurar.";break;
        case 28: linia=" Error: falta el valor d'humitat a configurar.";break;
        case 29: linia=" Error: falta indicar un valor de tensió vàlid per la bateria (-batt).";break;
        case 30: linia=" Error: falta indicar el valor de lluminositat (-llum).";break;
        case 31: linia=" Error: falta un nom de carpeta vàlid.";break;
        case 32: linia=" Error: falta una ruta d'accès vàlida (-source).";break;
        case 33: linia=" Inici de sessió de consola de control del sistema. Paràmetres introduïts: "+valor;break;
        case 34: linia=" Finalitza sessió de consola de control del sistema.";break;
        case 35: linia=" Error: falta indicar l'acció a realitzar(on/off) amb l'accionament (-obrir/-vent/-valve/-alarm).Valor introduït ("+valor+")";break;
        }
        linia="SCAFIA CON "+getStamp(1)+linia;
        registrarMov(linia,1);
    }


    /**Comprova si la dada passada com a argument te un nom de directori vàlid. Només s'accepten valors alfabètics
     * de l'alfabet anglès. Torna cert si el nom és vàlid.
     *
     * @param valor
     * @return
     */
    public boolean checkFolder(String valor){
        boolean token=true;
        int i=0;
        String valid="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUWXYZ";
        while (token && i<valor.length()){
            if(valid.contains(Character.toString(valor.charAt(i)))){
                i++;
            }else{
                liniaLog(17,valor);
                token=false;
            }
        }
        return token;
    }
    /**Comprova que la dada passada com camí de comunicació xarxa-sistema és correcta. Només es comprova que
     * contingui la següent estructura sf@IP:port, on port vàlid és entre 1024 i 65535.
     *
     * @param valor
     * @return
     */
    public boolean checkSource(String valor){
        String ip=null;
        String port=null;
        int portnum=0;
        boolean check=true;
        if(valor.contains("sf@") && valor.contains(":")){
            ip=valor.substring(3, valor.indexOf(':'));
            port=valor.substring(valor.indexOf(':')+1,valor.length());
            portnum=Integer.parseInt(port.trim());
            if (ip==null || port==null || (portnum<1025 || portnum>65535)){
                check=false;
            }
        }else{
            check=false;
        }
        if (!check){
            liniaLog(18,valor);
        }
        return check;
    }

    /**Comprova si el valor introduït de control sobre el límit de bateria és correcte.
     * Torna cert si aquest està entre 2 i 3 volts
     *
     * @param valor
     * @return
     */
    public boolean checkBatt(String valor){
        boolean check = false;
        double volts=0;
        try{
            volts = Double.parseDouble(valor.replace(',','.'));
            if (volts > 2.7 && volts < 3){
                check=true;
            }else{
                liniaLog(19,valor);
            }
        }catch (Exception e){
```

```
                liniaLog(19,valor);
                check=false;
        }
        return check;
}

/**Comprova si el valor passat per paràmetre s'ajusta las llindars del sensor de llum, que són entre 0 i 1023.
 *
 * @param valor
 * @return
 */
public boolean checkLlum(String valor){
        boolean check = false;
        try{
                int llum = Integer.parseInt(valor);

                if (llum >= 0 && llum <= 200){
                        check=true;
                }else{
                        liniaLog(20,valor);
                }
        }catch (Exception e){
                liniaLog(20,valor);
                check=false;
        }
        return check;
}

/**Comprova si la dada passada com argument compleix els límits de disseny (5 a 7200 segons)
 * Torna cert si el valor és vàlid.
 * @param valor
 * @return
 */
public boolean checkTiming(String valor){
        int segons = 0;
        boolean check = false;
        try{
                segons = Integer.parseInt(valor.trim());
                if (segons<5 || segons > 7200){
                        liniaLog(21,valor);
                        check=false;
                }else{
                        check=true;
                }
        }catch (Exception e){
                liniaLog(21,valor);
        }
        return check;
}

/**Comprova la dada passada com a paràmetre que correspongui a un Id de node vàlid. Els valors possibles són de 1 a 255.
 * Torna el valor de la mote si el valor és vàlid.
 * @param valor
 * @return
 */
public int checkHello(String valor){
        int mote = -1;
        try{
                mote = Integer.parseInt(valor.trim());
                if (mote >= 0 && mote < 65536){
                        liniaLog(22,valor);
                }else{
                        mote=65535;
                        liniaLog(23,valor);
                }
        }catch (Exception e){
                liniaLog(23,valor);
                mote=65535;
        }
        return mote;
}

/**Comprova la dada passada com a paràmetre que correspongui a un valor vàlid. Ha de ser >tMin i <TmaxL
 * Torna cert si el valor és vàlid.
 * @param valor
 * @return
 */
public boolean checkTMax(String valor){
        try{
                int temp = Integer.parseInt(valor.trim());
                if (temp<Integer.parseInt(getTmin()) || temp > Integer.parseInt(getTmaxl())){
                        liniaLog(24,valor);
                        return false;
                }else{
                        return true;
                }
        }catch (Exception e){
                liniaLog(24,valor);
                return false;
        }
}

/**Comprova la dada passada com a paràmetre que correspongui a un valor vàlid. Ha de ser >tMinl i <Tmax
 * Torna cert si el valor és vàlid.
 * @param valor
 * @return
 */
public boolean checkTMin(String valor){
        try{
```

```
                int temp = Integer.parseInt(valor.trim());
                if (temp<Integer.parseInt(getTminl()) || temp > Integer.parseInt(getTmax())){
                        liniaLog(24,valor);
                        return false;
                }else{
                        return true;
                }
        }catch (Exception e){
                liniaLog(24,valor);
                return false;
        }
}

/**Comprova la dada passada com a paràmetre que correspongui a un valor vàlid. Ha de ser >tMin, >Tminl i >Tmax
 * Torna cert si el valor és vàlid.
 * @param valor
 * @return
 */
public boolean checkTMaxL(String valor){
        try{
                int temp = Integer.parseInt(valor.trim());
                if (temp < Integer.parseInt(getTminl()) || temp < Integer.parseInt(getTmin()) || temp < Integer.parseInt(getTmax())){
                        liniaLog(24,valor);
                        return false;
                }else{
                        return true;
                }
        }catch (Exception e){
                liniaLog(24,valor);
                return false;
        }
}

/**Comprova la dada passada com a paràmetre que correspongui a un valor vàlid. Ha de ser <tMin, <Tmax i <Tmaxl
 * Torna cert si el valor és vàlid.
 * @param valor
 * @return
 */
public boolean checkTMinL(String valor){
        try{
                int temp = Integer.parseInt(valor.trim());
                if (temp > Integer.parseInt(getTmin()) || temp > Integer.parseInt(getTmax()) || temp > Integer.parseInt(getTmaxl())){
                        liniaLog(24,valor);
                        return false;
                }else{
                        return true;
                }
        }catch (Exception e){
                liniaLog(24,valor);
                return false;
        }
}

/**Comprova la dada passada com a paràmetre que correspongui a un valor vàlid. Ha de ser >HMin i <HmaxL
 * Torna cert si el valor és vàlid.
 *
 * @param valor
 * @return
 */
public boolean checkHMax(String valor){
        try{
                int hum = Integer.parseInt(valor.trim());
                if (hum<Integer.parseInt(getHmin()) || hum > Integer.parseInt(getHmaxl())){
                        liniaLog(25,valor);
                        return false;
                }else{
                        return true;
                }
        }catch (Exception e){
                liniaLog(25,valor);
                return false;
        }
}

/**Comprova la dada passada com a paràmetre que correspongui a un valor vàlid. Ha de ser >Hminl i <HMax
 * Torna cert si el valor és vàlid.
 * @param valor
 * @return
 */
public boolean checkHMin(String valor){
        try{
                int hum = Integer.parseInt(valor.trim());
                if (hum<Integer.parseInt(getHminl()) || hum > Integer.parseInt(getHmax())){
                        liniaLog(25,valor);
                        return false;
                }else{
                        return true;
                }
        }catch (Exception e){
                liniaLog(25,valor);
                return false;
        }
}

/**Comprova la dada passada com a paràmetre que correspongui a un valor vàlid. Ha de ser >HMinl, >HMin i >Hmax
 * Torna cert si el valor és vàlid.
 *
 * @param valor
 * @return
```

```
 */
public boolean checkHMaxL(String valor){
	try{
		int hum = Integer.parseInt(valor.trim());
		if (hum < Integer.parseInt(getHminl()) || hum < Integer.parseInt(getHmin()) || hum < Integer.parseInt(getHmax())){
			liniaLog(25,valor);
			return false;
		}else{
			return true;
		}
	}catch (Exception e){
		liniaLog(25,valor);
		return false;
	}
}

/**Comprova la dada passada com a paràmetre que correspongui a un valor vàlid. Ha de ser >Hminl i <HMax
 * Torna cert si el valor és vàlid.
 * @param valor
 * @return
 */
public boolean checkHMinL(String valor){
	try{
		int hum = Integer.parseInt(valor.trim());
		if (hum > Integer.parseInt(getHmin()) || hum > Integer.parseInt(getHmax()) || hum > Integer.parseInt(getHmaxl())){
			liniaLog(25,valor);
			return false;
		}else{
			return true;
		}
	}catch (Exception e){
		liniaLog(25,valor);
		return false;
	}
}


/**Torna els valors actuals del fitxer de configuració del sistema
 *
 */
public void printINIValues(){
	System.out.println("=====================================");
	System.out.println("SCAFIA - Valors de configuració actuals");
	System.out.println("=====================================");
	System.out.println(getStamp(1));
	System.out.println("\r[Valors operatius]");
	System.out.println("\tTemperatura màxima (tMax): "+getTmax()+"ºC");
	System.out.println("\tTemperatura mínima (tMin): "+getTmin()+"ºC");
	System.out.println("\tHumitat màxima (hMax): "+getHmax()+"%");
	System.out.println("\tHumitat mínima (hMin): "+getHmin()+"%");
	System.out.println("\tTemps de lectura de sensors (t): "+getTemps()+" segons");
	System.out.println("\r[Valors límit del sistema]");
	System.out.println("\tTemperatura màxima (tMaxL): "+getTmaxl()+"ºC");
	System.out.println("\tTemperatura mínima (tMinL): "+getTminl()+"ºC");
	System.out.println("\tHumitat màxima (hMaxL): "+getHmaxl()+"%");
	System.out.println("\tHumitat mínima (hMinL): "+getHminl()+"%");
	System.out.println("\tLluminositat canvi nit/dia (llum): "+getLlum());
	System.out.println("\tBateria límit en node (batt): "+getBatt()+"Vcc");
	System.out.println("\r[Altres valors del sistema]");
	System.out.println("\tCarpeta de logs del sistema: "+getfolderLog());
	System.out.println("\tCarpeta d'estadístiques: "+getfolderStats());
	System.out.println("\tRuta de comunicació sistema-xarxa de nodes: "+getSource());
	System.out.println("");
}



/**Analitza l'estat d'alarmes del sistema per si s'ha d'indicar a la consola la seva existència.
 *
 */
public String flagAlarm(){
	String flagA=null;
	if (getFlagAlarm()){
		flagA = "SCAFIA A! ";
	}else{
		flagA = "SCAFIA - ";
	}
	return flagA;
}

public boolean getFlagGAlarm(){
	if (alarmGTemp||alarmGHum||alarmNet){
		return true;
	}else{
		return false;
	}
}

/**Conversió de valor de la tensió de referència del sensor de batería a un valor normalitzat
 * Torna el valor normalitzat.
 *
 * @param vrefMilivolts
 * @param counts
 * @return
 */
public double passarAVolts(int vrefMilivolts, int counts){
	return (((double)counts)/1024)*(((double)vrefMilivolts)/1000);
}
```

```
    /**Funció que torna el valor, passat per paràmetre, a un valor normalitzat #0.00
     *
     * @param d
     * @return
     */
    public String formatejarDouble(double d){
            DecimalFormat df = new DecimalFormat("#0.00");
            return df.format(d);
    }
}
```

D.3 Mota remota (mS)

D.3.1 InitC.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: InitC.nc
//
// Aquest component només marca amb els leds del node
// que aquest s'ha iniciat.

configuration InitC {}
implementation
{
  components MainC, InitP, LedsC;
  components HallC, MoteIdC, SensorC;

  components new TimerMilliC() as Timer0;

  InitP -> MainC.Boot;

  InitP.Timer0 -> Timer0;
  InitP.Leds -> LedsC;
}
```

D.3.2 InitP.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: InitP.nc
//
//
// Aquest component només marca amb els leds del node
// que aquest s'ha iniciat.


#include "Timer.h"

module InitP
{
  uses interface Timer<TMilli> as Timer0;
  uses interface Leds;
  uses interface Boot;
}

implementation

{
   uint8_t i=0;

    event void Boot.booted()
     {
            call Timer0.startOneShot(100);
     }


//Quan s'engega la mote ho indiquem amb una seqüència de colors (cada 1/3 segons aprox.):
// Seqüència: Verd -> Verd + Groc -> Verd + Groc + Vermell -> Verd i vermell -> Verd

  event void Timer0.fired()  {
   i++;
   if(i==1){
          call Leds.led2On();
          call Timer0.startOneShot(300);
   }else if(i==2){
                  call Leds.led1On();
                  call Timer0.startOneShot(300);
      }else if(i==3){
                        call Leds.led0On();
                        call Timer0.startOneShot(300);
         }else if(i==4){
                  call Leds.led0Off();
                  call Leds.led1Off();
                  call Leds.led2Off();
                        call Timer0.startOneShot(300);
                   }else if(i==5){
                        call Leds.led0On();
                  call Leds.led2On();
                   }
   }
}
```

D.3.3 HallC.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: HallC.nc
//
// Aquest component fa la detecció d'activitat al sensor Hall i l'envia
// via ràdio al sistema.



configuration HallC{}
implementation{
   components MainC,HallP,LedsC;
   components new TimerMilliC() as Timer;
   components HplAtm128GeneralIOC as IO;
   components HplAtm128InterruptC as IOInt;
   components ActiveMessageC as SerialActiveMessageHallC;
   components new AMSenderC(AM_HALLMSG) as AMSenderHallC;

   HallP.HallPinIn -> IO.PortE7;
   HallP.HallPinInterrupt -> IOInt.Int7;

   HallP.Boot -> MainC;
   HallP.TimerInici -> Timer;

   HallP.PowerSensor -> IO.PortE3;
   HallP.Leds -> LedsC;

   HallP.AMSend -> AMSenderHallC;
   HallP.Packet -> AMSenderHallC;
   components ActiveMessageC;
   HallP.SplitControl -> ActiveMessageC;
   HallP.PacketAcknowledgements->ActiveMessageC;

   components LocalIeeeEui64C;
   HallP.LocalIeeeEui64 -> LocalIeeeEui64C;

}
```

D.3.4 HallP.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: HallP.nc
//
// Aquest component fa la detecció d'activitat al sensor Hall i l'envia
// via ràdio al sistema.

#include "AuxVar.h"

module HallP{
   uses interface Boot;
   uses interface Timer<TMilli> as TimerInici;
   uses interface GeneralIO as PowerSensor;
   uses interface HplAtm128Interrupt as HallPinInterrupt;
   uses interface GeneralIO as HallPinIn;
   uses interface Leds;
   uses interface Packet;
   uses interface AMSend;
   uses interface SplitControl;
   uses interface LocalIeeeEui64;
   uses interface PacketAcknowledgements;
}
implementation{

   task void sendMessage();
   message_t packet;
   HallMsg* data;
   uint8_t inici=0;
   uint16_t sinkMote=0; //Adreça del node base

   event void Boot.booted(){

         data=(HallMsg*) call Packet.getPayload(&packet,sizeof(HallMsg));
         data->moteId=TOS_NODE_ID; //Carreguem la id del node al registre

         call SplitControl.start();

         // Es configura com a entrada el pin on està el sensor Hall
         call HallPinIn.makeInput();
         call HallPinIn.clr();

         // Es fa detecció de la interrupció en el flanc de pujada. S'ha de resetejar abans d'activar
         call HallPinInterrupt.edge(TRUE);
         call HallPinInterrupt.clear();
         call HallPinInterrupt.enable();

         //S'inicia un temporitzador per estabilitzar el sistema
```

```
        call TimerInici.startOneShot(5000);
    }

    event void TimerInici.fired(){
        //S'activa la detecció al pin i es marca apagant el Led 0 (vermell)
        call PowerSensor.set();
        call Leds.led0Toggle();
    }

    //Quan es produeix una detecció al sensor Hall s'envia el missatge
    async event void HallPinInterrupt.fired(){
        if (inici==1){
            call Leds.led0Toggle();
            call Leds.led0Toggle();
            post sendMessage();}
        atomic inici=1;
    }

    //Enviament del missatge tipus HallMsg corresponent a la detecció en el sensor
    task void sendMessage(){
//      if(call AMSend.send(AM_BROADCAST_ADDR, &packet, sizeof(HallMsg))==SUCCESS){
        call PacketAcknowledgements.requestAck(&packet);
        if(call AMSend.send(sinkMote, &packet, sizeof(HallMsg))==SUCCESS){
            call Leds.led1Toggle();
            call Leds.led1Toggle();
        }else{
            post sendMessage();
        }
    }

    event void SplitControl.stopDone(error_t error){}

    event void SplitControl.startDone(error_t error){}
    event void AMSend.sendDone(message_t *msg, error_t error){   }

}
```

D.3.5 MoteIdC.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: MoteIdC.nc
//
// Aquest component envia els valors TOS_NODE_ID i
// adreça MAC quan es pulsa el botó d'usuari del node.

#include "AuxVar.h"

configuration MoteIdC{
}
implementation{

    components MoteIdP,MainC;

//  components ActiveMessageC as SerialActiveMessageC;
    components ActiveMessageC;
    MoteIdP.SplitControl -> ActiveMessageC;
    MoteIdP.PacketAcknowledgements->ActiveMessageC;
    components new AMSenderC(AM_MOTEMSG) as AMSenderC;
    MoteIdP.AMSend -> AMSenderC;
    MoteIdP.Packet -> AMSenderC;

    components HplAtm128GeneralIOC as IO;
    components HplAtm128InterruptC as IOInt;
    components LedsC;

    MoteIdP.Leds -> LedsC;
    MoteIdP.ButtonPin -> IO.PortE7;
    MoteIdP.Button -> IOInt.Int6;
    MoteIdP.Boot-> MainC;

    components LocalIeeeEui64C;
    MoteIdP.LocalIeeeEui64 -> LocalIeeeEui64C;
}
```

D.3.6 MoteIdP.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: MoteIdP.nc
//
// Aquest component envia els valors TOS_NODE_ID i
// adreça MAC quan es pulsa el botó d'usuari del node.

#include "AuxVar.h"
```

```
module MoteIdP{
   uses interface HplAtm128Interrupt as Button;
   uses interface Boot;
   uses interface Packet;
   uses interface AMSend;
   uses interface Leds;
   uses interface GeneralIO as ButtonPin;
   uses interface SplitControl;
   uses interface LocalIeeeEui64;
   uses interface PacketAcknowledgements;
}
implementation{

   task void sendMessage();
   message_t packet;
   MoteMsg* data;
   uint16_t sinkMote=0; //Adreça node base

   event void Boot.booted(){
         ieee_eui64_t mac;
         uint8_t i;

         //Obtenim la mida del missatge
         data = (MoteMsg*) call Packet.getPayload(&packet, sizeof(MoteMsg));

         //S'obte la identificacio de node
         data->moteId = TOS_NODE_ID;

         //S'obte el valor de la MAC i s'insereix al registre
         mac = call LocalIeeeEui64.getId();
         for(i=0;i<8;i++){
               data->macAddr[i]=mac.data[i];
         }

         call SplitControl.start();

         // Configurem el boto d'usuari per a que emeti els valors d'identificacio
         call ButtonPin.makeInput();
         call ButtonPin.clr();

         // Configura el boto com a interrupcio de pujada (flanc)
         // Es fa un reset abans d'activar-lo
         call Button.edge(TRUE);
         call Button.clear();
         call Button.enable();
   }

   //Al fer la deteccio del boto s'envia el missatge amb l'Id de la mota
   async event void Button.fired(){
         post sendMessage();
   }

   //Enviem el missatge de detecció de polsació del botó
   task void sendMessage(){
         call PacketAcknowledgements.requestAck(&packet);
         if(call AMSend.send(sinkMote, &packet, sizeof(MoteMsg))==SUCCESS){
               call Leds.led1Toggle();
               call Leds.led1Toggle();
         }else{
               call Leds.led0Toggle();
               call Leds.led0Toggle();
               post sendMessage();
         }
   }

   event void SplitControl.stopDone(error_t error){}
   event void SplitControl.startDone(error_t error){}
   event void AMSend.sendDone(message_t *msg, error_t error){}
}
```

D.3.7 SensorC.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: SensorC.nc
//
//Aquest component llegeix els sensors del node, complint un cicle
//de duració programada, que pot ser reprogramat i els envia al sistema
// per al seu registre. A més avisa de la seva activació al sistema.


#include "AuxVar.h"

configuration SensorC{

}
implementation{
   components MainC, SensorP;
   components new TimerMilliC() as Timer;
   components new TimerMilliC() as TimerStabilize;
   components new TimerMilliC() as TimerInit;
```

```
    components HplAtm128GeneralIOC as IO;
    components LedsC as LedsC;

    SensorP.Leds -> LedsC;
    SensorP.Boot -> MainC;
    SensorP.Timer -> Timer;
    SensorP.TimerStabilize -> TimerStabilize;
    SensorP.TimerInit -> TimerInit;

    components new AdcReadClientC() as LectorADC;
    SensorP.LecturaAdc -> LectorADC;
    LectorADC.Atm128AdcConfig -> SensorP.Atm128AdcConfig;

    components ActiveMessageC as SerialActiveMessageC;
    components new AMSenderC(AM_SENSORSMSG) as SerialAMSenderC;
    components new AMSenderC(AM_HELLOMSG) as SerialAMSenderHelloC;
    components new AMReceiverC(AM_HELLOMSG) as SerialAMReceiverHelloC;
    components new AMSenderC(AM_SETTINGS) as SerialAMSenderSettingsC;
    components new AMReceiverC(AM_SETTINGS) as SerialAMReceiverSettingsC;

    SensorP.Packet -> SerialAMSenderC;
    SensorP.AMSend -> SerialAMSenderC;
    SensorP.PacketHello -> SerialAMSenderHelloC;
    SensorP.AMSendHello -> SerialAMSenderHelloC;
    SensorP.PacketSettings -> SerialAMSenderSettingsC;
    SensorP.AMSendSettings -> SerialAMSenderSettingsC;
    SensorP.powerSerie -> SerialActiveMessageC;
    SensorP.PacketAcknowledgements -> SerialActiveMessageC;

    SensorP.SinkPowerSensors -> IO.PortG1;
    SensorP.ReceiveHello -> SerialAMReceiverHelloC.Receive;
    SensorP.ReceiveSettings -> SerialAMReceiverSettingsC.Receive;
}
```

D.3.8 SensorP.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: SensorP.nc
//
//Aquest component llegeix els sensors del node, complint un cicle
//de duració programada, que pot ser reprogramat i els envia al sistema
// per al seu registre. A més avisa de la seva activació al sistema.

#include "AuxVar.h"

module SensorP{
    uses interface Boot;
    uses interface Timer<TMilli> as Timer;
    uses interface Timer<TMilli> as TimerStabilize;
    uses interface Timer<TMilli> as TimerInit;
    uses interface GeneralIO as SinkPowerSensors;

    uses interface Read<uint16_t> as LecturaAdc;
    provides interface Atm128AdcConfig;

    uses interface SplitControl as powerSerie;
    uses interface Packet;
    uses interface AMSend;
    uses interface AMSend as AMSendHello;
    uses interface AMSend as AMSendSettings;
    uses interface Receive as ReceiveHello;
    uses interface Receive as ReceiveSettings;
    uses interface Packet as PacketHello;
    uses interface Packet as PacketSettings;
    uses interface Leds;
    uses interface PacketAcknowledgements;
}
implementation{

    enum{
        INITIALIZING_COMM,
        READY,
        WARMING_UP,
        WARMED_UP,
        READING_BAT,
        BAT_READ,
        READING_PHOTO,
        PHOTO_READ,
        READING_TEMP,
        TEMP_READ,
        READING_HUM,
        HUM_READ,
        SENSORS_READ,
        DATA_SENT,
        ERROR
    };
    message_t packet;
    task void seguentPas();
    void _turnOnSensors();
    void _turnOffSensors();
    task void sendValuesSensors();
```

```
task void sayHello();
task void blinkFail();
task void blinkSend();
uint8_t estat=READY;
SensorsMsg* data;
Settings* timing;
uint32_t timeset;
uint16_t sinkMote=0; //Adreça del node base

event void Boot.booted(){
	//A l'inici s'apaguen els sensors
	_turnOffSensors();

	//Es carrega el valor per defecte de cicle de lectura. Serà un cicle
	//de 30 segons.
	timing = (Settings*) call Packet.getPayload(&packet, sizeof(Settings));
	timing->moteId=TOS_NODE_ID;
	timing->setmilitimer=30000;
	timeset=timing->setmilitimer;

	atomic estat=INITIALIZING_COMM;
	call powerSerie.start();

	//iniciem el timer que fara que es llegeixin dels sensors
	call Timer.startPeriodic(timeset);
}

//S'activen els sensors del node
void _turnOnSensors(){
	call SinkPowerSensors.makeOutput();
	call SinkPowerSensors.clr();
}

//Se desactiven els sensors del node
void _turnOffSensors(){
	call SinkPowerSensors.makeInput();
	call SinkPowerSensors.set();
}

//El temporitzador de cicle marca quant llegirem els valors dels sensors
event void Timer.fired(){
	if(estat==READY){
		post seguentPas();
	}
}

// Seqüència d'accions de lectura dels sensors:
// 1- engegar sensors i esperar estabilitzarse
// 2- llegir bateria
// 3- llegir temperatura
// 4- llegir photo
// 5- llegir humitat
// 6- apagar sensors
// 7- enviar les dades dels sensors
// 8- preparat per iniciar de nou el cicle quan ho marqui el temporitzador

task void seguentPas(){
	switch(estat){
		case INITIALIZING_COMM:
			break;
		case READY:	// Pas 1
			data = (SensorsMsg*) call Packet.getPayload(&packet, sizeof(SensorsMsg));
			atomic estat= WARMING_UP;
			data->counter++;
			data->vrefmilivolts=2560;
			data->moteId = TOS_NODE_ID;
			_turnOnSensors();
			call TimerStabilize.startOneShot(100);
			break;
		case WARMED_UP:	// Pas 2
			atomic estat=READING_BAT;
			if(call LecturaAdc.read()!=SUCCESS){
				atomic estat=ERROR;
				post seguentPas();
			}
			break;
		case BAT_READ: // Pas 3
			atomic estat=READING_PHOTO;
			if(call LecturaAdc.read()!=SUCCESS){
				atomic estat=ERROR;
				post seguentPas();
			}
			break;
		case PHOTO_READ: // Pas 4
			atomic estat=READING_TEMP;
			if(call LecturaAdc.read()!=SUCCESS){
				atomic estat=ERROR;
				post seguentPas();
			}
			break;
		case TEMP_READ: // Pas 5
			atomic estat=READING_HUM;
			if(call LecturaAdc.read()!=SUCCESS){
				atomic estat=ERROR;
				post seguentPas();
			}
			break;
		case HUM_READ: // Pas 6
			_turnOffSensors();
```

```
                atomic estat = SENSORS_READ;
                post seguentPas();
                break;
            case SENSORS_READ: // Pas 7
                data->militimer = timeset;
                post sendValuesSensors();
                break;
            case DATA_SENT: // Pas 8
                post blinkSend();
            case ERROR:
            default:
                atomic estat=READY;
                _turnOffSensors();
                break;
        }
    }

    //Una vegada llegits tots els sensors s'envien els valors la node base
    task void sendValuesSensors(){
            call PacketAcknowledgements.requestAck(&packet);
            if(call AMSend.send(sinkMote, &packet, sizeof(SensorsMsg))!=SUCCESS){
                post blinkSend();
                atomic estat=ERROR;
                post seguentPas();
            }else{
                post blinkFail();
                post sendValuesSensors();
            }
    }


    //Si es produeix un error en la transmissió s'indica en el led vermell
    task void blinkFail(){
        call Leds.led0Toggle();
        call Leds.led0Toggle();
    }

    //L'enviament d'una dada es marca amb el LED groc
    task void blinkSend(){
        call Leds.led1Toggle();
        call Leds.led1Toggle();
    }

    //Quan el temporitzador d'estabilització s'activa, ens indica que els sensors ja estan preparats
    //Canviem el valor d'estat per continuar.
    event void TimerStabilize.fired(){
        if(estat==WARMING_UP){
            atomic estat = WARMED_UP;
        }
        else{
            atomic estat=ERROR;
        }
        post seguentPas();
    }

    //La lectura de cada ADC es passa al seu valor corresponent
    event void LecturaAdc.readDone(error_t result, uint16_t val){
        if(result==SUCCESS){
            if(estat==READING_BAT){
                data->countsBat = val;
                atomic estat=BAT_READ;
            }else if(estat == READING_PHOTO){
                data->countsPhoto = val;
                atomic estat=PHOTO_READ;
            }else if(estat== READING_TEMP){
                data->countsTemp = val;
                atomic estat=TEMP_READ;
            }else if(estat== READING_HUM){
                data->countsHum = val;
                atomic estat=HUM_READ;
            }else{      //aquest estat no es possible, pero s'ha de retornar alguna cosa
                atomic estat=ERROR;
            }
        }
        else{
            atomic estat= ERROR;
        }
        post seguentPas();
    }

    //Segons el punt del cicle de lectura, es llegeix del sensor que correspon
    async command uint8_t Atm128AdcConfig.getChannel(){
        if(estat==READING_BAT){
            return ATM128_ADC_SNGL_ADC0;
        }else if(estat == READING_PHOTO){
            return ATM128_ADC_SNGL_ADC1;
        }else if(estat== READING_TEMP){
            return ATM128_ADC_SNGL_ADC2;
        }else if(estat== READING_HUM){
            return ATM128_ADC_SNGL_ADC3;
        }else{      //aquest estat no es possible, pero s'ha de retornar alguna cosa
            return ATM128_ADC_SNGL_ADC0;
        }
    }


    async command uint8_t Atm128AdcConfig.getRefVoltage(){
        return  ATM128_ADC_VREF_2_56;
    }
```

```
async command uint8_t Atm128AdcConfig.getPrescaler(){
	return ATM128_ADC_PRESCALE;
}

event void powerSerie.startDone(error_t error){
	atomic estat=READY;
	call TimerInit.startOneShot(1500);
}

event void powerSerie.stopDone(error_t error){}

event void AMSend.sendDone(message_t *msg, error_t error){
	atomic estat=DATA_SENT;
	post seguentPas();
}

event void TimerInit.fired(){
	post sayHello();
}

message_t packetSerial;
uint8_t sayingHello = 0;
uint32_t numberHello=0;

//Enviem missatge de node iniciat, que inclou Id, comptador i temps de cicle de lectura
task void sayHello(){
	HelloMsg* m;
	if(sayingHello==0){
		atomic sayingHello=1;
		m = (HelloMsg*)call PacketHello.getPayload(&packetSerial, sizeof(HelloMsg));
		m->seqNum = numberHello;
		m->moteId = TOS_NODE_ID;
		m->militimer = timeset;
		call PacketAcknowledgements.requestAck(&packetSerial);
		if (call AMSendHello.send(sinkMote, &packetSerial, sizeof(HelloMsg))==SUCCESS){
			post blinkSend();
		}else{
			post blinkFail();
			post sayHello();
		}
	}
}

//Detecció de recepció d'un missatge HelloMsg adreçat al node directament
event message_t * ReceiveHello.receive(message_t *msg, void *payload, uint8_t len){
	if(sayingHello==0){
		HelloMsg* m = (HelloMsg*)payload;
		if((m->moteId == TOS_NODE_ID)||(m->moteId == 65535L)){
			numberHello = m->seqNum;
			post sayHello();
		}
	}
	return msg;
}

event void AMSendHello.sendDone(message_t *msg, error_t error){
	atomic sayingHello=0;
}

uint8_t chSettings=0;
uint32_t numberSettings=0;
task void confirmTimeChange();
task void setTimer();

//S'envia el valor de node, comptador i temps de lectura actual al node
task void confirmTimeChange(){
	Settings* s;
	if(chSettings==0){
		atomic chSettings=1;
		s = (Settings*)call PacketSettings.getPayload(&packetSerial, sizeof(Settings));
		s->moteId = TOS_NODE_ID;
		s->setmilitimer = timeset;
		s->seqNum = numberSettings;
		call PacketAcknowledgements.requestAck(&packetSerial);
		if(call AMSendSettings.send(sinkMote, &packetSerial, sizeof(Settings))==SUCCESS){
			post blinkSend();
		}else{
			post blinkFail();
			post confirmTimeChange();
		}
	}
}

//Es fa el canvi del cicle de lectura al nou temps que s'ha rebut al node
task void setTimer(){
	if(chSettings==0){
		call Leds.led0Toggle();
		call Timer.stop();
		call Timer.startPeriodic(timeset);
		post confirmTimeChange();
		call Leds.led0Toggle();
	}
}

//Es rep un missate Settings amb un nou valor de temps de cicle de lectura de sensors
event message_t * ReceiveSettings.receive(message_t *msg, void *payload, uint8_t len){
	if(chSettings==0){
		Settings* s = (Settings*)payload;
```

```
                if((s->moteId == TOS_NODE_ID)||(s->moteId == 65535L)){
                    timeset = s->setmilitimer;
                    numberSettings = s->seqNum;
                    post setTimer();
                }
            }
            return msg;
        }

        event void AMSendSettings.sendDone(message_t *msg, error_t error){
            atomic chSettings=0;
        }

    }
```

D.3.9 AuxVar.h

```
    //TFC ETIS 2n Semestre curs 2010-2011
    // Nom del Projecte: SCAFIA
    //
    // Consultor: Jordi Bécares Ferrés
    // @author Joan Carles Martínez Rodríguez
    // Fitxer: AuxVar.h

    #ifndef AUXVAR
    #define AUXVAR

    //Descripció dels diferents missatges
    // AM_SENSORSMSG   -> Lectures als sensors de node
    // AM_HELLOMSG             -> Inici de node
    // AM_MOTEMSG              -> Informació del node
    // AM_HALLMSG      -> Activació sensor Hall
    // AM_SETTINGS     -> Fixar cicle de lectura del sensors
    enum {
        AM_SENSORSMSG   = 40,
        AM_HELLOMSG             = 41,
        AM_MOTEMSG              = 42,
        AM_HALLMSG      = 43,
        AM_SETTINGS     = 44
    };

    typedef nx_struct SensorsMsg {
        nx_uint16_t moteId;     //Identificació del node
        nx_uint16_t countsTemp; //Valor de temperatura
        nx_uint16_t countsPhoto; //Valor de lluminositat
        nx_uint16_t countsBat; //Valor de bateria interna
        nx_uint16_t countsHum;      //Valor d'humitat
        nx_uint16_t vrefmilivolts; //Tensió de referència de la bateria
        nx_uint32_t militimer; //Valor del cicle de lectura de sensors
        nx_uint32_t counter;    //Comptador o seqüència interna
    } SensorsMsg;

    //Registre o estructura del missatge d'inici del node
    typedef nx_struct HelloMsg{
        nx_uint16_t moteId; //Identificació del node
        nx_uint32_t seqNum; //Identificador del missatge
        nx_uint32_t militimer; //Temps de cicle de lectura de sensors programat
    } HelloMsg;

    //Registre o estructura del missatge d'activació del sensor Hall del node
    typedef nx_struct HallMsg{
        nx_uint16_t moteId; //Identificació del node
    } HallMsg;

    //Registre o estructura del missatge amb id del node i la seva MAC
    //¡ que s'envia al pulsar el botó d'usuari
    typedef nx_struct MoteMsg{
        nx_uint16_t moteId; //Identificació del node
        nx_uint8_t macAddr[8]; //Identificació del node MAC_address
    }MoteMsg;

    //Registre o estructura del missatge de configuració del temps de cicle de lectura
    typedef nx_struct Settings{
        nx_uint16_t moteId; //Identificació del node
        nx_uint32_t setmilitimer; //Temps de cicle de lectura de sensors
        nx_uint32_t seqNum; //Comptador o seqüència interna
    }Settings;


    #endif /* AUXVAR */
```

D.4 Mota enllaç (mB)

D.4.1 ActionC.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: ActionC.nc
//
//Aquest component tradueix la senyalització d'activació/desactivació, que
//rep via sèrie la mote, en la simulació d¡activar o desactivar una sortida
//que controlaria un component de potència. En el nostre cas la simulació
//es realitza sobre el LED vermell (LED0), de forma que si està encés llavors
//la sortida està activa, altrament estaria desactivada. Les sortides simulades
//són les següents:
//-1: finestres
//-2: electroventiladors
//-3: electrovàlvula de control d'aigua
//-4: alarma general
//Com que només disposem d'un LED, pot passar que sembla que desactivar el LED
//també fa que es desactivin totes les sortides, però la lògica d'aquestes es controla
//des de la consola del sistema on restaria activa.

#include "AuxVarB.h"

configuration ActionC{
}
implementation{

   components MainC,ActionP;
   components LedsC;
   components SerialActiveMessageC;
   components new SerialAMSenderC(AM_ACTIONMSG) as SerialAMSenderActionC;
   components new SerialAMReceiverC(AM_ACTIONMSG) as SerialAMReceiverActionC;

   ActionP.Boot-> MainC;
   ActionP.Leds -> LedsC;

   ActionP.actionControl -> SerialActiveMessageC;

   ActionP.ReceiveAction -> SerialAMReceiverActionC.Receive;
   ActionP.PacketAction -> SerialAMSenderActionC;
   ActionP.AMSendAction -> SerialAMSenderActionC;
}
```

D.4.2 ActionP.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: ActionP.nc
//
//Aquest component tradueix la senyalització d'activació/desactivació, que
//rep via sèrie la mote, en la simulació d¡activar o desactivar una sortida
//que controlaria un component de potència. En el nostre cas la simulació
//es realitza sobre el LED vermell (LED0), de forma que si està encés llavors
//la sortida està activa, altrament estaria desactivada. Les sortides simulades
//són les següents:
//-1: finestres
//-2: electroventiladors
//-3: electrovàlvula de control d'aigua
//-4: alarma general
//Com que només disposem d'un LED, pot passar que sembla que desactivar el LED
//també fa que es desactivin totes les sortides, però la lògica d'aquestes es controla
//des de la consola del sistema on restaria activa.

#include "AuxVarB.h"

module ActionP{
   uses interface Boot;

   uses interface SplitControl as actionControl;
   uses interface AMSend as AMSendAction;
   uses interface Receive as ReceiveAction;
   uses interface Packet as PacketAction;
   uses interface Leds;
}
implementation{

   message_t packet;
   task void iniciServei();
   task void actionEstat();
   task void tractarAction();
   task void sortida();

   ActionMsg* data;

   event void Boot.booted(){
            call actionControl.start();
```

```
    }


    event void actionControl.startDone(error_t error){
        if (error!=SUCCESS){
            post iniciServei();
        }
    }

    task void iniciServei(){
            call actionControl.start();
    }

    event void actionControl.stopDone(error_t error){}

    uint8_t sayingAction = 0;
    uint32_t numberAction = 0;
    uint8_t accionamentN = 0;
    uint8_t accionamentOnOff = 0;

    //Enviem missatge de quin accionament activem/desactivem segons petició rebuda
    task void actionEstat(){
        ActionMsg* m;
        if(sayingAction==0){
            atomic sayingAction=1;
            m = (ActionMsg*)call PacketAction.getPayload(&packet, sizeof(ActionMsg));
            m->seqNum = numberAction;
            m->moteId = TOS_NODE_ID;
            m->action = accionamentN;
            m->maniobra = accionamentOnOff;
            call AMSendAction.send(AM_BROADCAST_ADDR, &packet, sizeof(ActionMsg));
            call Leds.led1Toggle();
        }
    }

    //Detecció de recepció d'un missatge ActionMsg adreçat al node directament
    event message_t *ReceiveAction.receive (message_t *msg, void *payload, uint8_t len){
        if(sayingAction==0){
            ActionMsg* m = (ActionMsg*)payload;
            if(m->moteId == TOS_NODE_ID){
                numberAction = m->seqNum;
                accionamentN = m -> action;
                accionamentOnOff = m -> maniobra;
                post tractarAction();
            }
        }
        return msg;
    }

    event void AMSendAction.sendDone (message_t *msg, error_t error){
        atomic sayingAction=0;
        call Leds.led1Toggle();
    }

    //Tractem l'accionament a activar segons la petició rebuda. Es simula sobre una única
    //sortida que s'activarà o desactivarà, en aquest cas el LED vermell, però que facilita
    //la substitució per un altre tipus de tractament. Es tracten fins a les 4 possibilitats
    //d'accionaments
    task void tractarAction(){
        switch(accionamentN){
      case 1: //Finestres motoritzades
                post sortida();
        break;
      case 2: //Electroventiladors
                post sortida();
        break;
      case 3: //Electrovàlvula que controla polvorització
                post sortida();
        break;
      case 4: //Alarmes generals del sistema
                post sortida();
        break;
        }
        post actionEstat();
    }

    //Es tracta l'activació de la sortida. Se simula sobre el LED vermell.
    //Si el LED vermell està encés llavors la sortida estarà activada, altrament estarà desactivada
    task void sortida(){
        if (accionamentOnOff==0){
            call Leds.led0Off();
        }else{
            call Leds.led0On();
        }
    }
}
```

D.4.3 BaseStationC.nc

```
// $Id: BaseStationC.nc,v 1.7 2010/06/29 22:07:13 scipio Exp $

/*                                                      tab:4
 * Copyright (c) 2000-2003 The Regents of the University  of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```

```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * - Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * - Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the
 *   distribution.
 * - Neither the name of the University of California nor the names of
 *   its contributors may be used to endorse or promote products derived
 *   from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL
 * THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * Copyright (c) 2002-2003 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-LICENSE
 * file. If you do not find these files, copies can be found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */

/**
 * The TinyOS 2.x base station that forwards packets between the UART
 * and radio.It replaces the GenericBase of TinyOS 1.0 and the
 * TOSBase of TinyOS 1.1.
 *
 * <p>On the serial link, BaseStation sends and receives simple active
 * messages (not particular radio packets): on the radio link, it
 * sends radio active messages, whose format depends on the network
 * stack being used. BaseStation will copy its compiled-in group ID to
 * messages moving from the serial link to the radio, and will filter
 * out incoming radio messages that do not contain that group ID.</p>
 *
 * <p>BaseStation includes queues in both directions, with a guarantee
 * that once a message enters a queue, it will eventually leave on the
 * other interface. The queues allow the BaseStation to handle load
 * spikes.</p>
 *
 * <p>BaseStation acknowledges a message arriving over the serial link
 * only if that message was successfully enqueued for delivery to the
 * radio link.</p>
 *
 * <p>The LEDS are programmed to toggle as follows:</p>
 * <ul>
 * <li><b>RED Toggle:</b>: Message bridged from serial to radio</li>
 * <li><b>GREEN Toggle:</b> Message bridged from radio to serial</li>
 * <li><b>YELLOW/BLUE Toggle:</b> Dropped message due to queue overflow in either direction</li>
 * </ul>
 *
 * @author Phil Buonadonna
 * @author Gilman Tolle
 * @author David Gay
 * @author Philip Levis
 * @date August 10 2005
 */

configuration BaseStationC {
}
implementation {
  components MainC, BaseStationP, LedsC;
  components SensorBC, ActionC;
  components ActiveMessageC as Radio, SerialActiveMessageC as Serial;

  MainC.Boot <- BaseStationP;

  BaseStationP.RadioControl -> Radio;
  BaseStationP.SerialControl -> Serial;

  BaseStationP.UartSend -> Serial;
  BaseStationP.UartReceive -> Serial.Receive;
  BaseStationP.UartPacket -> Serial;
  BaseStationP.UartAMPacket -> Serial;

  BaseStationP.RadioSend -> Radio;
  BaseStationP.RadioReceive -> Radio.Receive;
  BaseStationP.RadioSnoop -> Radio.Snoop;
  BaseStationP.RadioPacket -> Radio;
  BaseStationP.RadioAMPacket -> Radio;
  BaseStationP.PacketAcknowledgements->Radio;

  BaseStationP.Leds -> LedsC;
}
```

D.4.4 BaseStationP.nc

```
// $Id: BaseStationP.nc,v 1.12 2010/06/29 22:07:14 scipio Exp $

/*                                                      tab:4
 * Copyright (c) 2000-2005 The Regents of the University  of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * - Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * - Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the
 *   distribution.
 * - Neither the name of the University of California nor the names of
 *   its contributors may be used to endorse or promote products derived
 *   from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL
 * THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * Copyright (c) 2002-2005 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the attached INTEL-LICENSE
 * file. If you do not find these files, copies can be found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite 1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */

/*
 * @author Phil Buonadonna
 * @author Gilman Tolle
 * @author David Gay
 * Revision:    $Id: BaseStationP.nc,v 1.12 2010/06/29 22:07:14 scipio Exp $
 */

/*
 * BaseStationP bridges packets between a serial channel and the radio.
 * Messages moving from serial to radio will be tagged with the group
 * ID compiled into the BaseStation, and messages moving from radio to
 * serial will be filtered by that same group id.
 */

#include "AM.h"
#include "Serial.h"

module BaseStationP @safe() {
  uses {
    interface Boot;
    interface SplitControl as SerialControl;
    interface SplitControl as RadioControl;

    interface AMSend as UartSend[am_id_t id];
    interface Receive as UartReceive[am_id_t id];
    interface Packet as UartPacket;
    interface AMPacket as UartAMPacket;

    interface AMSend as RadioSend[am_id_t id];
    interface Receive as RadioReceive[am_id_t id];
    interface Receive as RadioSnoop[am_id_t id];
    interface Packet as RadioPacket;
    interface AMPacket as RadioAMPacket;
     interface PacketAcknowledgements;

    interface Leds;
  }
}

implementation
{
  enum {
    UART_QUEUE_LEN = 12,
    RADIO_QUEUE_LEN = 12,
  };

  message_t  uartQueueBufs[UART_QUEUE_LEN];
  message_t  * ONE_NOK uartQueue[UART_QUEUE_LEN];
  uint8_t    uartIn, uartOut;
  bool       uartBusy, uartFull;
```

```
message_t  radioQueueBufs[RADIO_QUEUE_LEN];
message_t  * ONE_NOK radioQueue[RADIO_QUEUE_LEN];
uint8_t    radioIn, radioOut;
bool       radioBusy, radioFull;

task void uartSendTask();
task void radioSendTask();

void dropBlink() {
  call Leds.led0Toggle();
  call Leds.led0Toggle();
}

void failBlink() {
  call Leds.led0Toggle();
  call Leds.led0Toggle();
}

event void Boot.booted() {
  uint8_t i;

  for (i = 0; i < UART_QUEUE_LEN; i++)
    uartQueue[i] = &uartQueueBufs[i];
  uartIn = uartOut = 0;
  uartBusy = FALSE;
  uartFull = TRUE;

  for (i = 0; i < RADIO_QUEUE_LEN; i++)
    radioQueue[i] = &radioQueueBufs[i];
  radioIn = radioOut = 0;
  radioBusy = FALSE;
  radioFull = TRUE;

  call RadioControl.start();
  call SerialControl.start();
}

event void RadioControl.startDone(error_t error) {
  if (error == SUCCESS) {
    radioFull = FALSE;
  }
}

event void SerialControl.startDone(error_t error) {
  if (error == SUCCESS) {
    uartFull = FALSE;
  }
}

event void SerialControl.stopDone(error_t error) {}
event void RadioControl.stopDone(error_t error) {}

uint8_t count = 0;

message_t* ONE receive(message_t* ONE msg, void* payload, uint8_t len);

event message_t *RadioSnoop.receive[am_id_t id](message_t *msg,
                                                void *payload,
                                                uint8_t len) {
  return receive(msg, payload, len);
}

event message_t *RadioReceive.receive[am_id_t id](message_t *msg,
                                                  void *payload,
                                                  uint8_t len) {
  return receive(msg, payload, len);
}

message_t* receive(message_t *msg, void *payload, uint8_t len) {
  message_t *ret = msg;

  atomic {
    if (!uartFull)
    {
      ret = uartQueue[uartIn];
      uartQueue[uartIn] = msg;

      uartIn = (uartIn + 1) % UART_QUEUE_LEN;

      if (uartIn == uartOut)
        uartFull = TRUE;

      if (!uartBusy)
        {
          post uartSendTask();
          uartBusy = TRUE;
        }
    }
      else
    dropBlink();
    }

  return ret;
}

uint8_t tmpLen;

task void uartSendTask() {
  uint8_t len;
```

```
  am_id_t id;
  am_addr_t addr, src;
  message_t* msg;
  am_group_t grp;
  atomic
    if (uartIn == uartOut && !uartFull)
  {
    uartBusy = FALSE;
    return;
  }

  msg = uartQueue[uartOut];
  tmpLen = len = call RadioPacket.payloadLength(msg);
  id = call RadioAMPacket.type(msg);
  addr = call RadioAMPacket.destination(msg);
  src = call RadioAMPacket.source(msg);
  grp = call RadioAMPacket.group(msg);
  call UartPacket.clear(msg);
  call UartAMPacket.setSource(msg, src);
  call UartAMPacket.setGroup(msg, grp);

  if (call UartSend.send[id](addr, uartQueue[uartOut], len) == SUCCESS)
        {
    call Leds.led1Toggle();
    call Leds.led1Toggle();
        }
  else
    {
  failBlink();
  post uartSendTask();
    }
}

event void UartSend.sendDone[am_id_t id](message_t* msg, error_t error) {
  if (error != SUCCESS)
    failBlink();
  else
    atomic
  if (msg == uartQueue[uartOut])
    {
      if (++uartOut >= UART_QUEUE_LEN)
        uartOut = 0;
      if (uartFull)
        uartFull = FALSE;
    }
  post uartSendTask();
}

event message_t *UartReceive.receive[am_id_t id](message_t *msg,
                                                 void *payload,
                                                 uint8_t len) {
  message_t *ret = msg;
  bool reflectToken = FALSE;

  atomic
    if (!radioFull)
  {
    reflectToken = TRUE;
    ret = radioQueue[radioIn];
    radioQueue[radioIn] = msg;
    if (++radioIn >= RADIO_QUEUE_LEN)
      radioIn = 0;
    if (radioIn == radioOut)
      radioFull = TRUE;

    if (!radioBusy)
      {
        post radioSendTask();
        radioBusy = TRUE;
      }
  }
    else
  dropBlink();

  if (reflectToken) {
    //call UartTokenReceive.ReflectToken(Token);
  }

  return ret;
}

task void radioSendTask() {
  uint8_t len;
  am_id_t id;
  am_addr_t addr,source;
  message_t* msg;

  atomic
    if (radioIn == radioOut && !radioFull)
  {
    radioBusy = FALSE;
    return;
  }

  msg = radioQueue[radioOut];
  len = call UartPacket.payloadLength(msg);
  addr = call UartAMPacket.destination(msg);
  source = call UartAMPacket.source(msg);
  id = call UartAMPacket.type(msg);
```

```
    call RadioPacket.clear(msg);
    call RadioAMPacket.setSource(msg, source);
     call PacketAcknowledgements.requestAck(msg);
    if (call RadioSend.send[id](addr, msg, len) == SUCCESS)
          {
      call Leds.led1Toggle();
      call Leds.led1Toggle();
          }
    else
      {
    failBlink();
    post radioSendTask();
      }
  }

  event void RadioSend.sendDone[am_id_t id](message_t* msg, error_t error) {
    if (error != SUCCESS)
      failBlink();
    else
      atomic
    if (msg == radioQueue[radioOut])
      {
        if (++radioOut >= RADIO_QUEUE_LEN)
          radioOut = 0;
        if (radioFull)
          radioFull = FALSE;
      }

    post radioSendTask();
  }
}
```

D.4.5 SensorBC.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: SensorBC.nc
//
//Aquest component llegeix els sensors del node, complint un cicle
//de duració programada, que pot ser reprogramat i els envia al sistema
//per al seu registre. A més avisa de la seva activació al sistema.
//Correspon a la versió del node-enllaç que actua com a sensor de referència.


#include "AuxVarB.h"

configuration SensorBC{

}
implementation{
  components MainC, SensorBP;
  components new TimerMilliC() as Timer;
  components new TimerMilliC() as TimerStabilize;
  components new TimerMilliC() as TimerInit;
  components HplAtm128GeneralIOC as IO;
  components LedsC as LedsC;

  SensorBP.Leds -> LedsC;
  SensorBP.Boot -> MainC;
  SensorBP.Timer -> Timer;
  SensorBP.TimerStabilize -> TimerStabilize;
  SensorBP.TimerInit -> TimerInit;

  components new AdcReadClientC() as LectorADC;
  SensorBP.LecturaAdc -> LectorADC;
  LectorADC.Atm128AdcConfig -> SensorBP.Atm128AdcConfig;

  components SerialActiveMessageC;
  components new SerialAMSenderC(AM_SENSORSMSG);
  SensorBP.Packet -> SerialAMSenderC;
  SensorBP.AMSend -> SerialAMSenderC;

  components new SerialAMSenderC(AM_HELLOMSG) as SerialAMSenderHelloC;
  components new SerialAMReceiverC(AM_HELLOMSG) as SerialAMReceiverHelloC;
  SensorBP.ReceiveHello -> SerialAMReceiverHelloC.Receive;
  SensorBP.PacketHello -> SerialAMSenderHelloC;
  SensorBP.AMSendHello -> SerialAMSenderHelloC;

  components new SerialAMSenderC(AM_SETTINGS) as SerialAMSenderSettingsC;
  components new SerialAMReceiverC(AM_SETTINGS) as SerialAMReceiverSettingsC;
  SensorBP.ReceiveSettings -> SerialAMReceiverSettingsC.Receive;
  SensorBP.PacketSettings -> SerialAMSenderSettingsC;
  SensorBP.AMSendSettings -> SerialAMSenderSettingsC;

  SensorBP.powerSerie -> SerialActiveMessageC;

  SensorBP.SinkPowerSensors -> IO.PortG1;
}
```

D.4.6 SensorBP.nc

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: SensorBP.nc
//
//Aquest component llegeix els sensors del node, complint un cicle
//de duració programada, que pot ser reprogramat i els envia al sistema
// per al seu registre. A més avisa de la seva activació al sistema.

#include "AuxVarB.h"

module SensorBP{
   uses interface Boot;
   uses interface Timer<TMilli> as Timer;
   uses interface Timer<TMilli> as TimerStabilize;
   uses interface Timer<TMilli> as TimerInit;
   uses interface GeneralIO as SinkPowerSensors;

   uses interface Read<uint16_t> as LecturaAdc;
   provides interface Atm128AdcConfig;

   uses interface SplitControl as powerSerie;
   uses interface Packet;
   uses interface AMSend;
   uses interface AMSend as AMSendHello;
   uses interface AMSend as AMSendSettings;
   uses interface Receive as ReceiveHello;
   uses interface Receive as ReceiveSettings;
   uses interface Packet as PacketHello;
   uses interface Packet as PacketSettings;
   uses interface Leds;
}
implementation{

   enum{
         INITIALIZING_COMM,
         READY,
         WARMING_UP,
         WARMED_UP,
         READING_BAT,
         BAT_READ,
         READING_PHOTO,
         PHOTO_READ,
         READING_TEMP,
         TEMP_READ,
         READING_HUM,
         HUM_READ,
         SENSORS_READ,
         DATA_SENT,
         ERROR
   };
   message_t packet;
   task void seguentPas();
   void _turnOnSensors();
   void _turnOffSensors();
   task void iniciServei();
   task void sayHello();

   uint8_t estat=READY;

   SensorsMsg* data;
   Settings* timing;
   uint32_t timeset;

   event void Boot.booted(){
         //A l'inici s'apaguen els sensors
         _turnOffSensors();

         //Es carrega el valor per defecte de cicle de lectura. Serà un cicle
         //de 30 segons.
         timing = (Settings*) call Packet.getPayload(&packet, sizeof(Settings));
         timing->moteId=TOS_NODE_ID;
         timing->setmilitimer=30000;
         timeset=timing->setmilitimer;

         atomic estat=INITIALIZING_COMM;
         call powerSerie.start();

         //iniciem el timer que fara que es llegeixin dels sensors
         call Timer.startPeriodic(timeset);
   }

   //S'activen els sensors del node
   void _turnOnSensors(){
         call SinkPowerSensors.makeOutput();
         call SinkPowerSensors.clr();
   }

   //Se desactiven els sensors del node
   void _turnOffSensors(){
         call SinkPowerSensors.makeInput();
         call SinkPowerSensors.set();
   }
```

```
//El temporitzador de cicle marca quant llegirem els valors dels sensors
event void Timer.fired(){
	if(estat==READY){
		post seguentPas();
	}
}

// Seqüència d'accions de lectura dels sensors:
// 1- engegar sensors i esperar estabilitzarse
// 2- llegir bateria
// 3- llegir temperatura
// 4- llegir photo
// 5- llegir humitat
// 6- apagar sensors
// 7- enviar les dades dels sensors
// 8- preparat per iniciar de nou el cicle quan ho marqui el temporitzador

task void seguentPas(){
	switch(estat){
		case INITIALIZING_COMM:
			break;
		case READY:	// Pas 1
			data = (SensorsMsg*) call Packet.getPayload(&packet, sizeof(SensorsMsg));
			atomic estat= WARMING_UP;
			data->counter++;
			data->vrefmilivolts=2560;
			data->moteId = TOS_NODE_ID;
			_turnOnSensors();
			call TimerStabilize.startOneShot(100);
			break;
		case WARMED_UP:	// Pas 2
			atomic estat=READING_BAT;
			if(call LecturaAdc.read()!=SUCCESS){
				atomic estat=ERROR;
				post seguentPas();
			}
			break;
		case BAT_READ: // Pas 3
			atomic estat=READING_PHOTO;
			if(call LecturaAdc.read()!=SUCCESS){
				atomic estat=ERROR;
				post seguentPas();
			}
			break;
		case PHOTO_READ: // Pas 4
			atomic estat=READING_TEMP;
			if(call LecturaAdc.read()!=SUCCESS){
				atomic estat=ERROR;
				post seguentPas();
			}
			break;
		case TEMP_READ: // Pas 5
			atomic estat=READING_HUM;
			if(call LecturaAdc.read()!=SUCCESS){
				atomic estat=ERROR;
				post seguentPas();
			}
			break;
		case HUM_READ: // Pas 6
			_turnOffSensors();
			atomic estat = SENSORS_READ;
			post seguentPas();
			break;
		case SENSORS_READ: // Pas 7
			data->militimer = timeset;
			if(call AMSend.send(AM_BROADCAST_ADDR, &packet, sizeof(SensorsMsg))!=SUCCESS){
				call Leds.led1Toggle();
				atomic estat=ERROR;
				post seguentPas();
			}
			break;
		case DATA_SENT: // Pas 8
				call Leds.led1Toggle();
		case ERROR:
		default:
			atomic estat=READY;
			_turnOffSensors();
			break;
	}
}

//Quan el temporitzador d'estabilització s'activa, ens indica que els sensors ja estan preparats
//Canviem el valor d'estat per continuar.
event void TimerStabilize.fired(){
	if(estat==WARMING_UP){
		atomic estat = WARMED_UP;
	}
	else{
		atomic estat=ERROR;
	}
	post seguentPas();
}

//La lectura de cada ADC es passa al seu valor corresponent
event void LecturaAdc.readDone(error_t result, uint16_t val){
	if(result==SUCCESS){
		if(estat==READING_BAT){
			data->countsBat = val;
```

```
                atomic estat=BAT_READ;
            }else if(estat == READING_PHOTO){
                data->countsPhoto = val;
                atomic estat=PHOTO_READ;
            }else if(estat== READING_TEMP){
                data->countsTemp = val;
                atomic estat=TEMP_READ;
            }else if(estat== READING_HUM){
                data->countsHum = val;
                atomic estat=HUM_READ;
            }else{      //aquest estat no es possible, pero s'ha de retornar alguna cosa
                atomic estat=ERROR;
            }
        }
        else{
            atomic estat= ERROR;
        }
        post seguentPas();
    }

    //Segons el punt del cicle de lectura, es llegeix del sensor que correspon
    async command uint8_t Atm128AdcConfig.getChannel(){
        if(estat==READING_BAT){
            return ATM128_ADC_SNGL_ADC0;
        }else if(estat == READING_PHOTO){
            return ATM128_ADC_SNGL_ADC1;
        }else if(estat== READING_TEMP){
            return ATM128_ADC_SNGL_ADC2;
        }else if(estat == READING_HUM){
            return ATM128_ADC_SNGL_ADC3;
        }else{      //aquest estat no es possible, pero s'ha de retornar alguna cosa
            return ATM128_ADC_SNGL_ADC0;
        }
    }

    //Es torna la referència de voltatge del conversor ADC
    async command uint8_t Atm128AdcConfig.getRefVoltage(){
        return  ATM128_ADC_VREF_2_56;
    }

    async command uint8_t Atm128AdcConfig.getPrescaler(){
        return ATM128_ADC_PRESCALE;
    }

    event void powerSerie.startDone(error_t error){
        if (error != SUCCESS){
            post iniciServei();
        }else{
            atomic estat=READY;
            call Leds.led2On();
            call Timer.startPeriodic(timeset);
            call TimerInit.startOneShot(1500);
        }
    }

    task void iniciServei(){
            call powerSerie.start();
    }

    event void powerSerie.stopDone(error_t error){}

    event void AMSend.sendDone(message_t *msg, error_t error){
        atomic estat=DATA_SENT;
        call Leds.led1Toggle();
        post seguentPas();
    }

    event void TimerInit.fired(){
        post sayHello();
    }

    message_t packetSerial;
    uint8_t sayingHello = 0;
    uint32_t numberHello=0;

    //Enviem missatge de node iniciat, que inclou Id, comptador i temps de cicle de lectura
    task void sayHello(){
        HelloMsg* m;
        if(sayingHello==0){
            atomic sayingHello=1;
            m = (HelloMsg*)call PacketHello.getPayload(&packetSerial, sizeof(HelloMsg));
            m->seqNum = numberHello;
            m->moteId = TOS_NODE_ID;
            m->militimer = timeset;
            call AMSendHello.send(AM_BROADCAST_ADDR, &packetSerial, sizeof(HelloMsg));
            call Leds.led1Toggle();
        }
    }

    //Detecció de recepció d'un missatge HelloMsg adreçat al node directament
    event message_t *ReceiveHello.receive (message_t *msg, void *payload, uint8_t len){
        if(sayingHello==0){
            HelloMsg* m = (HelloMsg*)payload;
            if((m->moteId == TOS_NODE_ID)||(m->moteId == 65535L)){
                numberHello = m->seqNum;
                post sayHello();
            }
        }
        return msg;
```

```
    }

    event void AMSendHello.sendDone (message_t *msg, error_t error){
        atomic sayingHello=0;
        call Leds.led1Toggle();
    }

    uint8_t chSettings=0;
    uint32_t numberSettings=0;
    task void confirmTimeChange();
    task void setTimer();

    //S'envia el valor de node, comptador i temps de lectura actual al node
    task void confirmTimeChange(){
        Settings* s;
        if(chSettings==0){
            atomic chSettings=1;
            s = (Settings*)call PacketSettings.getPayload(&packetSerial, sizeof(Settings));
            s->moteId = TOS_NODE_ID;
            s->setmilitimer = timeset;
            s->seqNum = numberSettings;
            call AMSendSettings.send (AM_BROADCAST_ADDR, &packetSerial, sizeof(Settings));
            call Leds.led1Toggle();
        }
    }

    //Es fa el canvi del cicle de lectura al nou temps que s'ha rebut al node
    task void setTimer(){
        if(chSettings==0){
            call Leds.led0Toggle();
            call Timer.stop();
            call Timer.startPeriodic(timeset);
            post confirmTimeChange();
            call Leds.led0Toggle();
        }
    }

    //Es rep un missate Settings amb un nou valor de temps de cicle de lectura de sensors
    event message_t *ReceiveSettings.receive (message_t *msg, void *payload, uint8_t len){
        if(chSettings==0){
            Settings* s = (Settings*)payload;
            if((s->moteId == TOS_NODE_ID)||(s->moteId == 65535L)){
                timeset = s->setmilitimer;
                numberSettings = s->seqNum;
                post setTimer();
            }
        }
        return msg;
    }

    event void AMSendSettings.sendDone (message_t *msg, error_t error){
        atomic chSettings=0;
        call Leds.led1Toggle();
    }

}
```

D.4.7 AuxVarB.h

```
//TFC ETIS 2n Semestre curs 2010-2011
// Nom del Projecte: SCAFIA
//
// Consultor: Jordi Bécares Ferrés
// @author Joan Carles Martínez Rodríguez
// Fitxer: AuxVar.h

#ifndef AUXVARB
#define AUXVARB

//Descripció dels diferents missatges
//  AM_SENSORSMSG   -> Lectures als sensors de node
//  AM_HELLOMSG             -> Inici de node
//  AM_SETTINGS         -> Fixar cicle de lectura del sensors
enum {
  AM_SENSORSMSG   = 40,
  AM_HELLOMSG             = 41,
  AM_SETTINGS         = 44,
  AM_ACTIONMSG      = 45
};

typedef nx_struct SensorsMsg {
  nx_uint16_t moteId;     //Identificació del node
  nx_uint16_t countsTemp; //Valor de temperatura
  nx_uint16_t countsPhoto; //Valor de lluminositat
  nx_uint16_t countsBat; //Valor de bateria interna
  nx_uint16_t countsHum;        //Valor d'humitat
  nx_uint16_t vrefmilivolts; //Tensió de referència de la bateria
  nx_uint32_t militimer; //Valor del cicle de lectura de sensors
  nx_uint32_t counter;    //Comptador o seqüència interna
} SensorsMsg;

//Registre o estructura del missatge d'inici del node
typedef nx_struct HelloMsg{
  nx_uint16_t moteId; //Identificació del node
  nx_uint32_t seqNum; //Identificador del missatge
  nx_uint32_t militimer; //Temps de cicle de lectura de sensors programat
```

```
} HelloMsg;

//Registre o estructura del missatge de configuració del temps de cicle de lectura
typedef nx_struct Settings{
   nx_uint16_t moteId; //Identificació del node
   nx_uint32_t setmilitimer; //Temps de cicle de lectura de sensors
   nx_uint32_t seqNum; //Comptador o seqüència interna
}Settings;

//Registre o estructura del missatge d'activació/desactivació d'accionaments
typedef nx_struct ActionMsg{
   nx_uint16_t moteId; //Identificació del node
   nx_uint8_t action; //Accionament que volem activar/desactivar
   nx_uint8_t maniobra; //1 = activar; 0 = desactivar
   nx_uint32_t seqNum; //Comptador o seqüència interna
}ActionMsg;

#endif /* AUXVARB */
```

Annex E. Sortides de fitxer

Es representa un petit exemple dels fitxers de dietari i estadístic del sistema

E.1 Fitxer de dietari scafia_aaaammdd.log

```
SCAFIA CON 23-05-2011 16:49:01 Inici de sessió de consola de control del sistema. Paràmetres introduïts:  -iniciLog
SCAFIA CON 23-05-2011 16:49:01 S'ha demanat l'inici de registre de telemetria i accions del sistema (-iniciLog)
SCAFIA - TEL =>| 23-05-2011 16:49:03 [138] Valors Node Ref. 0 Bateria(V):[2.96] Lluminositat(Lux):[23.14] Temperatura(°C):[29.00] Humitat(%):[41.14]
SCAFIA - TEL =>| 23-05-2011 16:49:03 [138] Valors Node Ref. 0 Bateria(V):[2.96] Lluminositat(Lux):[23.14] Temperatura(°C):[29.00] Humitat(%):[41.14]
SCAFIA - TEL =>| 23-05-2011 16:49:13 [139] Valors Node Ref. 0 Bateria(V):[2.96] Lluminositat(Lux):[23.31] Temperatura(°C):[28.75] Humitat(%):[40.97]
SCAFIA - TEL =>| 23-05-2011 16:49:13 [139] Valors Node Ref. 0 Bateria(V):[2.96] Lluminositat(Lux):[23.31] Temperatura(°C):[28.75] Humitat(%):[40.97]
SCAFIA CON 23-05-2011 16:51:06 Inici de sessió de consola de control del sistema. Paràmetres introduïts:  -iniciLog
SCAFIA CON 23-05-2011 16:51:06 S'ha demanat l'inici de registre de telemetria i accions del sistema (-iniciLog)
SCAFIA CON 23-05-2011 16:51:13 Inici de sessió de consola de control del sistema. Paràmetres introduïts:  -iniciLog
SCAFIA CON 23-05-2011 16:51:13 S'ha demanat l'inici de registre de telemetria i accions del sistema (-iniciLog)
SCAFIA - TEL =>| 23-05-2011 16:51:20 [152] Valors Node Ref. 0 Bateria(V):[2.96] Lluminositat(Lux):[22.97] Temperatura(°C):[29.00] Humitat(%):[41.29]
SCAFIA - TEL =>| 23-05-2011 16:51:30 [153] Valors Node Ref. 0 Bateria(V):[2.96] Lluminositat(Lux):[9.12] Temperatura(°C):[28.75] Humitat(%):[40.67]
SCAFIA A! NET 23-05-2011 16:51:33 La xarxa de nodes està caiguda, o els nodes tenen un temps de lectura de sensors incorrecte.
SCAFIA A! TEL =>| 23-05-2011 16:51:40 [154] Valors Node Ref. 0Bateria(V):[2.96] Lluminositat(Lux):[2.20] Temperatura(°C):[29.50] Humitat(%):[54.81]
SCAFIA A! TEL =>| 23-05-2011 16:51:49 [155] Valors Node Ref. 0Bateria(V):[2.96] Lluminositat(Lux):[2.20] Temperatura(°C):[30.00] Humitat(%):[55.02]
SCAFIA CON 23-05-2011 17:26:23 Inici de sessió de consola de control del sistema. Paràmetres introduïts:  -iniciLog
SCAFIA CON 23-05-2011 17:26:23 S'ha demanat l'inici de registre de telemetria i accions del sistema (-iniciLog)
SCAFIA - TEL =>| 23-05-2011 17:26:29 [368] Valors Node Ref. 0 Bateria(V):[2.96] Lluminositat(Lux):[23.31] Temperatura(°C):[29.25] Humitat(%):[39.38]
SCAFIA - TEL =>| 23-05-2011 17:26:39 [369] Valors Node Ref. 0 Bateria(V):[2.96] Lluminositat(Lux):[23.31] Temperatura(°C):[29.25] Humitat(%):[38.79]
SCAFIA A! NET 23-05-2011 17:26:43 La xarxa de nodes està caiguda, o els nodes tenen un temps de lectura de sensors incorrecte.
SCAFIA A! ACC 23-05-2011 17:26:43 S'activa el sistema d'alarma de la instal·lació.
SCAFIA A! ACC |=> 23-05-2011 17:26:43 [0] S'envia al node 0 que les alarmes generals de sistema s'han d'activar.
SCAFIA A! ACC =>| 23-05-2011 17:26:44 [0] El node 0 informa  que les alarmes generals de sistema estan activats.
SCAFIA A! TEL =>| 23-05-2011 17:26:49 [370] Valors Node Ref. 0Bateria(V):[2.96] Lluminositat(Lux):[23.52] Temperatura(°C):[29.25] Humitat(%):[39.38]
...
```

E.2 Fitxer d'exemple de valors de telemetria scafia_aaaammdd.csv

1;29-05-2011 22:27:12;0;2.96;14.3581081081081O9;28.749999999999996;48.216007714561236
2;29-05-2011 22:27:16;0;2.96;14.189189189189189;28.749999999999996;47.476498393785754
3;29-05-2011 22:27:21;0;2.96;14.358108108108109;29.000000000000004;47.35431445158968
4;29-05-2011 22:27:26;0;2.96;14.189189189189189;28.749999999999996;47.920203986251046
2;29-05-2011 22:27:29;1;2.7800000000000002;174.46043165467626;30.000000000000004;53.24134494459345
5;29-05-2011 22:27:31;0;2.955;14.213197969543147;28.749999999999996;47.18069466547557
6;29-05-2011 22:27:36;0;2.955;14.213197969543147;29.000000000000004;47.79826114957334
7;29-05-2011 22:27:41;0;2.955;14.55160744500846;29.000000000000004;47.5022966842509
8;29-05-2011 22:27:46;0;2.955;14.55160744500846;28.749999999999996;47.77230212209595
9;29-05-2011 22:27:51;0;2.955;14.55160744500846;29.000000000000004;47.79826114957334
10;29-05-2011 22:27:56;0;2.955;14.55160744500846;29.000000000000004;47.5022966842509
3;29-05-2011 22:27:59;1;2.775;169.00900900900905;30.500000000000004;53.59632848818565
11;29-05-2011 22:28:00;0;2.955;14.55160744500846;29.000000000000004;47.650278916912114
4;29-05-2011 22:28:04;1;2.775;163.78378378378378;30.75;53.62556343978429
12;29-05-2011 22:28:05;0;2.955;14.720812182741117;29.000000000000004;47.20633221892847
5;29-05-2011 22:28:09;1;2.775;174.95495495495499;31.000000000000004;54.39797199604292
13;29-05-2011 22:28:10;0;2.955;14.720812182741117;29.000000000000004;47.5022966842509
6;29-05-2011 22:28:13;1;2.77;149.45848375451263;30.75;52.882827104053206
14;29-05-2011 22:28:15;0;2.955;14.382402707275803;29.000000000000004;47.79826114957334
7;29-05-2011 22:28:18;1;2.77;141.87725631768953;30.500000000000004;51.814733081376154
15;29-05-2011 22:28:20;0;2.955;14.55160744500846;29.000000000000004;48.09422561489577
8;29-05-2011 22:28:23;1;2.77;142.77978339350182;30.25;51.786500728275485
16;29-05-2011 22:28:25;0;2.955;14.55160744500846;29.000000000000004;47.650278916912114
9;29-05-2011 22:28:28;1;2.77;136.64259927797835;30.25;53.12196922843158
17;29-05-2011 22:28:30;0;2.955;14.55160744500846;28.749999999999996;47.62440025794084
10;29-05-2011 22:28:33;1;2.77;129.6028880866426;30.25;53.270354617337816
18;29-05-2011 22:28:35;0;2.955;14.55160744500846;29.000000000000004;48.09422561489577
11;29-05-2011 22:28:38;1;2.77;140.7942238267148;30.25;53.270354617337816
19;29-05-2011 22:28:39;0;2.955;14.55160744500846;29.000000000000004;47.5022966842509
12;29-05-2011 22:28:43;1;2.77;132.31046931407943;30.25;55.050979284212616
20;29-05-2011 22:28:44;0;2.955;14.720812182741117;29.000000000000004;47.650278916912114
13;29-05-2011 22:28:48;1;2.77;127.43682310469315;30.25;52.67681306171288
21;29-05-2011 22:28:49;0;2.955;14.890016920473773;29.000000000000004;47.5022966842509
14;29-05-2011 22:28:53;1;2.77;136.4620938628159;30.500000000000004;48.54847483555875
22;29-05-2011 22:28:54;0;2.955;15.059221658206429;29.000000000000004;47.79826114957334
15;29-05-2011 22:28:57;1;2.77;138.26714801444044;30.25;47.03816828327601
23;29-05-2011 22:28:59;0;2.96;14.527027027027028;29.000000000000004;47.79826114957334
16;29-05-2011 22:29:02;1;2.77;127.61732851985559;30.25;46.593012116557304
24;29-05-2011 22:29:04;0;2.96;14.527027027027028;29.000000000000004;47.946243382234556
17;29-05-2011 22:29:07;1;2.765;100.72332730560578;30.25;46.44462672765108
25;29-05-2011 22:29:09;0;2.96;14.695945945945946;29.000000000000004;47.35431445158968
18;29-05-2011 22:29:12;1;2.765;98.73417721518987;30.000000000000004;46.122725007711836
26;29-05-2011 22:29:14;0;2.96;14.527027027027028;29.000000000000004;47.35431445158968
19;29-05-2011 22:29:17;1;2.77;128.33935018050542;30.25;45.70269978311991
27;29-05-2011 22:29:19;0;2.96;14.527027027027028;29.000000000000004;47.20633221892847
20;29-05-2011 22:29:22;1;2.77;129.42238267148014;30.000000000000004;45.38120209762001
28;29-05-2011 22:29:23;0;2.96;14.527027027027028;29.000000000000004;47.946243382234556
21;29-05-2011 22:29:27;1;2.77;134.29602888086643;30.000000000000004;44.93628835156491
29;29-05-2011 22:29:28;0;2.96;14.527027027027028;29.000000000000004;47.20633221892847
22;29-05-2011 22:29:32;1;2.77;137.72563176895306;30.000000000000004;45.23289751560163
30;29-05-2011 22:29:33;0;2.96;14.358108108108109;29.000000000000004;47.79826114957334
23;29-05-2011 22:29:36;1;2.77;138.26714801444044;30.000000000000004;46.122725007711836
31;29-05-2011 22:29:38;0;2.96;14.358108108108109;29.000000000000004;47.35431445158968
24;29-05-2011 22:29:41;1;2.77;132.31046931407943;30.000000000000004;47.90237999193223
32;29-05-2011 22:29:43;0;2.96;14.527027027027028;29.000000000000004;47.650278916912114
28;29-05-2011 22:30:01;1;2.77;135.5595667870036;30.25;45.554314394213684
36;29-05-2011 22:30:02;0;2.96;14.358108108108109;29.000000000000004;48.24220784755699
29;29-05-2011 22:30:06;1;2.77;126.89530685920577;30.000000000000004;45.826115843675105
37;29-05-2011 22:30:07;0;2.96;14.358108108108109;29.000000000000004;47.946243382234556
30;29-05-2011 22:30:11;1;2.7600000000000002;75.72463768115941;29.749999999999996;47.728083916050736
38;29-05-2011 22:30:12;0;2.96;13.85135135135135;29.000000000000004;47.79826114957334
39;29-05-2011 22:30:17;0;2.96;13.85135135135135;29.000000000000004;47.20633221892847
40;29-05-2011 22:30:22;0;2.955;14.213197969543147;29.000000000000004;47.20633221892847
41;29-05-2011 22:30:27;0;2.955;14.382402707275803;29.000000000000004;47.5022966842509
42;29-05-2011 22:30:32;0;2.955;14.213197969543147;29.000000000000004;47.5022966842509
43;29-05-2011 22:30:37;0;2.955;14.213197969543147;29.000000000000004;47.650278916912114
1;29-05-2011 22:30:38;1;2.96;113.51351351351352;30.25;45.405929005307456
44;29-05-2011 22:30:42;0;2.955;14.213197969543147;29.000000000000004;48.83413677820186
2;29-05-2011 22:30:42;1;2.765;116.2748643761302;30.000000000000004;44.046460859454704
45;29-05-2011 22:30:46;0;2.96;14.020270270270027;29.000000000000004;47.650278916912114
46;29-05-2011 22:30:51;0;2.96;12.837837837837839;29.249999999999996;47.38006033850684
126;29-05-2011 22:31:08;0;2.955;17.597292724196276;29.000000000000004;46.76238552094481

Exemple d'ús: els valors anteriors mostrats en forma gràfica per a cada mota/node

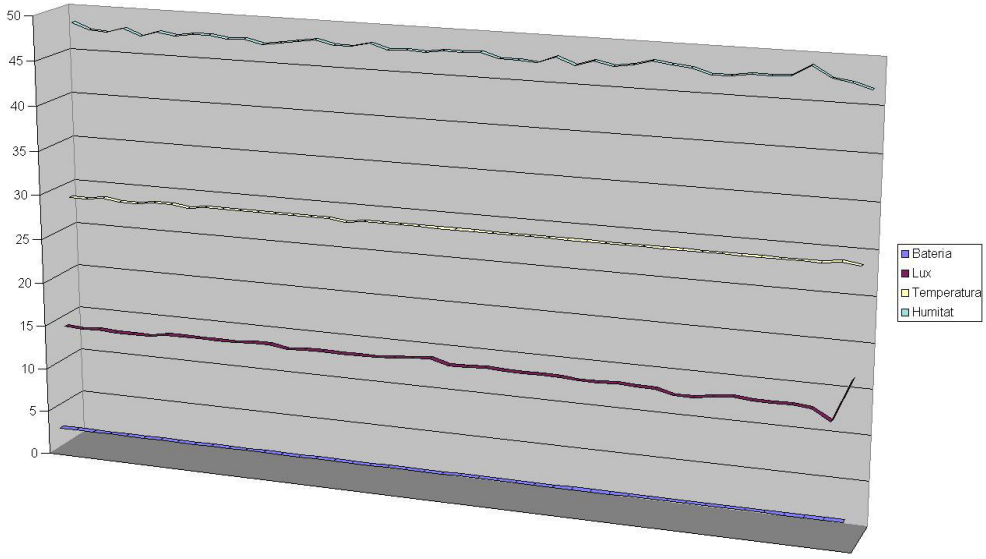


Figura E.1. Gràfic de valors mostrejats per la mota *moteB*

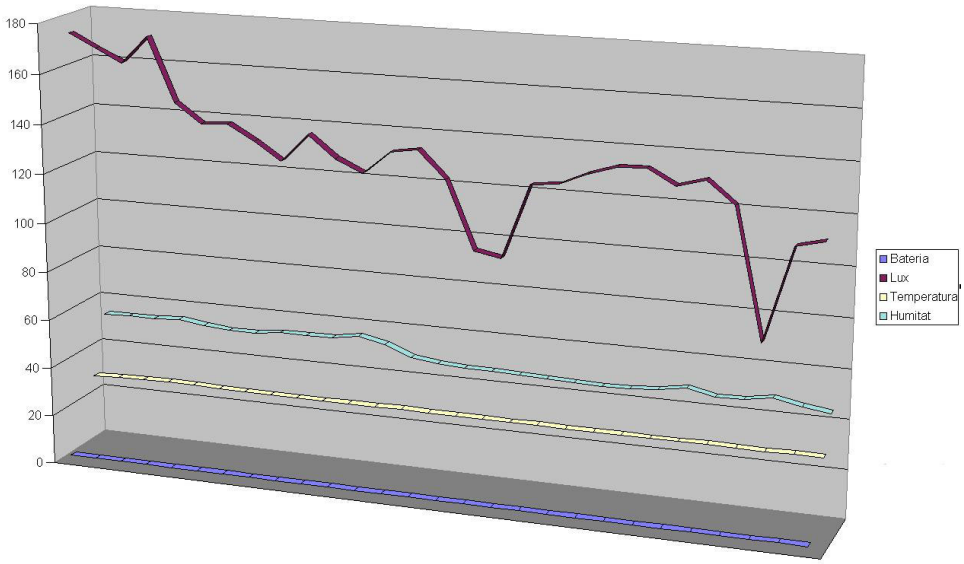


Figura E.2. Gràfic de valors mostrejats per la mota *moteS*