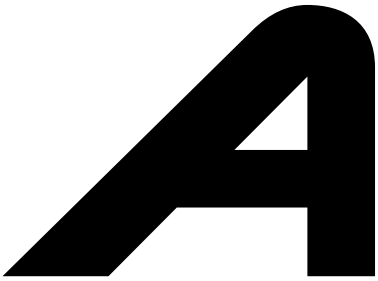
Monitor de l'Estat Operatiu de l'Enllumenat Públic

Ezequiel Díaz Bellido
Enginyeria Tècnica de Telecomunicacions especialitat en Telemàtica
UOC

Consultor: Jordi Bécares Ferrés
Data Lliurament: 10/05/2011



Virginia que se fue cuando todo esto empezaba,
mi madre Antonia,
mi hermano Dani,
mi familia,
Paco, Ricard, Toni, Mireia, Sílvia, Quim y Tatiana porque lo chanan,
Rubén por esas lecciones de termometría,
María por esos consejos de Java,
Jordi Bécares por la paciencia, los consejos y los ánimos,
la UOC por haber puesto IP a un sueño...

y sobre todo a ti, NEKANE

Resum

Aquest treball implementa i fa ús d'una xarxa de sensors sense fils del tipus ZigBit per crear un sistema de monitoratge de l'estat operatiu de l'enllumenat públic. El treball té com a objectiu principal reduir el consum energètic gràcies a que permet a l'usuari un millor control dels elements d'il·luminació públics.

Índex de continguts

Índex d'il·lustracions

1. Introducció

1.1. Justificació

El consum energètic és un tema que sembla no preocupar massa als països quan la cosa va bé i no és tenen problemes per pagar un excés innecessari de consum. Però, si dins d'un àmbit de crisi econòmica, afegim una crisi energètica com la que estem vivint actualment a Catalunya i Espanya, la cosa es complica molt i és realment quan ens toca pensar com podríem optimitzar el consum a més a més d'abaratir costos.

Així doncs, donada la crisi i l'actual problema energètic, es necessari activar mesures d'estalvi energètic i per tant desenvolupar eines que permetin millorar l'eficiència energètica i el consum a gran escala. Aquestes mesures ens permetran regular el consum a curt termini, i a llarg termini ens proporcionaran un estalvi econòmic.

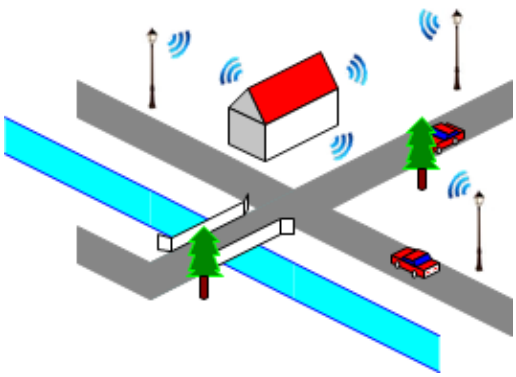
Altrament, hem de ser conscients de que aquestes mesures no forçadament haurien d'estar lligades a moments de crisi puntuals, tothom que fa servir energia hauria de tenir present que és un bé escàs i que l'eficiència energètica d'avui és l'energia del demà.

La idea d'aquest projecte sorgeix després d'observar que bona part de l'enllumenat públic que ens envolta, no disposa de cap tipus d'element de control i que moltes vegades el seu funcionament no és correcte, amb el conseqüent consum energètic innecessari. Així doncs, creiem que una bona justificació del projecte és el sol fet d'intentar reduir aquest consum energètic innecessari amb la implementació d'un sistema de sensors de baix cost.

1.2. Descripció del projecte

Amb aquest projecte es pretén proporcionar una eina que faciliti el control del consum elèctric del mobiliari urbà dedicat a l'enllumenat, sense necessitat d'una forta inversió i aprofitant el mobiliari urbà ja existent sense necessitat de fer-hi modificacions.

Es proposa un projecte a través del qual, un cop finalitzat, es proporciona una eina per saber de forma centralitzada i sense necessitat de cables, l'estat operatiu del mobiliari urbà d'enllumenat d'una localitat. Es pretén conèixer en tot moment si una farola concreta ha d'estar funcionant i si efectivament està funcionant:



Il·lustració 1.2.1: Faroles urbanització

El projecte principalment **es basa en la idea de saber si una làmpada està encesa per la calor que desprèn quan funciona**.

Per dur a terme el projecte, es necessari instal·lar una mota (COU_1_2 24 [1]) a cada element d'enllumenat que es vol monitoritzar. Aquestes motes faran servir **TinyOS 2.1.1** [2] com a base i hauran de ser programades per poder realitzar les funcions que es requereixen. Cada mota prendrà informació de la llum solar (sensor fotodíode) i de la temperatura de làmpada (sensor tèrmic). Aquests nodes es comunicaran amb un ordinador central que recollirà alertes i posteriorment presentarà a l'usuari l'estat de les faroles monitoritzades. La idea principal és estalviar energia i **es pretén detectar les faroles que estan enceses quan no cal, però també permetrà detectar faroles que no s'encenen quan cal**. Des de l'ordinador central, es podrà també modificar els paràmetres llindar de llum i temperatura de les Motes.

Ha de quedar clar que no es pretén donar cap ordre d'encesa i apagat de l'enllumenat, **no es pretén per tant controlar les faroles remotament**, aquesta part queda totalment fora de l'abast del projecte i continuarà fent-se de la mateixa manera que es feia abans de la implementació d'aquest projecte.

Com a eina d'estalvi energètic, el projecte té més sentit com s'havia pensat inicialment: s'havia pensat en crear una xarxa de Motes aprofitant la tecnologia “multi-hop” que ens ofereix TinyOS, per poder cobrir una zona de faroles molt amplia (els paquets saltarien de Mota en Mota fins arribar al seu destí), però donat que nomes tenim 2 Motes i una d'elles l'hem de fer servir com a base, aquesta topologia no s'ha pogut implementar. No obstant, **aquest projecte permet la connexió de Motes en topologia d'estrella** de tal manera que es una molt bona base per després poder adaptar-la a qualsevol altre topologia, amb la finalitat de crear una xarxa de Motes que s'adapti a les necessitats de cadascú i que cobreixi una zona urbana molt més amplia.

Il·lustració 1.2.2:Topologia Estrella

Il·lustració 1.2.3:Topologia multi-hop (reenviament de paquets)

1.3. Objectius d'aquest TFC

L'objectiu principal d'aquest projecte consisteix en **estalviar energia** a la xarxa d'enllumenat públic. Per tal d'aconseguir-ho, el nostre objectiu principal és **proporcionar una solució sense fils de monitoratge de faroles o elements d'enllumenat.** Per assolir aquest objectiu principal, es necessari assolir uns objectius intermedis que detallarem a continuació.

1.3.1. Crear una xarxa de sensors sense fils

S'implanta un sensor COU_1_2 24 a cada element d'enllumenat que es vulgui monitoritzar. Aquests sensors reportaran les alertes en l'enllumenat a una estació base que estarà connectada a un PC.

1.3.2. Monitoritzar temperatura de la làmpada

Necessitem saber si una farola està encesa o no. Per saber quin es el seu estat, com que les làmpades desprenen calor en estar enceses, es pot realitzar una mesura de la temperatura i aquesta ens indicarà el seu estat. Aquest paràmetre es variable en funció del tipus de làmpada.

1.3.3. Monitoritzar la llum ambiental

Necessitem saber quin és el valor de la llum ambiental per determinar quan és de dia o no. Aquest paràmetre es gairebé universal: L'ull humà es qui decideix quan es veu i quan no.

1.3.4. Detectar incidències en faroles

Realitzar les dues mesures anteriors cada cert temps de mostreig, comparar-les amb uns llindars i determinar si la farola està operant correctament o si s'ha produït una incidència.

Els dos tipus d'incidència que es poden donar són:

-La farola està encesa de dia.

-La farola està apagada de nit.

1.3.5. Enviament d'alertes a l'estació base

Les motes que detectin una incidència en la farola que monitoritzen, enviaran una alerta a la base mitjançant el protocol de comunicacions 802.15.4. Aquest protocol és l'encarregat de les comunicacions sense fils de les motes COU_1_2_24 amb les que treballem. L'alerta enviada contindrà la informació de les mesures que han ocasionat l'alerta, el tipus d'alerta, un nombre identificatiu de la mota que ha causat l'alerta i els llindars de temperatura i llum que estaven establerts en el moment de l'alerta.

1.3.6. Proporcionar una interfície gràfica a l'usuari

Per tal de que l'usuari pugui veure aquestes alertes de forma centralitzada, se li ha de proporcionar un entorn gràfic a través del qual pugui veure l'estat de les faroles.

1.3.7. Proveir d'un sistema de llindars modificable

Per tal de poder adaptar el sistema a les necessitats de qualsevol usuari, es necessari crear un sistema a través del qual l'usuari pugui fer servir la interfície gràfica per introduir els seus propis llindars i que aquests es transmetin a les motes per tal de detectar incidències en les faroles mitjançant aquests nous paràmetres.

1.3.8. Proporcionar eficiència al projecte

Donat que l'objectiu principal és l'estalvi energètic, el nostre projecte com a entitat ha de donar exemple. Com a directriu general en tot el projecte ens marquem l'objectiu de lliurar un codi el més eficient possible i que faci un ús intel·ligent dels recursos que s'utilitzen. Per aconseguir aquest objectiu és necessari que el codi de les motes sigui òptim i que es consumeix-hi el mínim possible de bateries: Això és, fer servir sensors, ràdio i CPU únicament quan és imprescindible. També es obligació del projecte avisar a l'usuari, que depenent de l'ús que se li doni al sistema, és consumirà menys o més energia. No és el mateix agafar mostres cada 3 minuts que cada 3 segons. Tampoc és el mateix encendre les faroles a les 7 d'una tarda d'estiu que encendre-les a las 10.

1.4. Enfocament i mètodes seguits

Per tal d'assolir els objectius proposats i tot seguint les recomanacions del tutor, en primer lloc vaig generar un pla de treball en el que s'intentava plasmar quina seria la dinàmica d'aquest Treball de Final de Carrera.

En aquest pla de treball exposàvem els objectius, les fites, les tasques i els esdeveniments més rellevants per tal de que el nostre projecte tirés endavant. En aquest període inicial, paral·lelament a la creació del pla de treball, calia també investigar si el projecte era viable i si la pedra angular del projecte, comprovar l'estat de les làmpades mitjançant mesures de temperatura, seria una tècnica fiable o hauríem de desestimar el projecte. Tot desconeixent les possibilitats de TinyOS, vaig fer recerca sobre quines podien ser les temperatures ambientals màximes que es podien donar a la superfície terrestre a qualsevol part del món. Aquesta dada la considerava vital per poder establir un llindar de temperatura que em permetés detectar bombetes enceses [3][4][5][6][7][8]. Fins i tot vaig consultar a un amic llicenciat en física sobre el tema. Més tard i coneixent una mica més TinyOS, vaig arribar a la conclusió de que no era necessari establir un llindar fixe, es podia arreglar el problema afegint al sistema un llindar variable i modificable per l'usuari. Així doncs, una mota instal·lada al Finlàndia podia esser configurada amb un llindar de temperatura més baix que una instal·lada al Marroc.

Així doncs, retornant al pla de treball, aquest **pla de treball ha estat una eina indispensable** que ha estat consultada durant tot el decurs del treball i que m'ha ajudat molt en els moments en que potser em trobava una mica perdut.

Un cop realitzat el pla, calia posar-lo en marxa, i el més imprescindible després d'allò era preparar l'entorn de treball i començar un període de formació i investigació per aprendre totes les possibilitats que el TinyOS i les motes hem podien oferir.

En la fase d'anàlisi, es van identificar els diferents elements del sistema i vaig començar a posar-hi noms. D'aquí per exemple van sortir noms com motaFarola, FarolaAppC, motaBase, eclipse, LightC, meshprog o GUI d'usuari. Tots aquests elements identificats van ser relacionats directament amb els objectius que s'havien de complir de tal manera que la resta de feina pendent fos únicament fer-ne ús d'ells o fer-los funcionar dins del sistema.

Un cop arribats a aquest punt, la línia i els mètodes de treball seguits per tal d'assolir les fites del pla de treball han estat els següents i en ordre:

- Proves amb els sensors.
- Elaboració d'una primera versió dels mòduls de llum i temperatura en TinyOS.
- Elaboració d'una primera versió “esquelet” del codi de les motes i el PC en TinyOS i Java respectivament.
- Proves i correcció de problemes.
- Alimentació de la documentació per la memòria.
- Implementació de funcionalitats pendents tant al codi de les motes com al del PC.
- Proves i correcció de problemes.
- Documentació del codi i elaboració del document d'execució.
- Elaboració de la memòria i la presentació del projecte.

1.5. Planificació del projecte

Donat que el temps per realitzar el TFC era curt i donat que moltes de les tasques eren completament noves per nosaltres, fer una bona planificació de cada una de elles es convertia en un punt molt important del projecte. Aquesta planificació ens permetria tenir una visió global del desenvolupament del projecte i ens donaria un 'timing' per saber quant de temps tenim per realitzar cada tasca i en cas de passar-se poder reestructurar per donar cabuda al desfàs de temps. El primer que vam realitzar, dins del pla de projecte, va ser una relació de tasques que s'havien de realitzar en termes generals, més endavant aquestes tasques es van especificar més en profunditat fins arribar a les que a continuació mostrem:

Nom de la tasca	Temps	Inici	Final
Pla de treball i planificació del TFC	14,88 **dies**	**lun 07/03/11**	**vie 25/03/11**
Conèixer les possibilitats de les motes i de TinyOS	9 dies	lun 07/03/11	jue 17/03/11
Pensar en una proposta de treball	5 dies	lun 07/03/11	vie 11/03/11
Preparació de l'entorn de treball	9 dies	lun 07/03/11	jue 17/03/11
Realitzar tutorials de TinyOS	7 dies	jue 17/03/11	vie 25/03/11
Creació del document “Pla de treball”	6 dies	vie 18/03/11	vie 25/03/11
Lliurament del document "Pla de treball"	0 dies	vie 25/03/11	vie 25/03/11

Entrega del codi (1/2)	21,88 **dies**	**sáb 26/03/11**	**mar 26/04/11**
Proves amb els sensors a utilitzar.	3 dies	sáb 26/03/11	mar 29/03/11
Decidir la tècnica a emprar per detectar bombetes	1 dia	mar 29/03/11	mié 30/03/11
Primera versió del codi de la MotaPC	5 dies	mié 30/03/11	mié 06/04/11
Primera versió del codi de la Mota	5 **dies**	**mié 06/04/11**	**mié 13/04/11**
Llegir el sensor de llum	1 dia	mié 06/04/11	jue 07/04/11
Llegir el sensor de temperatura	2 dies	jue 07/04/11	lun 11/04/11
rutina de detecció d'estat	1 dia	lun 11/04/11	mar 12/04/11
rutina d'enviament d'alertes	1 dia	mar 12/04/11	mié 13/04/11
Primera versió del codi del PC	5 **dies**	**mié 13/04/11**	**mié 20/04/11**
disseny del formulari	2 dies	mié 13/04/11	vie 15/04/11
rutines tractament informació	1 dia	vie 15/04/11	lun 18/04/11
rutines recepció de missatges	1 dia	lun 18/04/11	mar 19/04/11
rutines enviament de missatges	1 dia	mar 19/04/11	mié 20/04/11
Proves inicials	1 dia	mié 20/04/11	jue 21/04/11
Elaboració del document adjunt al codi	2 dies	jue 21/04/11	lun 25/04/11
Elaboració del document d'autoavaluació.	1 dia	lun 25/04/11	mar 26/04/11
Lliurament de codi i documents	0 dies	mar 26/04/11	mar 26/04/11
Entrega del codi (2/2)	23,88 **dies**	**mié 27/04/11**	**lun 30/05/11**
Implementació de les funcionalitats pendents	5 dies	mié 27/04/11	mar 03/05/11
Estudi dels bugs: Corregir-los	2 dies	mar 03/05/11	jue 05/05/11
versió final del codi de la MotaPC	5 dies	jue 05/05/11	jue 12/05/11
versió final del codi de la Mota	5 **dies**	**jue 12/05/11**	**jue 19/05/11**
Llegir el sensor de llum	1 dia	jue 12/05/11	vie 13/05/11
Llegir el sensor de temperatura	2 dies	vie 13/05/11	mar 17/05/11
rutina de detecció d'estat	1 dia	mar 17/05/11	mié 18/05/11
rutina d'enviament d'alertes	1 dia	mié 18/05/11	jue 19/05/11
versió final del codi del PC	4 **dies**	**jue 19/05/11**	**mié 25/05/11**
disseny del formulari	1 dia	jue 19/05/11	vie 20/05/11
rutines tractament informació	1 dia	vie 20/05/11	lun 23/05/11
rutines recepció de missatges	1 dia	lun 23/05/11	mar 24/05/11
rutines enviament de missatges	1 dia	mar 24/05/11	mié 25/05/11
Proves finals i identificació d'errors	1 dia	mié 25/05/11	jue 26/05/11
Elaboració del document final adjunt al codi	1 dia	jue 26/05/11	vie 27/05/11
Elaboració del document final d'autoavaluació	1 dia	vie 27/05/11	lun 30/05/11
Lliurament final de codi i documents	0 dies	lun 30/05/11	lun 30/05/11
Memòria del TFC	39 **dies**	**lun 18/04/11**	**vie 10/06/11**
Realitzar el primer esborrany	29 dies	lun 18/04/11	jue 26/05/11
Lliurar esborrany a consultor	0 dies	vie 27/05/11	vie 27/05/11
Correccions segons recomanacions	4 dies	vie 27/05/11	mié 01/06/11
Elaboració de la presentació	6 dies	jue 02/06/11	jue 09/06/11
Lliurament final de la memòria i presentació.	0 dies	vie 10/06/11	vie 10/06/11

On es poden veure 4 fites ven diferenciades i que les vam fer coincidir amb els lliuraments que s'havien de realitzar. Detallem una mica les tasques:

1.5.1. Fita 1: Pla de treball i planificació del TFC (Del 7 al 25 de març)

- Conèixer les possibilitats de les motes i de TinyOS. Per aconseguir-ho cal llegir la wiki [1] de l'assignatura i de TinyOS [2].
- Pensar en una proposta de treball tenint en compte l'apres a l'apartat anterior i exposar-la al consultor.
- Preparació de l'entorn de treball. Instal·lació de tot el software necessari per programar les Motes i el sistema que es vol desenvolupar.
- Realitzar tutorials de TinyOS per anar agafant soltesa amb la plataforma[2].
- Creació i lliurament del document “Pla de treball”, aquest document.

1.5.2. Fita 2: Entrega del codi (1/2) (Del 25 de març al 26 d'abril)

- Proves amb els sensors a utilitzar. Per poder realitzar el software de les Motes és necessari conèixer amb força detalls el funcionament dels sensors. Anotar mesures realitzades per adjuntar a la memòria.
- Decidir la tècnica a emprar per detectar bombetes enceses amb el sensor de temperatura.
- Elaborar una primera versió “esquelet” del codi de la MotaPC.
- Elaborar una primera versió “esquelet” del codi de la Mota que situarem a les faroles.
  - Realitzar un codi inicial per llegir del sensor de Llum.
  - Realitzar un codi inicial per llegir del sensor de Temperatura.
  - Escriure el codi inicial per fer les rutines de detecció de l'estat de la farola.
  - Escriure el codi inicial per fer les rutines d'enviament de missatges.
- Elaborar una primera versió “esquelet” del codi que executarà el PC per mostrar les dades.
  - Fer un primer disseny del formulari gràfic
  - Realitzar una primera versió de les rutines de tractament de la informació.
  - Realitzar una primera versió de les rutines de recepció de missatges.
  - Realitzar una primera versió de les rutines d'enviament de missatges.
- Proves inicials i intent de correcció de problemes (bugs).
- Lliurament de tot el codi desenvolupat encara que no realitzi totes les funcions.
- Elaboració del document adjunt al codi on es descrigui l'aplicació, s'informi dels bugs, s'informi de les funcionalitats pendents per implementar i que inclogui les instruccions de funcionament i compilació.
- Elaboració del document d'autoavaluació.

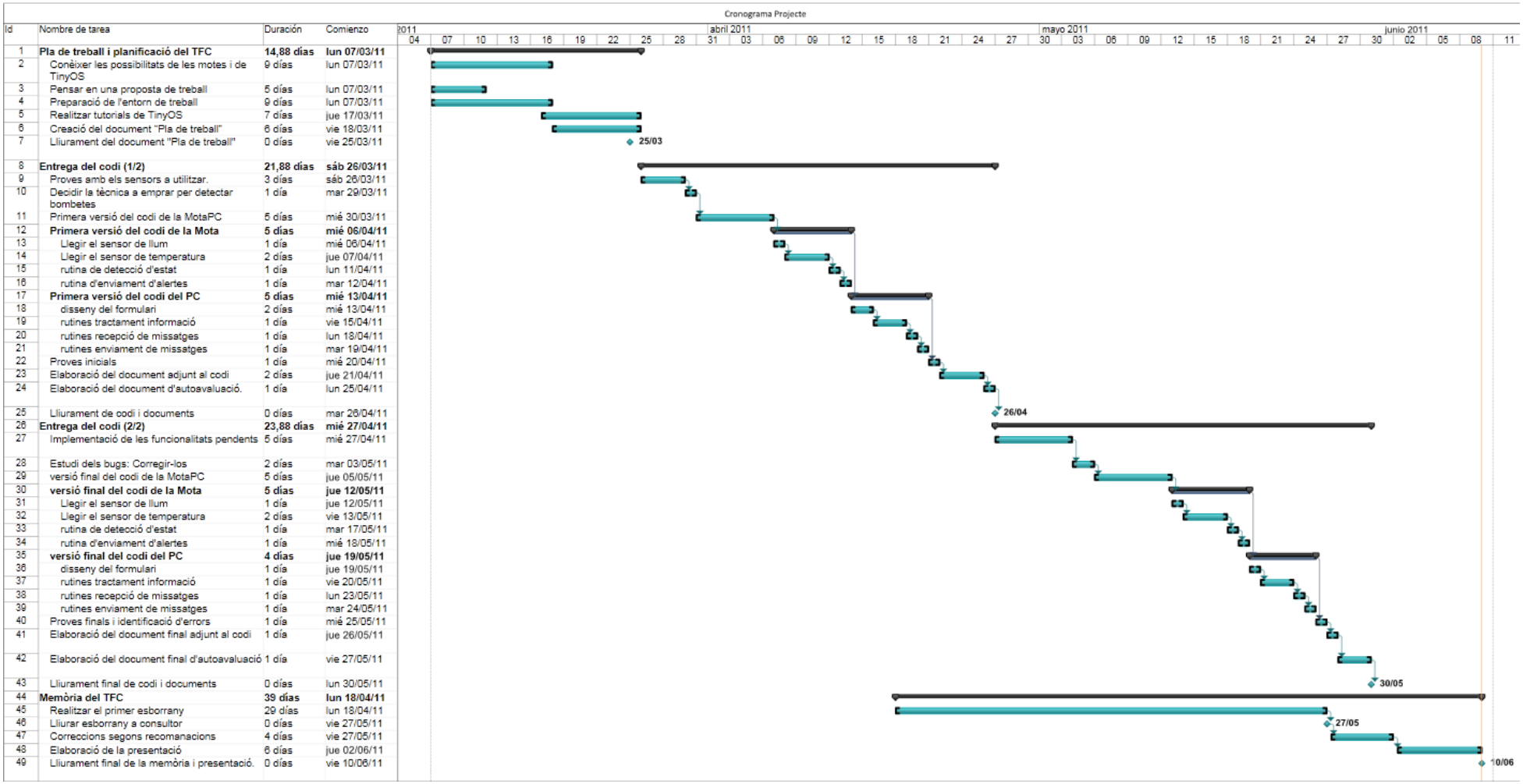
1.5.3. Fita 3: Entrega del codi (2/2) (Del 26 d'abril al 31 de maig)

- Implementació de les funcionalitats pendents si n'hi han pendents.
- Estudi dels bugs: Corregir-los de forma definitiva.
- Elaborar la versió final del codi de la MotaPC.
- Elaborar la versió final del codi de la Mota que situarem a les faroles.
  - Finalitzar el codi per llegir del sensor de Llum.
  - Finalitzar el codi per llegir del sensor de Temperatura.
  - Finalitzar el codi per fer les rutines de detecció de l'estat de la farola.
  - Finalitzar el codi per fer les rutines d'enviament de missatges.
- Elaborar la versió final del codi que executarà el PC per mostrar les dades.
  - Finalitzar el disseny del formulari gràfic.
  - Finalitzar la versió de les rutines de tractament de la informació.
  - Finalitzar la versió de les rutines de recepció de missatges.
  - Finalitzar la versió de les rutines d'enviament de missatges.
- Proves finals i identificació, localització i documentació dels errors que no s'han pogut corregir.
- Lliurament final de tot el codi desenvolupat.
- Elaboració del document adjunt al codi on es descrigui l'aplicació, s'informi dels bugs identificats i que inclogui les instruccions de funcionament i compilació.
- Elaboració del document d'autoavaluació.

1.5.4. Fita 4: Memòria del TFC (Del 17 d'abril al 10 de juny)

- Realitzar el primer esborrany de la memòria del TFC.
- Lliurar esborrany a consultor per la seva revisió (unes 2 setmanes abans del lliurament final).
- Correccions segons recomanacions del consultor.
- Elaboració de la presentació.
- Lliurament final de la memòria i presentació.

1.5.5. Cronograma



Id	Nombre de tarea	Duración	Comienzo
1	**Pla de treball i planificació del TFC**	**14,88 días**	**lun 07/03/11**
2	Conèixer les possibilitats de les motes i de TinyOS	9 días	lun 07/03/11
3	Pensar en una proposta de treball	5 días	lun 07/03/11
4	Preparació de l'entorn de treball	9 días	lun 07/03/11
5	Realitzar tutorials de TinyOS	7 días	jue 17/03/11
6	Creació del document "Pla de treball"	6 días	vie 18/03/11
7	Lliurament del document "Pla de treball"	0 días	vie 25/03/11
8	**Entrega del codi (1/2)**	**21,88 días**	**sáb 26/03/11**
9	Proves amb els sensors a utilitzar.	3 días	sáb 26/03/11
10	Decidir la tècnica a emprar per detectar bombetes	1 día	mar 29/03/11
11	Primera versió del codi de la MotaPC	5 días	mié 30/03/11
12	**Primera versió del codi de la Mota**	**5 días**	**mié 06/04/11**
13	Llegir el sensor de llum	1 día	mié 06/04/11
14	Llegir el sensor de temperatura	2 días	jue 07/04/11
15	rutina de detecció d'estat	1 día	lun 11/04/11
16	rutina d'enviament d'alertes	1 día	mar 12/04/11
17	**Primera versió del codi del PC**	**5 días**	**mié 13/04/11**
18	disseny del formulari	2 días	mié 13/04/11
19	rutines tractament informació	1 día	vie 15/04/11
20	rutines recepció de missatges	1 día	lun 18/04/11
21	rutines enviament de missatges	1 día	mar 19/04/11
22	Proves inicials	1 día	mié 20/04/11
23	Elaboració del document adjunt al codi	2 días	jue 21/04/11
24	Elaboració del document d'autoavaluació.	1 día	lun 25/04/11
25	Lliurament de codi i documents	0 días	mar 26/04/11
26	**Entrega del codi (2/2)**	**23,88 días**	**mié 27/04/11**
27	Implementació de les funcionalitats pendents	5 días	mié 27/04/11
28	Estudi dels bugs: Corregir-los	2 días	mar 03/05/11
29	versió final del codi de la MotaPC	5 días	jue 05/05/11
30	**versió final del codi de la Mota**	**5 días**	**jue 12/05/11**
31	Llegir el sensor de llum	1 día	jue 12/05/11
32	Llegir el sensor de temperatura	2 días	vie 13/05/11
33	rutina de detecció d'estat	1 día	mar 17/05/11
34	rutina d'enviament d'alertes	1 día	mié 18/05/11
35	**versió final del codi del PC**	**4 días**	**jue 19/05/11**
36	disseny del formulari	1 día	jue 19/05/11
37	rutines tractament informació	1 día	vie 20/05/11
38	rutines recepció de missatges	1 día	lun 23/05/11
39	rutines enviament de missatges	1 día	mar 24/05/11
40	Proves finals i identificació d'errors	1 día	mié 25/05/11
41	Elaboració del document final adjunt al codi	1 día	jue 26/05/11
42	Elaboració del document final d'autoavaluació	1 día	vie 27/05/11
43	Lliurament final de codi i documents	0 días	lun 30/05/11
44	**Memòria del TFC**	**39 días**	**lun 18/04/11**
45	Realitzar el primer esborrany	29 días	lun 18/04/11
46	Lliurar esborrany a consultor	0 días	vie 27/05/11
47	Correccions segons recomanacions	4 días	vie 27/05/11
48	Elaboració de la presentació	6 días	jue 02/06/11
49	Lliurament final de la memòria i presentació.	0 días	vie 10/06/11

Il·lustració 1.5.5.1:Cronograma

1.6. Recursos emprats

En el desenvolupament d'aquest TFC s'han fet servir gran varietat d'eines, algunes d'elles tan peculiars com un termòmetre de jardí o un termòmetre de laboratori. Aquest darrer mai el vaig poder fer servir ja que es va trencar abans de poder fer les seves primeres mesures.

Separarem les eines utilitzades en dos tipus ben diferenciats: Hardware i Software. Dins de hardware inclourem tot allò que és material sense fer un ús estricte de la paraula.

1.6.1. Recursos de Hardware

--Un kit de dues Motes model COU 1_2 24.

Il·lustració 1.6.1.1:Mota COU_1_2_24 A2

--Un parell de bateries recarregables del tipus AA.

Il·lustració 1.6.1.2:Piles AA

--Un PC de sobretaula amb les següents característiques:

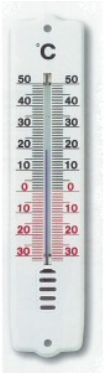
AMD phenom II X2 a 3,2GHz 4GB RAM, i Windows 7 64 bits.

Il·lustració 1.6.1.3:CPU AMD

--Bombetes de diferents tecnologies per fer les proves del concepte.

Il·lustració 1.6.1.4:Bombetes

--Termòmetre d'exteriors per fer mesures i calibrar el sensor de temperatura.



Il·lustració 1.6.1.5:Termòmetre

1.6.2. Recursos de Software

--VMware Player [9] per poder córrer de forma virtual el sistema operatiu recomanat per TinyOS Ubuntu 10.4. Es va optar per aquesta solució donat que era la única que ens garantia un funcionament correcte dels ports USB virtualitzats. Vam provar també amb VirtualBox però no funcionava correctament el redireccionament dels ports USB, imprescindible per poder programar les motes.

--Ubuntu 10.4 [10] Sistema operatiu del tipus Linux. Recomanat per TinyOS com a sistema operatiu suportat per la plataforma.

--Eclipse Galileo IDE[11] amb JDT[12] i CDT[13]. Plataforma de programació utilitzada per editar i programar codi en múltiples llenguatges de programació. Es caracteritza per ser molt modular i de codi obert. Aquestes característiques fan que sigui una plataforma IDE dels mes utilitzats arreu del món i que gairebé tingui mòduls per tots els llenguatges de programació existents. TinyOS no és una excepció i el llenguatge utilitzat per les motes, nesC, també disposa de mòduls per poder editar el codi d'una manera còmoda i amb funcionalitats de ressaltat de funcions conegudes o diferenciació en color de les variables. Nosaltres l'hem instal·lat amb JDT (Java Development Tools) i CDT (C/C++ Development Tools) ja que és un dels són requisit per poder instal·lar el mòdul de TinyOS que detallem a continuació.

--Yeti 2 mòdul per Eclipse [14]. Com parlàvem en l'apartat anterior, Eclipse disposa de diversos mòduls per fer més còmoda la programació amb TinyOS. Un d'aquests mòduls és Yeti 2 creat pel grup de recerca “Distributed Computing Group” de l'institut federal de tecnologia de Zurich (ETH). Entre d'altre funcionalitats, les mes destacades ser

--TinyOS 2.1.1 [2]. Es tracta del sistema operatiu que fan servir les motes COU_1_2 24. Disposa de llicència BSD i està dissenyat per una gran varietat de dispositius sense fils de baix consum. Mes endavant, a l'apartat 2.1, en parlarem amb més detall d'aquest sistema operatiu.

--TinyOS modificat per les motes COU24 [15]. Donat que la mota que farem servir no es troba al paquet oficial de TinyOS, es necessari descarregar la carpeta del TinyOS lleugerament modificada i que contempla les llibreries necessàries pel nostre dispositiu.

--Meshnetics serial programmer for Linux (meshprog) [16]. Es tracta del programa que farem sevir per pujar el codi nesC compilat a les motes. Originalment aquest programa es va dissenyar per programar les motes ZigBit però també serveix pel nostre tipus de dispositiu.

1.7. Productes obtinguts

La realització d'aquest projecte del Treball de Final de Carrera ha donat lloc al lliurament del següents productes:

--Aquest document en forma de **memòria** amb tot tipus de detalls referents al projecte realitzat i al seguiment de l'assignatura TFC al llarg de tot el semestre.

--El codi font de l'aplicació ***TFCFarola*** programat per nosaltres pel sistema operatiu TinyOS i especialment pel que executen les motes COU_1_2 24. Aquesta és l'aplicació que s'ha de carregar a les motes que vulguem destinar a les faroles per monitoritzar el seu estat.

--El codi font de l'aplicació ***TFCBase*** programat per la **Universitat de Berkeley** pel sistema operatiu TinyOS. Aquesta aplicació no és de creació nostre però hem decidit lliurar-la juntament amb la resta de lliuraments per facilitar el procés de posta en marxa del sistema. És l'aplicació que ha d'executar la mota connectada al PC base que executarà la GUI.

--El codi font de l'aplicació ***TFCOrdinador*** programat per nosaltres per la màquina virtual de Java. Es tracta del codi del programa que s'executarà al PC que té la mota Base connectada al USB. A través d'aquesta interfície gràfica, l'usuari podrà comprovar l'estat de les motes farola i canviar els paràmetres de monitorització.

--Un document en forma de **presentació** que explica de forma clara, senzilla i entenedora la feina realitzada i els resultats obtinguts. És un document que sintetitza tota la feina desenvolupada.

1.8. Descripció dels següents capítols

En els següents capítols d'aquest document es parlarà més detalladament dels aspectes més rellevants de les tecnologies utilitzades, del procés de creació del producte, del seu disseny, del seu funcionament intern, de les proves realitzades i de la manera de fer-lo servir. També parlarem de quina és la seva projecció en el mercat, quin és el valor econòmic de tot plegat i quines són les conclusions a les que hem arribat en finalitzar aquest projecte:

Al capítol 2 es fa un petit resum de quin és l'estat actual de TinyOS i tot el que l'envolta: Sistemes de sensors, protocols de comunicació, implicació en desenvolupament de millores, grans projectes que en fan ús... També en parlarem de projectes amb certa similitud al nostre i dels sectors als que va dirigit aquest tipus de desenvolupament especific.

Al capítol 3 explicarem el disseny del nostre sistema final i les decisions que s'han hagut de prendre per tal de desenvolupar-lo. Farem el mateix amb cadascuna de les parts independents del sistema: TFCFarola i TFCOrdinador.

Al capítol 4 farem una descripció més tècnica i detallada del sistema final, del TFCFarola i del TFCOrdinador. Aquesta descripció no inclourà codi però si que proporcionarem els diagrames i les estructures perquè qualsevol entès en la matèria pugui fer-se una idea tècnica del funcionament de la xarxa de sensors creada.

Al capítol 5 parlarem de les diferents proves que hem anat realitzant per comprovar el correcte funcionament del producte i per poder implementar les funcionalitats que ofereix el sistema.

Al capítol 6 es fa un petit estudi de quina és la projecció del nostre sistema, quins són els punts forts i quins són els més febles. És un bon apartat per que el lector sàpiga si l'autor opina que el projecte pot anar més enllà del prototip.

Al capítol 7 farem una valoració econòmica orientativa del cost que suposaria portar aquest projecte a la pràctica.

Entrats al capítol 8 es recolliran les conclusions personals sobre el projecte. Es compararan els objectius inicialment declarats al pla de treball amb els realment aconseguits. També es proposaran alguns exemples de futures ampliacions i millores per qui vulgui continuar amb la mateixa línia de treball o desenvolupament.

I ja per finalitzar aquest apartat, als capítols 9, 10 i 11 trobarem el glossari de termes, la bibliografia utilitzada i els documents annexos

2. Antecedents

Podem separar les tecnologies que fem servir en aquest projecte en quatre grans blocs: Xarxes de sensors sense fils, motes, sistemes operatius i protocols de comunicacions. Parlarem de com estan aquests grans blocs avui en dia i ens traslladarem al passat per conèixer quins són els orígens. Respecte al nostre projecte, parlarem especialment de les xarxes de monitorització energètica, de la mota COU_1_2_24, de TinyOS i del protocol ZigBee.

2.1. Estat de l'art

2.1.1. Xarxes de sensors sense fils[17]

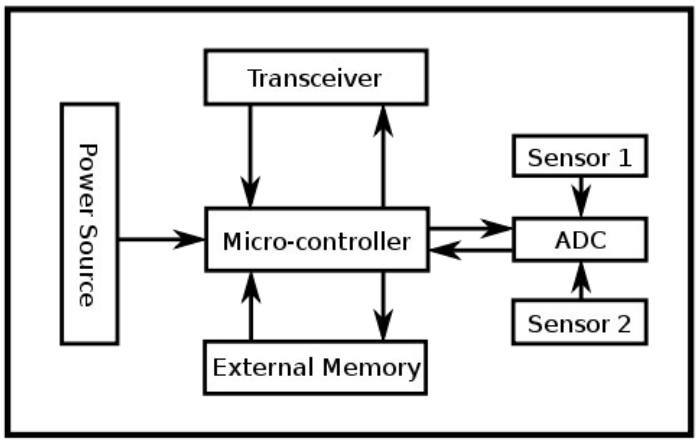
Són xarxes que estan formades per un grup de sensors (nodes) amb certes capacitats sensitives i de computació i que disposen de comunicació sense fil i que poden formar en diverses topologies. També conegudes per les seves sigles en anglès **WSN**, el seu principal comés és l'obtenció i tractament de dades provinents d'elements actius propers als nodes. Aquests sensors es caracteritzen per la gestió eficient de la energia que els alimenta i per la seva mida reduïda que els fa molt versàtils per les diferents tasques que se'ls hi poden demanar.

Com la majoria d'avenços tecnològics, tenen el seu origen en iniciatives militars i avui en dia s'utilitzen en una gran diversitat de camps com poden ser l'aplicació ambiental, sanitària, domèstica, comercial, industrial o com el cas que ens ocupa, l'aplicació energètica.

En quant a l'**aplicació energètica**, avui en dia es fan servir per monitoritzar el consum energètic i l'eficiència de fàbriques, oficines, llars i en el cas del nostre projecte, ciutats.

2.1.2. Motes[18]

Es diuen motes els nodes d'una xarxa de sensors sense fils que són capaços de computar, recollir informació i comunicar-se amb altres nodes de la xarxa. Els components principals d'una mota són un microcontrolador, un transceptor de radio, memòria externa, una font d'alimentació i un o més sensors traduïts generalment per un ADC. A la següent imatge podem veure un esquema típic d'aquests components:



Il·lustració 2.1.2.1:Arquitectura típica d'una mota

Tot i que els sensors sense fils ha existit durant dècades, el concepte modern de mota el trobem per primera vegada el 1998 al projecte “Smartdust” [19] de la Universitat de California a Berkeley i al projecte “Sensor Webs Project” de la NASA. Com a curiositat, el terme mota li van posar els creadors de “Smartdust”, que en anglès significa “pols intel·ligent”, comparant-les metafòricament amb les motes de pols. Tot i que els projectes van finalitzar, aquests van donar pas a més projectes d'investigació i gracies a la llei de Moore, les motes han anat incrementant les seves capacitats i reduint el consum. Com a exemple de motes actualment desenvolupades podem trobar les de la següent taula:

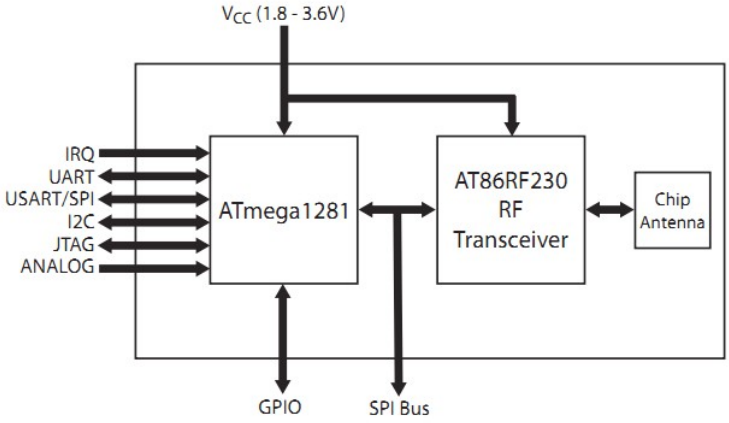
Sensor Node Name	Microcontroller	Transceiver	Program+Data Memory	External Memory	Programming	Remarks
AVRraven Atmel AVR#Raven v	AtMega1284p + ATmega3290p	AT86RF230	128 Kbytes + 16 Kbytes	256 kB?	C	
COOKIES[1]	ADUC841	ETRX2 TELEGESIS	4 Kbytes + 62 Kbytes	4 Mbit	C	Platform with hardware reconfigurability (Spartan 3FPGA based)
BEAN	MSP430F169	CC1000 (300-1000 MHz) with 78.6 kbit/s		4 Mbit		YATOS Support
BTnode	Atmel ATmega 128L (8 MHz @ 8 MIPS)	Chipcon CC1000 (433-915 MHz) and Bluetooth (2.4 GHz)	64+180 K RAM	128K FLASH ROM, 4K EEPROM	C and nesC Programming	BTnut and TinyOS support
COTS	ATMEL Microcontroller 916 MHz					
Dot	ATMEGA163		1K RAM	8-16K Flash	weC	
EPIC mote	Texas Instruments MSP430 microcontroller	250 kbit/s 2.4 GHz IEEE 802.15.4 Chipcon Wireless Transceiver	10k RAM	48k Flash		TinyOS
Egs[2]	ARM Cortex M3	CC2520, Mitsumi's class 2 Bluetooth module		2 Gbit		TinyOS
Eyes	MSP430F149	TR1001		8 Mbit		PeerOS Support
EyesIFX v1	MSP430F149	TDA5250 (868 MHz) FSK		8 Mbit		TinyOS Support
EyesIFX v2	MSP430F1611	TDA5250 (868 MHz) FSK		8 Mbit		TinyOS Support
FlatMesh FM1	16 MHz	802.15.4-compliant		660 sensor readings	Over-air control	Commercial system, for digital sensors
FlatMesh FM2	16 MHz	802.15.4-compliant		660 sensor readings	Over-air control	Commercial system, built-in tilt sensor
GWnode	PIC18LF8722	BiM (173 MHz) FSK	64k RAM	128k flash	C	Custom OS
IMote	ARM core 12 MHz	Bluetooth with the range of 30 m	64K SRAM	512K Flash		TinyOS Support
IMote 1.0	ARM 7TDMI 12-48 MHz	Bluetooth with the range of 30 m	64K SRAM	512K Flash		TinyOS Support

Sensor Node Name	Microcontroller	Transceiver	Program+Data Memory	External Memory	Programming	Remarks
IMote 2.0	Marvell PXA271 ARM 11-400 MHz	TI CC2420 802.15.4/ZigBee compliant radio	32 MB SRAM	32 MB Flash		Microsoft .NET Micro, Linux, TinyOS Support
INDriya_CS_03A14 [3]	Atmel ATmega 128L [4]	IEEE 802.15.4 compliant XBee radios	128 KB FLASH + 4 KB RAM	Expansion available	C-programming & nesC compliant	Comprehensive Sensor mote with: Ambient Light,temperature, accelerometer,JPEG camera,PIR, sound sensor, TinyOS TinyOS compliant, IPv6 network supportive stacks for internetworking & so on.
Iris Mote	ATmega 1281	Atmel AT86RF230 802.15.4/ZigBee compliant radio	8K RAM	128K Flash	nesC	TinyOS, MoteWorks Support
KMote	TI MSP430	250 kbit/s 2.4 GHz IEEE 802.15.4 Chipcon Wireless Transceiver	10k RAM	48k Flash		TinyOS and SOS Support
Mica	ATmega 103 4 MHz 8-bit CPU	RFM TR1000 radio 50 kbit/s	128+4K RAM	512K Flash	nesC Programming	TinyOS Support
Mica2	ATMEGA 128L	Chipcon 868/916 MHz	4K RAM	128K Flash		TinyOS, SOS and MantisOS Support
Mica2Dot	ATMEGA 128		4K RAM	128K Flash		
MicaZ	ATMEGA 128	TI CC2420 802.15.4/ZigBee compliant radio	4K RAM	128K Flash	nesC	TinyOS, SOS, MantisOS and Nano-RK Support
Monnit WIT	TI CC1110	868/900 MHz	4K RAM		C#	multi sensor boards
Mulle	Renesas M16C	Atmel AT86RF230 802.15.4 / Bluetooth 2.0	31K RAM	384K+4K Flash, 2 MB EEPROM	nesC, C programming	Contiki, TinyOS, lwIP: TCP/IP and Bluetooth Profiles: LAP, DUN, PAN and SPP Support
NeoMote	ATmega 128L	TI CC2420 802.15.4/ZigBee compliant radio	4K RAM	128K Flash	nesC	TinyOS, SOS, MantisOS, Nano-RK and Xmesh Support, Industrial end-use product.
Nymph	ATMEGA128L	CC1000		64 kB EEPROM		MantisOS Support
Redbee	MC13224V	2.4 GHz 802.15.4	96 KB RAM + 120 KB Flash		GCC (see mc1322x.devl.org), IAR	Contiki; standalone
Rene	ATMEL8535	916 MHz radio with bandwidth of 10 kbit/s	512 bytes RAM	8K Flash		TinyOS Support
SenseNode	MSP430F1611	Chipcon CC2420	10K RAM	48K Flash	C and NesC programming	GenOS and TinyOS Support
Shimmer	MSP430F1611	802.15.4 Shimmer SR7 (TI CC2420)	48 KB Flash 10 KB RAM	2 GB microSD Card	nes C and C Programming	TinyOS Support. Built in 3 Axis Accel, Tilt/Vib Sensor. Full range of expansion modules.
SunSPOT	ARM 920T	802.15.4	512K RAM	4 MB Flash	Java	Squawk Java ME Virtual Machine
Telos	MSP430		2K RAM			

Sensor Node Name	Microcontroller	Transceiver	Program+Data Memory	External Memory	Programming	Remarks
TelosB	Texas Instruments MSP430 microcontroller	250 kbit/s 2.4 GHz IEEE 802.15.4 Chipcon Wireless Transceiver	10k RAM	48k Flash		Contiki, TinyOS, SOS and MantisOS Support
Tinynode	Texas Instruments MSP430 microcontroller	Semtech SX1211	8K RAM	512K Flash	C Programming	TinyOS
T-Mote Sky	Texas Instruments MSP430 microcontroller	250 kbit/s 2.4 GHz IEEE 802.15.4 Chipcon Wireless Transceiver	10k RAM	48k Flash		Contiki, TinyOS, SOS and MantisOS Support
Waspmote	Atmel ATmega 1281	ZigBee/802.15.4/DigiMesh/RF , 2.4 GHz/868/900 MHz	8K SRAM	128K FLASH ROM, 4K EEPROM, 2 GB SD card	C/Processing	GPRS, Bluetooth, GPS modules, sensor boards..
weC	Atmel AVR AT90S2313	RFM TR1000 RF				
Wireless RS485	Atmega 128L	Chipcon CC2420 + Amplifier 250 kbit/s 2.4 GHz IEEE 802.15.4	4k RAM	128k Flash		Xmesh, TinyOS
XYZ	ML67 series ARM/THUMB microcontroller	CC2420 Zigbee compliant radio from Chipcon	32K RAM	256K Flash	C Programming	SOS Operating System Support
Zolertia Z1	Texas Instruments M	Chipcon CC2420 2.4 GHz IEEE 802.15.4 Wireless Transceiver	8 KB RAM	92 KB Flash	C, nesC	Contiki and TinyOS Support. 16 Mbit external flash + 2 digital on-board sensors
FireFly	Atmel ATmega 1281	Chipcon CC2420	8K RAM	128K FLASH ROM, 4K EEPROM	C Programming	Nano-RK RTOS Support
Ubimote1	TI's CC2430 SOC based on 8051 Core	TI's CC2430	8K RAM	128K FLASH ROM	C Programming	TI's ZStack, TinyOS Support
Ubimote2	TI's MSP430F2618	TI's CC2520	8K RAM	116K FLASH ROM	C Programming	TI's ZStack Support
VEmesh	TI MSP430	Semtech SX1211/1231, TI TRF6903	512B RAM	8K FLASH	Over-the-air Programming	FHSS; Interface to MODBUS, DALI, RS-232/485, TCP/IP

On es veu que hi ha una gran varietat, tant de motes com de microcontroladors, de memòria, de llenguatge de programació i d'aparells de radio.

La mota que fem servir nosaltres, la COU_1_2_24, integra microcontrolador, memòria i transceptor en un mateix mòdul encapsulat, el “ZigBit™ 2.4 GHz Wireless Module ATZB-24-A2”[20]:



Il·lustració 2.1.2.2:Diagrama de blocs ATZB-24-A2

La mota té les següents característiques:

--Microcontrolador ATmega1281 a 4Mhz amb 128KB de flash, 8KB de RAM i 4KB de EEPROM.

--Transceptor AT86RF230 a la banda de 2,4GHz compatible amb els protocols 802.15.4/ZigBee. Fins a 3dBm de potencia de sortida i una sensibilitat de recepció de fins a -101dBm.

--UART USB to SERIE CP2102 com a port de comunicacions sèrie.

--Sensor de temperatura MCP9700.

--Fotosensor PDV-P9003-1.

--Sensor de efecte Hall BU52011HFV.

--Tres díodes LED per utilitzar-los com indicadors.

2.1.3. Sistema operatiu de les WSN

Dins de la diversitat de motes que podem trobar, també trobem una amplia diversitat de sistemes operatius que les governen. Els sistemes operatius de les xarxes de sensors sense fils són menys complexos que els sistemes operatius de propòsit general. Això es degut principalment a dues raons: Primer, les WSN estan dissenyades per una aplicació concreta més que com a una plataforma general i segon perquè la necessitat de baixos costos i baix consum porta a que les motes hagin de fer servir microcontroladors que prescindeixin de mecanismes complexos i innecessaris.

Encara que es possible utilitzar sistemes operatius encastats com eCos o uC/OS a les xarxes de sensors, no en trauríem el màxim partit ja que aquests sistemes van estar pensats amb característiques de temps real i és possiblement TinyOS el primer sistema operatiu que es va dissenyar per les xarxes de sensors sense fils.

TinyOS està basat en un model de programació per esdeveniments en comptes del model multiprocés. El llenguatge de programació per esdeveniments que utilitza es diu nesC. Els programes estan composats de capturadors d'esdeveniments i de tasques amb una semàntica del tipus executar fins completar. Quan es dóna un esdeveniment, com per exemple l'arribada d'un paquet de dades o una lectura dels sensors, TinyOS emet la corresponent senyal d'esdeveniment per poder capturar-la. Les senyals capturades poden llençar fer-se servir per llençar tasques programades anteriorment.

A part de TinyOS, també tenim altres sistemes operatius dissenyats especificament per a xarxes WSN, un exemple el trobem en LiteOS de recent desenvolupament i que proporciona un nivell d'abstracció similar a UNIX i suporta programació en C. Un altre exemple de sistema operatiu encastat dissenyat per WSN's és Contiki, que utilitza un estil de programació més simple en C i que proporciona avenços com 6LoWPAN que és un acrònim de “IPv6 sobre xarxes d'àrea personal de baix consum”.

2.1.4. Estàndards de comunicacions sense fils

Sense adentrar-nos en els detalls de cadascun d'ells, a continuació mostrem els més coneguts:

--IEEE 802.15.4: És l'estàndard que especifica la capa física i la de control d'accés al medi per les xarxes d'area personal de baix cost (LR-WPANs).

--WirelessHART: Basat en 802.15.4 i iniciat al 2004 per HART Communicaction Foundation aquest protocol utilitza una arquitectura tipus “mesh” auto-organitzada, auto-sanejada i sincronitzada en el temps.

--ISA100.11a: estàndard de xarxes sense fils desenvolupat per la Societat d'Automatització Internacional (ISA) i es va dissenyar específicament per sistemes sense fils per a la automatització industrial.

--ZigBee: És l'especificació d'alt nivell per un conjunt de protocols de comunicacions basats en 802.15.4-2003 per fer servir petites ràdios digitals de baix consum. Intenta ser més simple i menys cara que altres WPANs com podria ser Bluetooth. Està enfocada a aplicacions de ràdio freqüència que no requereixen una taxa alta de dades, que necessiten una llarga autonomia de les bateries i que necessiten una xarxa segura.

2.2. Estudi de mercat

Referent a sistemes similars al projecte,hem trobat un parell de exemples. El primer es tracta d'una aplicació comercial anomenada “EcoWizard Energy Monitoring System” [21] i comercialitzada per l'empresa Crossbow que serveix per controlar el consum elèctric a edificis.

El segon exemple es un article universitari anomenat “A WSN-based Testbed for Energy Efficiency in Buildings” [22] que parla d'un sistema domòtic per estalviar energia a edificis.

No obstant el nostre projecte va més encaminat al sector de l'administració pública. Concretament al mobiliari urbà i no al d'edificis.

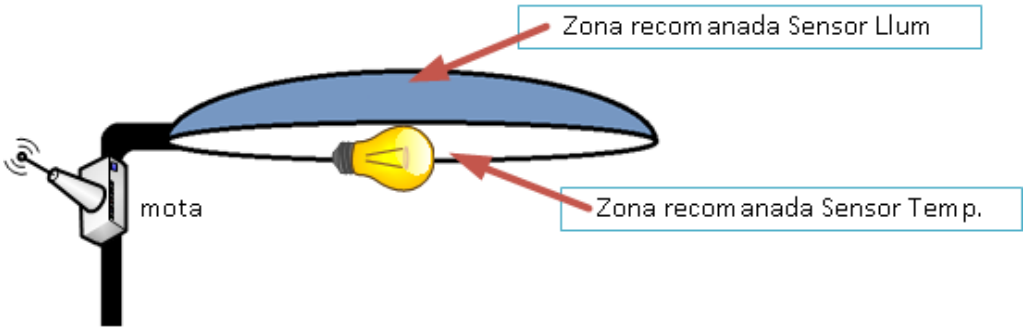
3. Descripció funcional

En aquest apartat intentaré explicar de manera esquemàtica, entenedora i no massa tècnica, el disseny del sistema de “Monitor de l'Estat Operatiu de l'Enllumenat Públic” que hem desenvolupat en aquest projecte de final de carrera. Explicaré quines són les decisions que s'han pres per escollir determinades vies de desenvolupament i quines són les raons per haver-les escollit. Així doncs, intentarem donar una visió global del disseny dels productes obtinguts que més endavant, en el següent capítol, desenvoluparem més tècnicament i de manera més detallada.

3.1. Sistema “Monitor de l'Estat Operatiu de l'Enllumenat Públic”

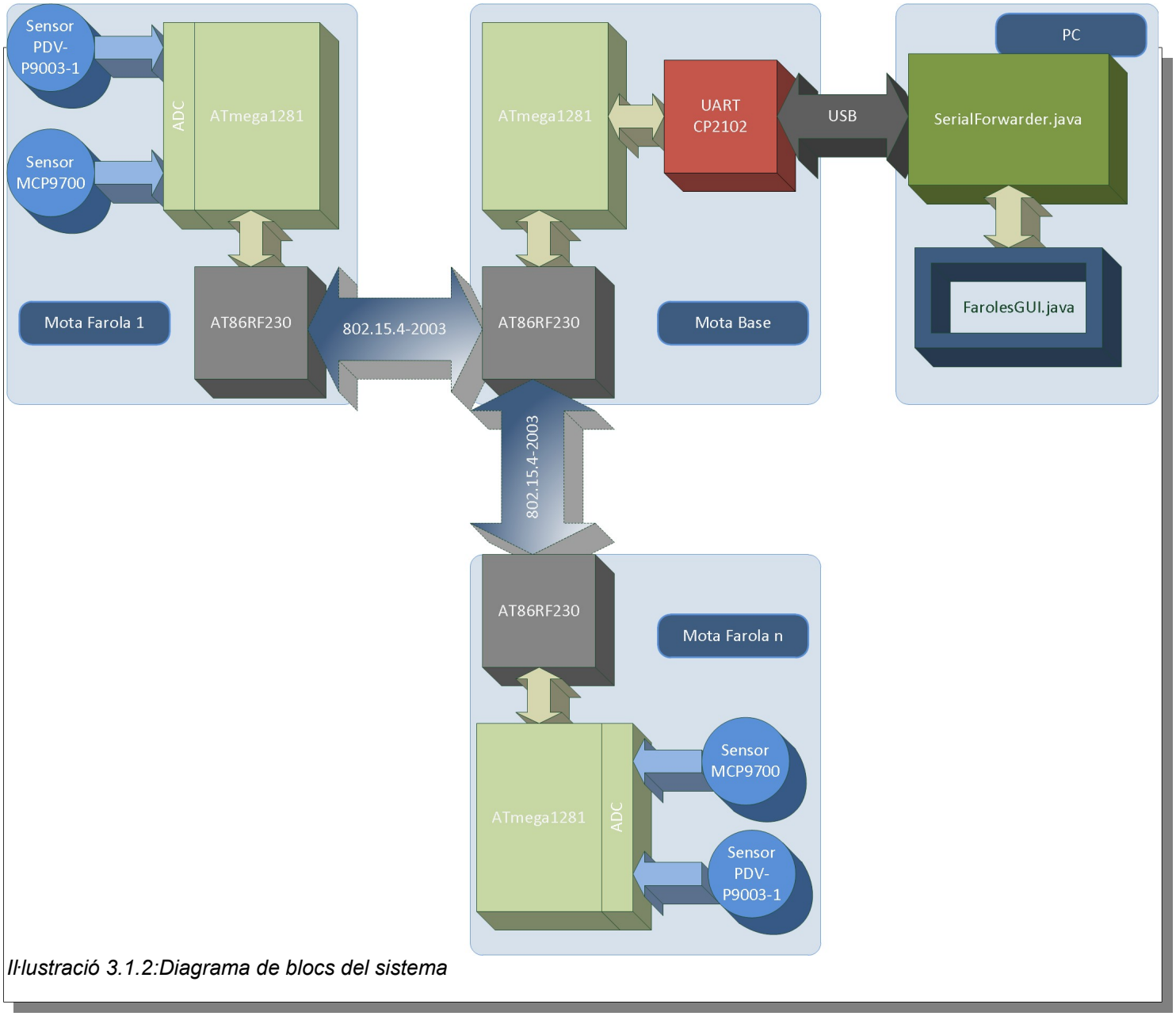
Com ja en parlàvem a la introducció d'aquesta memòria, el nostre sistema proporciona una eina que facilita el control del consum elèctric del mobiliari urbà dedicat a l'enllumenat. Principalment està dissenyat per enllumenat del tipus farola però l'àmbit de monitoratge es pot estendre a qualsevol element d'il·luminació que tingui com a element actiu una làmpada que desprengui un mínim de calor. Les proves de concepte que hem realitzat ens han permès comprovar que fins i tot els elements d'il·luminació mes eficients, energètica-ment parlant, desprenen calor. Per tant no importarà el tipus de làmpada sempre i quan es tingui present quina és la temperatura mitja de treball.

Retornant a la descripció funcional del sistema, s'ha fet servir una topologia de xarxa en forma d'estrella degut a les limitacions de material que teníem (dos motes de les quals una sempre ha de fer de base). Els nodes s'han d'instal·lar a l'element d'il·luminació de tal manera que el sensor de temperatura estigui suficientment a prop de la làmpada com per detectar les variacions de temperatura i suficientment allunyat com per no cremar-se si es tracta d'una làmpada que desprèn molta calor. El sensor de llum ha d'anat situat de tal manera que capti les variacions de la llum ambiental però que no rebi interferències de lluminàries artificials. A la figura 3.1.1 podem veure un exemple d'instal·lació adequada.



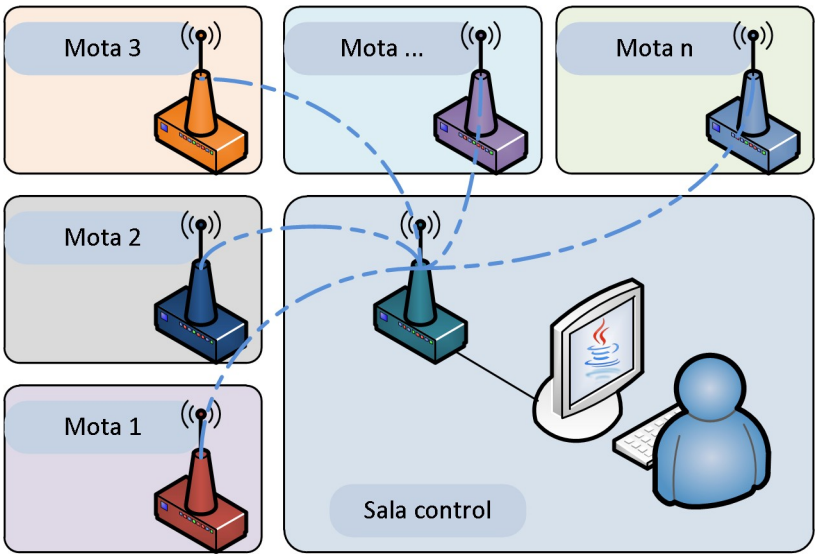
Il·lustració 3.1.1:Recomanacions instal·lació de sensors

Mitjançant els sensors i un sistema de llindars, el sistema comprova l'estat de les faroles i quan es produeix una alerta envia un missatge d'error a la estació base connectada al PC. Aquest PC recull les dades a través del port sèrie i les mostra per pantalla mitjançant una interfície gràfica realitzada amb Java, també facilitada per nosaltres. Aquesta interfície, a més a més de mostrar els missatges d'alerta, també permet a l'usuari la modificació dels llindars de temperatura, de llum i la freqüència de mostreig de les mesures.



Il·lustració 3.1.2:Diagrama de blocs del sistema

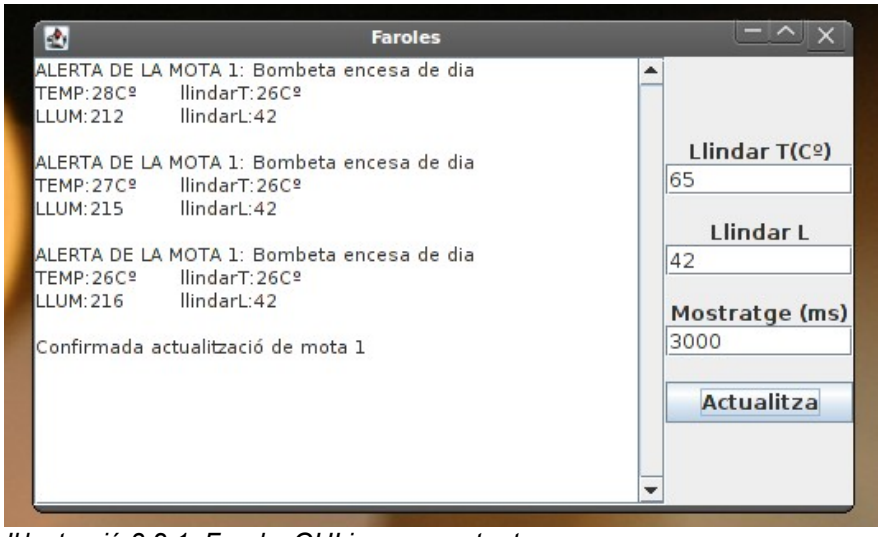
A la figura 3.1.2 podem veure el diagrama de blocs de tot el sistema amb una mica de detall dels elements que el composen. Si ens hi fixem, veurem com al mateix diagrama ja és contempla que la topologia és d'estrella, però perquè ens fem una millor idea, hem creat la figura 3.1.3 on la topologia de la xarxa queda reflectida de manera més clara.



Il·lustració 3.1.3:Topologia Estrella

3.2. TFCOrdinador (PC)

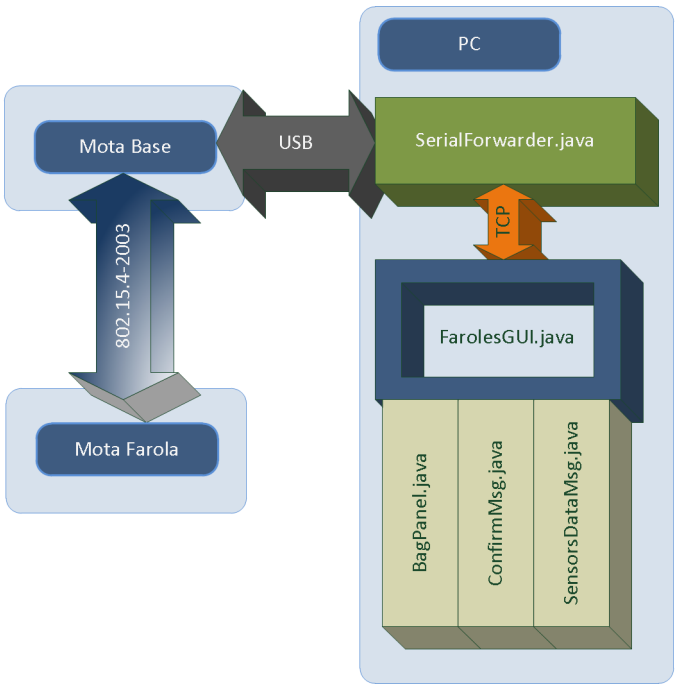
Es tracta del codi del programa que s'executarà al PC que té la mota Base connectada al USB. A través d'aquesta interfície gràfica, l'usuari podrà comprovar l'estat de les motes Farola i canviar els paràmetres abans esmentats. A la figura 3.2.1 veiem la finestra de l'aplicació.



Il·lustració 3.2.1: FarolesGUI.java executant-se

Veiem com a la dreta tenim els camps per introduir els nous llindars i el nou mostratge, i a l'esquerra tenim els missatges que arriben de les motes. Quan es prem el botó actualitza, l'aplicació envia un missatge de broadcast amb els nous llindars i la nova freqüència de

mostratge. Les motes que s'actualitzen correctament responen amb una confirmació com la que es pot veure a la figura. FarolesGUI.java requereix de SerialForwarder.java per funcionar donat que hi connecta pel port TCP 9002 i fa d'intermediari entre la mota base i el nostre programa per rebre i enviar els missatges. A la figura 3.2.2 podem veure les classes dels dos tipus de missatges que tractem i una tercera classe BagPanel.java que la fem servir per facilitar la creació del formulari gràfic.



Il·lustració 3.2.2:Diagrama de blocs de la part PC

3.3. TFCFarola (mota farola)

Es tracta de l'aplicació programada en llenguatge nesC que s'executarà a les motes destinades a les faroles, es a dir les que crearan la xarxa en estrella. L'aplicació principal és FarolaAppC.nc (és l'arxiu de configuració de FarolaC.nc). A nivell funcional, mesura la llum ambiental i la temperatura de la bombeta per determinar el correcte funcionament de la farola. La llum i la temperatura són comparades cada **x** temps amb uns llindars de llum i temperatura introduïts per l'usuari a l'aplicació FarolesGUI.java. L'usuari també pot variar el temps **x** de mostratge per tal d'evitar un consum excessiu (per les necessitats del projecte, no es necessari comprovar la llum cada 3 segons).

Un cop fetes les comparacions, es poden donar 3 estats:

1. **El funcionament es correcte**: Farola encesa de nit, o farola apagada de dia. (estat 0) Indicador VERD.
2. **La farola està encesa de dia.**(estat 1)Indicador TARONJA.
3. **La farola està apagada de nit.**(estat 2)Indicador VERMELL.

Si es donen les condicions 2 i 3, la mota enviarà un missatge de radio cada **x** temps a la

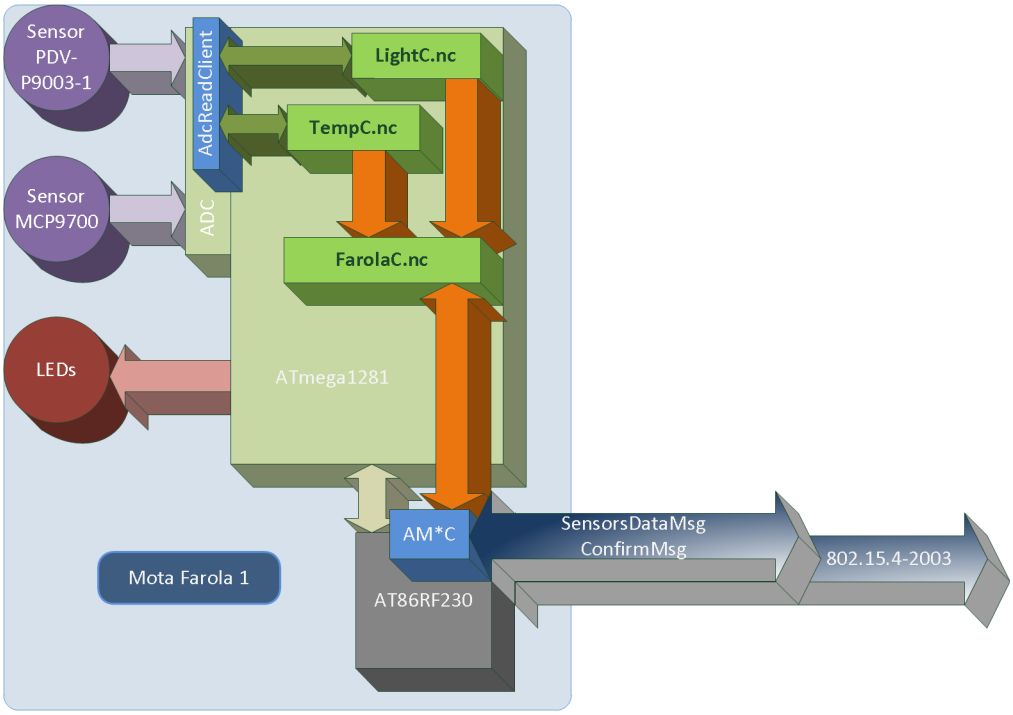
estació base fins que la situació es normalitzi (estat 0).

> **NOTA:**Cal a dir que el fet d'encendre els LED's per indicar els estats a la mota farola, és un fet merament didàctic i perquè podem saber quin és l'estat en tot moment des del laboratori, a la pràctica es convenient gastar el mínim de bateries possibles, pel que tenir un LED encès tot el dia quan estem parlant de consums total del sistema de pocs mA no es viable.

El missatge de radio, conté els camps necessaris per poder identificar la mota en qüestió, el codi d'alerta i informació de les mesures i dels llindars que té configurats en el moment de l'alerta.

Cada vegada que s'actualitza algun dels seus paràmetres i a mode de confirmació per informar a l'usuari, la mota respon amb un missatge del tipus ConfirmMsg (figura 3.2.1). Aquest mètode també ens servirà per comprovar l'estat general de les motes ja que també es pot fer servir a mode de 'ping'.

Si abans vèiem un diagrama de blocs general on apareixia la mota farola a grans trets, ara es el moment de veure un diagrama de blocs amb una mica més de detall per entendre millor el funcionament d'aquest component:



Il·lustració 3.3.1:Diagrama blocs TFCFarola

On a la capa superior podem veure els tres principals components que intervenen a la nostra aplicació (verd) relacionats amb els més significatius components del sistema operatiu (Blau). Els he posat a sobre del hardware perquè aquests components suposen una capa d'abstracció més. Així, els components AM*C (AMSenderC, AMReceiverC ...), estan a sobre del dispositiu de ràdio perquè suposen una abstracció d'aquest... El mateix amb el component de lectura del ADC, AdcReadClient. Espero que sigui fàcil d'entendre. A grans trets:

-LightC: Llegeix les dades del sensor de llum i les ofereix a FarolaC

-TemperatureC: Llegeix les dades del sensor de temperatura i les ofereix a FarolaC.

-FarolaC: Tracta les dades i està connectat als serveis d'AM*C per enviar o rebre missatges.

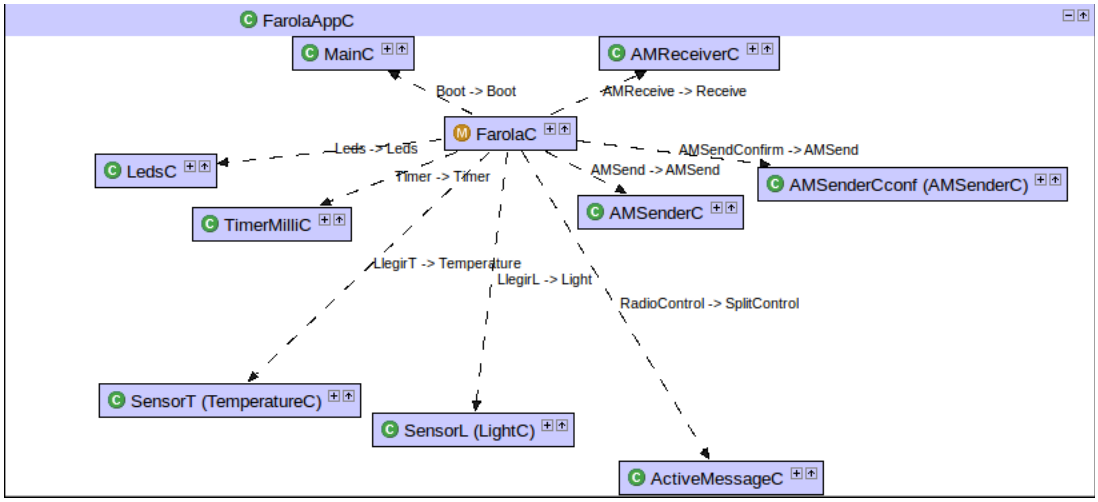
4. Descripció detallada

4.1. TFCFarola (mota farola)

Com ja hem introduït en l'apartat anterior, aquesta és l'aplicació encarregada d'executar la part del sistema de monitoratge destinat a les motes farola. Consta de tres components i els seus respectius arxius de configuració:

4.1.1. FarolaC

Component principal de l'aplicació. De fet, el seu arxiu de configuració **FarolaAppC** és el que es considera principal ja que al ser l'arxiu de configuració del nostre programa principal, té tots els arxius de l'aplicació relacionats en les connexions. A la figura 4.1.1.1 podem veure el diagrama d'aquestes connexions.



Il·lustració 4.1.1.1:Diagrama FarolaAppC

Com podem veure, FarolaC fa servir una gran diversitat de components. Les fletxes cap a fora indiquen que el mòdul FarolaC fa servir interfícies dels components cap a on van dirigides les fletxes, així, per exemple, des de FarolaC fem servir les interfícies de lectura dels components TemperatureC i LightC (LlegirT i LlegirL connectats a les interfícies TemperatureC.Temperature i LightC.Light respectivament).

A continuació detallem totes les connexions realitzades:

-FarolaC.Boot ->MainC.Boot: És el component que ens indica que la mota ha arrencat i ens dóna pas per començar a executar codi.

-FarolaC.Leds -> LedsC.Leds: Component per poder fer servir els leds.

-FarolaC.Timer->TimerMilliC.Timer: Component per poder fer servir el timer.

-FarolaC.LlegirT->SensorT.Temperature: Component programat per nosaltres i que ens ofereix la lectura de la temperatura.

-FarolaC.LlegirL->SensorL.Light: Component programat per nosaltres i que ens ofereix la

lectura de la llum.

-FarolaC.RadioControl->ActiveMessageC.SplitControl: Connectem splitcontrol a la capa de missatges actius d'accés a la radio multiplexat.

-FarolaC.AMSend->AMSenderC.AMSend: Component per enviar missatges actius del tipus SensorsDataMsg.

-FarolaC.AMSendConfirm->AMSenderCconf.AMSend: Component per enviar missatges actius del tipus ConfirmMsg.

-FarolaC.AMReceive->AMReceiverC.Receive: Component per rebre missatges actius del tipus SensorsDataMsg.

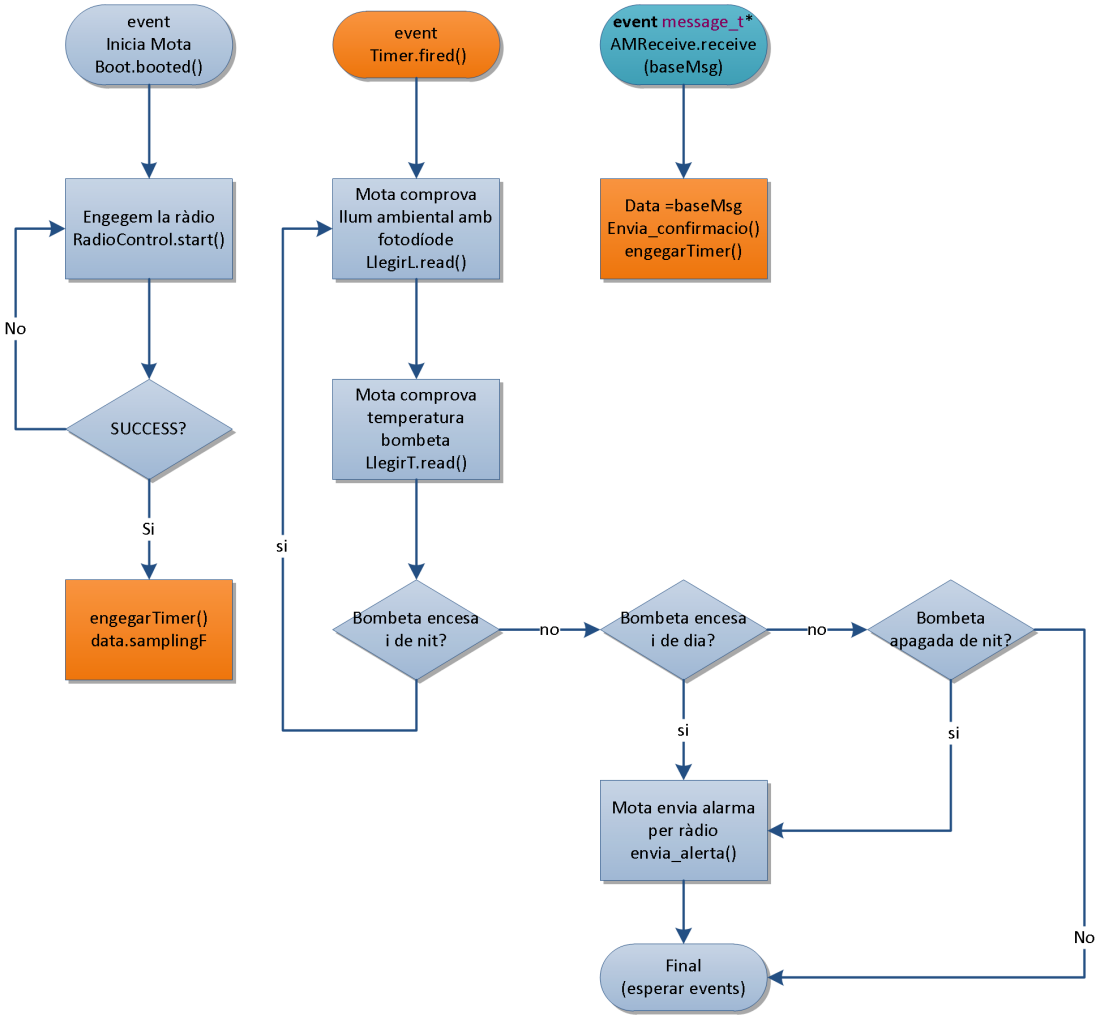
Com hem vist, existeixen **dos estructures de missatge** diferents: SensorsDataMsg i ConfirmMsg. Aquestes estructures estan declarades a l'arxiu de capçalera Farola.h que s'annexa juntament amb la resta de codi nesC. SensorDataMsg es la primera estructura amb la que vam començar a treballar. Vam pensar que amb una mateixa estructura o tipus de missatge seria suficient per satisfer les necessitats de missatgeria del projecte. Però no vam pensar en que, encara que a un missatge es pugui emmagatzemar la informació que vulguis, es necessari poder diferenciar uns tipus de missatges d'altres i es quan vam crear l'estructura de missatges ConfirmMsg específicament pels missatges que enviaven les motes al PC en actualitzar-se. Aquestes són les estructures fetes servir per la missatgeria:

```
typedef nx_struct SensorsDataMsg {    //estructura per intercanvi de missatges
        nx_uint16_t moteId;           //identificador de la mota que envia el missatge
        nx_uint16_t samplingF;        //frequencia de mostratge
        nx_uint16_t mesuraTemp;       //camp per guardar la temperatura
        nx_uint16_t mesuraLlum;       //camp per guardar la llum
        nx_uint16_t llindarTemp;      //camp per indicar el nou llindar de temperatura
        nx_uint16_t llindarLlum;      //camp per indicar el nou llindar de llum
        nx_uint8_t estat;             //estat de la farola: 0=OK, 1=Dia_encesa,
                                      //2=Nit_apagada
} SensorsDataMsg;

typedef nx_struct ConfirmMsg {//estructura per enviar confirmaciÃ³ d'actualitzaciÃ³
                              //de llindars a la base
        nx_uint16_t moteId;   //identificador de la mota
                              //que envia el missatge
} ConfirmMsg;
```

Mitjançant aquest Farola.h s'han creat les classes de Java SensorsDataMsg.java i ConfirmMsg.java que farem servir com a interfície d'accés als objectes de missatge des de l'aplicació FarolesGUI.java.

Pel que respecta a les tasques i rutines que s'executen a FarolaC, adjuntem un diagrama de flux:



Il·lustració 4.1.1.2:diagrama de flux FarolaC

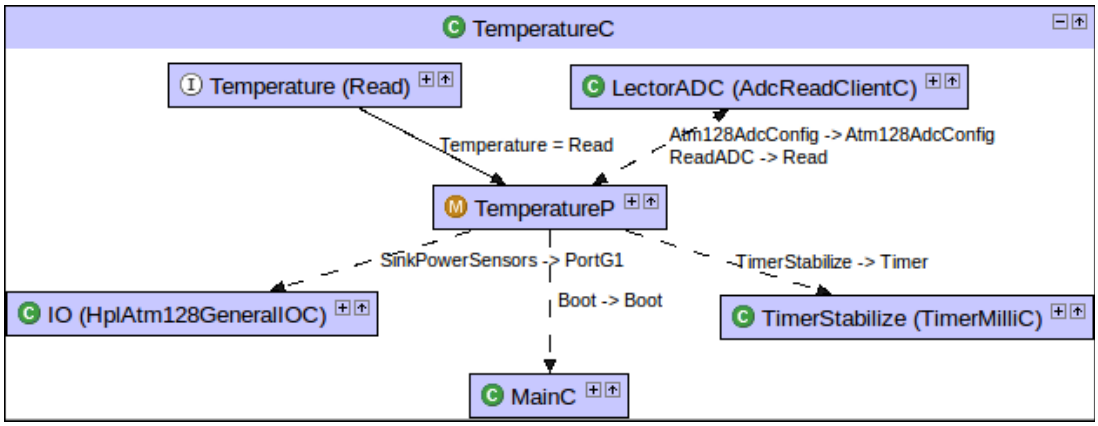
4.1.2. TemperatureP i LightP

En aquest apartat parlarem tant de TemperatureP com de LightP per igual perquè el codi de tots dos mòduls és exactament igual, l'únic que canvia és la configuració de la crida a l'ADC: pel de temperatura hem de retornar el canal 2 (`return ATM128_ADC_SNGL_ADC2;`) i per el sensor de llum hem de configurar el canal 1 (`return ATM128_ADC_SNGL_ADC1;`)

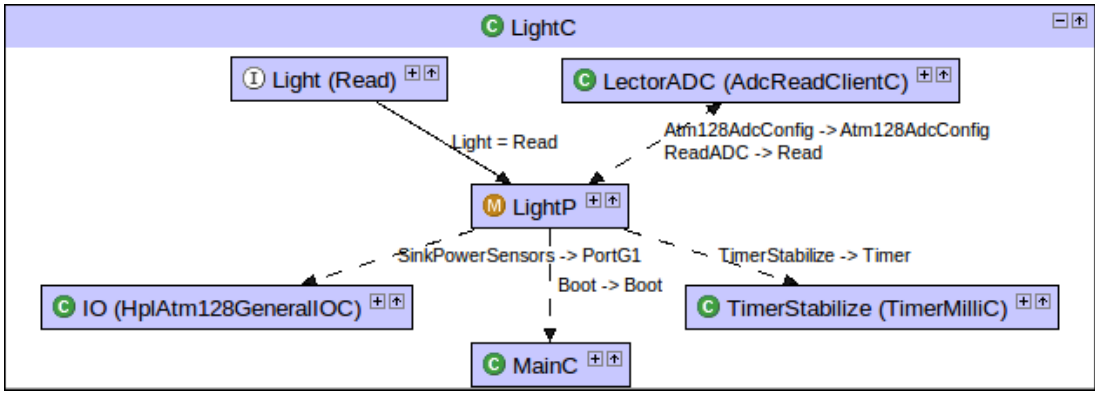
TinyOS en el moment en que el vam instal·lar al nostre ordinador de desenvolupament, per la mota COU_24_1_2 no proporcionava cap modul d'accés als sensors connectats a l'ADC. Per altres tipus de motes si que existien aquests components i des d'un primer moment vaig estar investigant sobre com crear els meus propis mòduls i que a més a més es poguessin fer servir per la resta de comunitat que fa servir la mateixa mota que jo.

> La idea d'aquests dos mòduls és la de proporcionar una capa d'abstracció per l'accés als sensors de temperatura i al de llum que es troben connectats al ADC del ATmega1281.

Mostrem el diagrama de connexions dels mòduls:



Il·lustració 4.1.2.1:Diagrama de connexions TemperatureC



Il·lustració 4.1.2.2:Diagrama de connexions LightC

I si ens fixem, aquest cop, a part de fletxes que surten, hi ha fletxes que entren al mòdul. Això vol dir que aquests mòduls a part de fer servir interfícies d'altres components, també proporcionen interfícies a altres components. Detallem les connexions que es fan servir (ho fem per LightC però el mateix s'aplica per TemperatureC):

-LightP.SinkPowerSensors->IO.PortG1: pin per encendre i apagar els sensors periferics.

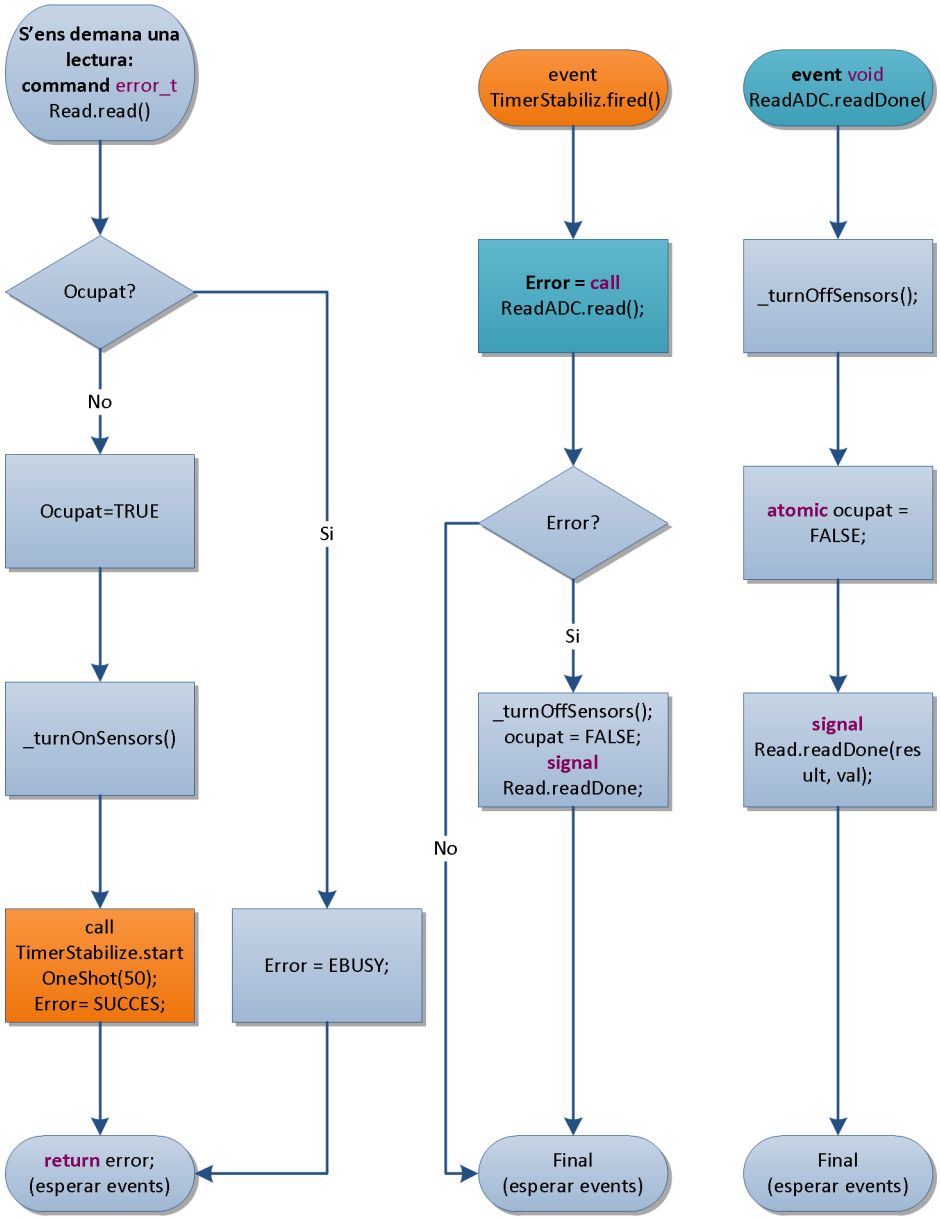
-LightP.Boot->BootC: És el component que ens indica que la mota ha arrencat i ens dóna pas per començar a executar codi.

-LightP.TimerStabilize->TimerStabilize: Temporitzador que farem servir per estabilitzar l'arranc dels sensors.

-LightP.Atm128AdcConfig<-LectorADC.Atm128AdcConfig: Proporcionem la configuració que volem fer servir per llegir del ADC al component AdcReadClientC.

-LightP.ReadADC->LectorADC.Read: Connectem a la interfície de Read del AdcReadClientC per poder demanar la lectura un cop configurat amb la connexió anterior.

Pel que respecta a les tasques i rutines que s'executen a TemperatureP i LightP, adjuntem els diagrames de flux:

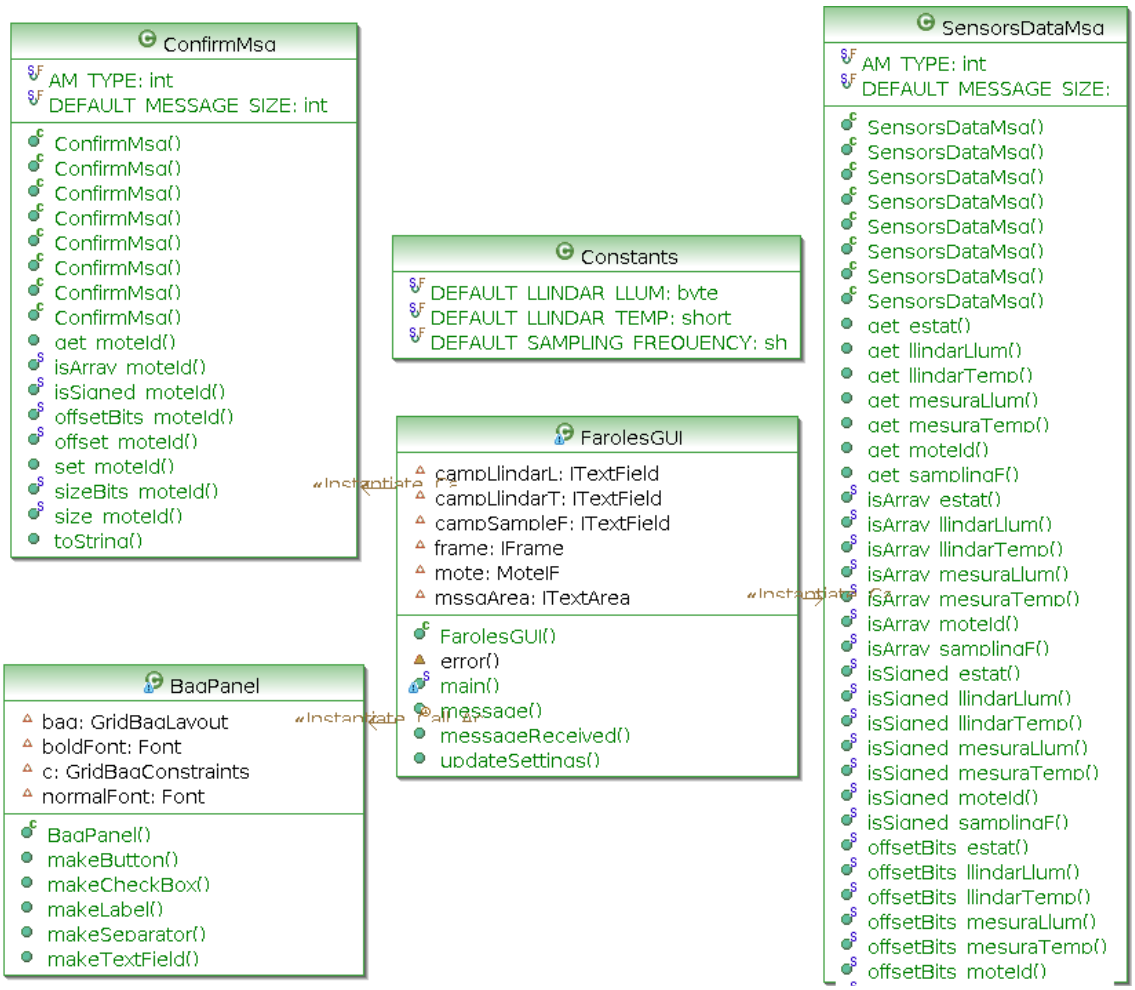


Il·lustració 4.1.2.3:Diagrama de flux de LightP i TemperatureP

4.2. TFCOrdinador (PC)

Es tracta de la carpeta que conté tot el codi necessari per executar l'aplicació de Java FarolesGUI que ens permetrà fer ús de la xarxa de sensors sense fils creada per les motes carregades amb el programari anteriorment descrit. La classe principal és FarolaGUI.java. No som experts en Java ni tampoc és el comés d'aquest projecte desenvolupar una aplicació de Java, així que bàsicament hem intentat realitzar una interfície senzilla però alhora funcional. Per poder fer servir aquesta aplicació és necessari per tant un entorn de Java correctament instal·lat i que tingui les classes de TinyOS, swing i awt (aquestes haurien de venir per defecte en qualsevol versió JRE recent) correctament configurades al $CLASSPATH de Java. Si fem

una ullada a la carpeta “src” que conté tot el codi, podrem veure els diferents elements que intervenen:



Il·lustració 4.2.1:Diagrama de classes de l'aplicació FarolesGUI

4.2.1. BagPanel.java

No em parlarem gaire d'aquesta classe donat que no ha estat creada per nosaltres i la seva aparició en aquesta memòria es merament per donar crèdit al seu autor i per informar del tipus d'ús que fem d'aquest recurs. Es tracta d'una classe creada per David Gay i amb Copyright de Intel Corporation de la qual farem servir els seus mètodes per crear el formulari (la zona de botons) de FarolesGUI.

4.2.2. ConfirmMsg.java i SensorsDataMsg.java

Es tracta de les classes de Java creades a partir de Farola.h fent servir l'eina 'mig'. Són les dues interfícies d'accés als objectes missatge que farem servir quan haguem d'enviar o rebre un missatge d'aquest dos tipus des de l'aplicació principal. Tenen tots els 'getters' i 'setters' dels diferents camps de l'estructura a part d'altres mètodes que nosaltres no hem fet servir. A la imatge 4.2.1 hem retallat SensorsDataMsg perquè en l'únic que es diferencia amb ConfirmMsg és en que té més camps, i per tant té més 'getters', 'setters' i resta de mètodes relacionats amb el nombre de entrades de la taula. La resta es igual.

4.2.3. FarolesGUI.java

És la classe principal de la nostra interfície gràfica, la que conté el mètode main(). La part de codi referent a les finestres i els gràfics està inspirada en l'aplicació Antitheft de les demostracions de TinyOS. Per tal de que un programa de Java pugui interactuar amb un entorn TinyOS és necessari carregar les classes TinyOS preparades per Java. Hi ha també aquestes classes per altres llenguatges com podríen ser python o C++ però ens van aconsellar fer servir Java perquè és el que fa servir la majoria de gent i existeix mes documentació i per tant més suport en cas de dubtes. Les llibreries que nosaltres hem fet servir han està net.TinyOS.meesage i net.TinyOS.util que contenen tot l'apartat de missatgeria de les motes i utilitats.

Resumint i sense intenció d'explicar tot el codi pas per (el codi ja està documentat línia per línia), la base es crear un objecte del tipus MoteIF (interfície mota) que representarà un nivell d'abstracció tal que el podrem tractar com si fos un canal de comunicacions amb les motes i enviar-hi o rebre missatges.

Una vegada es vol enviar un missatge, es crea un nou objecte de classe SensorDataMsg o ConfirmMsg, es configura amb les dades oportunes i es fa servir el mètode “send” de la classe MoteIF per enviar-lo. Per rebre missatges l'operació es molt similar a TinyOS, quan es rep un missatge, la senyal és capturada amb el mètode *messageReceived(int x, Message y)*. Aquest mètode rebrà el missatge *y* provinent de l'adreça *x*. I ens permetrà tractar aquestes dades de la manera que vulguem amb els 'getters' corresponents a cada camp. No es difícil, un cop s'entén la filosofia :).

De la resta de funcions no cal mencionar res especial, són coses de Java, aclarirem només les que tenen a veure amb les formules de conversió de valors de l'ADC a escala de graus centígrads. Per passar de la lectura del ADC que ens arriba de la mota graus centígrads, el primer que fem es passar a volts amb una tensió de referencia de 2,56V (2560) amb la següent formula i tenint en compte les característiques del ADC del ATmega1281:

$$V = \frac{counts}{1024} \cdot 2560/1000$$

Després de passar a volts i tenint en compte el calibratge que vam fer amb el termòmetre de jardí (veure annexes) i el coeficient de temperatura del MCP9700 (10mV/Cº), per passar a graus centígrads apliquem la següent formula tenint en compte que Vin per 0ºC és 522,5mV:

$$T = (V - 0{,}5225) \cdot 100$$

Aquestes formules les hem aplicat tant al enviar missatges amb nous llindars, com a l'hora de presentar missatges amb mesures per pantalla.

5. Viabilitat tècnica

Si bé creiem que amb el temps dedicat (7,5 crèdits) el projecte està en una fase força verda, també creiem que la viabilitat d'aquest projecte és bona. Com a punt feble creiem que tenim l'ús de bateries: les probes que he realitzat, donen un punt de negativitat: Un parell de bateries consultant els sensors cada 3 segons i enviant alertes de radio cada 3 segons aproximadament durant 10 hores al dia, han durat menys de tres dies. Això ens dóna com a resultat que les piles s'esgoten quan han realitzat 86400 mesures i han enviat uns 36000 missatges. Hem de tenir en compte que les nostres motes no envien missatges si no hi han alertes, i que la freqüència de monitoratge es pot ajustar, també s'haurien de fer més càlculs però penso que els resultats no son del tot bons.

També tenim com a punt negatiu el no haver pogut implantar el multi-hop per falta de temps. En un projecte com el nostre en el que volem donar cobertura a força faroles, actuar limitat pel radi de distancia al qual es troba la mota de la base es una limitació molt gran. Implementar el multi-hop no suposaria una complicació pel que també creiem que no ens atura el projecte.

La part bona és que la inversió es força petita i creiem que aconseguiria bons beneficis. Es cert que el projecte té coses a millorar, però també té punts forts com són la senzillesa de la instal·lació de les motes, la no necessitat de modificar les faroles en cap sentit, o la versatilitat que et dóna un sistema operatiu com TinyOS o una maquina virtual com la de Java. Treballem amb tecnologies emergents i d'altres que ja estan consolidades pel que el projecte no quedaria mort ni desfasat un cop realitzada la seva implantació en primera instancia. Els grans projectes no paren de créixer i el nostre projecte aniria ben acompanyat si TinyOS i Java segueixen un ritme evolutiu com el que han dut fins ara, també es important saber que els grans projectes mai es tanquen i és aconsellable continuar treballant en ell. Creiem que aquest projecte parteix d'una bona idea i d'una base tècnica que pot evolucionar a molt bons resultats. No es un projecte de moda en el que la gent pot canviar de gust, tampoc és una pàgina web on depens de la gent que es connecta, ni un projecte en el que depenguis de software de tercers que puguin tancar el negoci: és partir d'una idea bona en un projecte en el que hi ha molt poques variables, gairebé et garanteix l'èxit.

6. Valoració econòmica

Un projecte com el nostre, implantat per exemple a 20 faroles d'una urbanització, suposaria un valor econòmic estimat com el que veiem a continuació:

Producte	Preu €	Quantitat	Total €
Atmel ATZB-24-A2R	20,12	20	402,4
MCP9700-E/TO	0,32	20	6,4
pdv-p9003-1	1,58	20	31,6
Placa circuit	4,59	20	91,8
USB - UART Bridge,CP2102	2,83	20	56,6
Mà d'obra	10	160	1600
Asus EEEPC 1015PE	234	1	234
TOTAL			2422,8

On de forma orientativa s'ha tingut en compte que es crea una mota, es programa i s'instal·la en un dia laboral. També no s'ha calculat el preu dels components passius SMD ja que es tan baix que es pot depreciar.

Els preus es poden comprovar a les següents pàgines web:

-Atmel ATZB-24-A2R :

http://es.rs-online.com/web/search/searchBrowseAction.html?method=getProduct&R=6997313

-MCP9700-E/TO :

http://es.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=MCP9700&x=27&y=15

-pdv-p9003-1:

http://parts.digikey.com/1/parts/949084-photocell-23k-33k-ohm-4-20mm-pdv-p9003-1.html

-Placa circuit:

http://be.eurocircuits.com/Basic/Orders/PriceCalculator.aspx?Ln=en&HC=948&RC=843

-USB - UART Bridge,CP2102:

http://es.rs-online.com/web/search/searchBrowseAction.html?method=retrieveTfg&binCount=1&Nty=1&Ntx=mode%2bmatchallpartial&Ntk=I18NAll&Ne=4294958129&Nr=AND%28avl%3aes%2csearchDiscon_es%3aN%29&N=4294577151&Ntt=cp2102

-Asus EEEPC:

http://www.lifeinformatica.com

7. Conclusions

7.1. Conclusions

Un cop finalitzat tot el procés de desenvolupament, puc dir que estic molt satisfet amb la feina realitzada ja que ha estat molt didàctic i entretingut. També m'ha permès conèixer a fons tot el món dels sistemes encastats i les xarxes sense fils que fins ara, sincerament, pràcticament desconeixia. També m'ha sorprès la quantitat de sistemes i estàndards diferents que hi ha al mercat referent a les xarxes de sensors sense fils.

Parlant dels resultats, també m'ha sorprès fins on he arribat en tant poc temps. Podríem dir que he passat de gairebé no saber res d'un tema a gairebé dominar-lo en menys de 3 mesos.

El seguiment del pla de treball ha donat el seu fruit encara que alguna planificació que altre s'ha escapat de les mans.

Aquest treball també m'ha servit per demostrar la capacitat d'afrontar jo sol un projecte d'unes magnituds força importants. A la vida real aquests projectes no s'afronten tot sol, s'acostuma a crear un equip de treball on les tasques es reparteixen per donar paral·lelisme a les tasques.

La idea del projecte va sorgir després de donar-li moltes voltes a un concepte de xarxes que en aquell moment gairebé no coneixia, i tenia molts dubtes, però he de donar les gràcies al consultor perquè va saber interpretar les idees que tenia i em va aconsellar de manera molt esperançadora.

7.2. Proposta de millores

Si deixem córrer la imaginació, se m'acudeixen moltes de millores que es podrien aplicar al sistema, no és que el sistema sigui feble i sense contingut, és que en el món dels sistemes, les combinacions i adaptacions són pràcticament infinites. Intentaré plasmar una part significativa d'aquestes millores tot seleccionant les que penso que realment són necessàries donat el nivell de coneixement que tinc ara:

Part PC

-Arxivat de esdeveniments per poder realitzar estadístiques.

-Interfície d'usuari integrada amb google earth o amb algun altre sistema de mapes per saber quina farola està fallant de manera gràfica i localitzable.

-Sistema d'alertes per só. Si l'usuari no està mirant la pantalla una alerta sonora ajuda a cridar l'atenció.

-Alertes enviades per correu electrònic un cop rebudes.

-Integració de la finestra d'alertes amb twitter o web de missatges curts similar per poder mirar els events des de qualsevol lloc del planeta sense necessitar de muntar cap servidor web.

-Poder escollir el tipus de topologia de la xarxa i que aquesta s'actualitzi dinàmicament.

-Introduir perfils, de temporada de l'any per exemple, perquè els llindars s'actualitzin automàticament arribada la estació corresponent.

Part Mota

-Personalització del codi de la mota base. Hem detectat que en algunes ocasions falla.

-implementar multi-hop o altres topologies de xarxa per poder cobrir un gran territori de faroles.

-Crear un watchdog perquè la mota es reinicii automàticament en cas de errors.

-Sistema d'alertes de poca bateria.

-Un mode d'estalvi d'energia que canvii el temps de mostratge quan queda poca bateria i per donar temps a l'operari/a a canviar les piles sense que s'hagi d'estressar.

-Un sistema que quan es detecta molta llum amplia el temps de mostratge automàticament.

7.3. Bugs detectats

Després de fer les proves pertinents de qualitat del producte, no s'ha detectat cap bug significatiu que disminueixi la capacitat del projecte per acomplir els seus objectius. Únicament s'ha detectat una pèrdua de paquets de radio en determinades ocasions i de forma esporàdica que s'ha controlat amb la implementació del sistema de confirmació de missatges d'actualització que parlàvem en els apartats anteriors. Segons les proves i els assajos fets, hem comprovat que aquesta pèrdua de comunicació esporàdica és dóna també en els exemples proporcionats per la wiki de la mota cou24 (BlinkCou), i en les demos de TinyOS (Oscilloscope, BlinkToRado etc). Pel que podem concloure que no és problema del nostre codi sinó que d'algun dels altres factors implicats (medi hostil, hardware motes, mota Base, bugs en llibreries TinyOS, Ubuntu sobre vmWare etc...).

7.4. Autoavaluació

He tingut problemes amb el Java en mode gràfic. A ITTT només hem fet servir el C++ i no en mode gràfic, només consola. Un dels objectius proposats era crear una interfície d'usuari que fos vistosa i que no fos complicada d'utilitzar. Per aconseguir-ho era necessari aprendre quant abans millor l'us de les llibreries swing i awt de Java. Penso que al final me n'he sortit força bé i estic força sorprès del resultat final. Si fa 4 mesos em dius que faria una aplicació Java amb finestres i butonets et diria que no m'ho crec.

Els objectius estan gairebé tots complerts. M'ha quedat pendent un watchdog que mig he apanyat amb el sistema de “confirmació de missatges” que es pot fer servir com a 'ping'. També m'ha quedat pendent el tema del Quorum que no ha estat possible per falta de temps però que no és imprescindible. La resta han estat assolits: GUI, control d'errors, Hex ->Cº, etc. El temps que he dedicat no te'l sé calcular però t'asseguro que he fet moltes més hores de les que pensava que faria. Començar gairebé des de 0 amb 2 llenguatges de programació casi

nous no és fàcil. He passat nits molt llargues i caps de setmana molt durs. Imagina't com ha estat per dir que el dia de la final de la Champions, després del partit vaig haver de sortir corren cap a casa per posar-me a fer feina... Però ara estic content, l'esforç a servit per alguna cosa.

8. Glossari

COU_1_2 24: Dispositiu de proves construit específicament per els alumnes del TFC Sistemes encastats de la UOC.

TinyOS: sistema operatiu dissenyat exclusivament per sistemes encastats del tipus mota

mota: dispositiu independent membre de una xarxa de sensors que incorpora sensors i que disposa de certa capacitat de computació. Es caracteritzen per la seva mida reduïda i pel seu baix consum.

802.15.4: estàndard per les comunicacions d'àrea personal.

ZigBee: conjunt d'estàndards per les comunicacions d'àrea personal.

ZigBit: Encapsulat fabricat per ATMEL que disposa de microcontrolador i aparell de RF tot en un.

Meshprog: Programa per carregar les motes COU24 amb el codi compilat.

GUI: Interfície gràfica d'usuari.

JDT: Java Development Tools. Plugin per Eclipse IDE

CDT: C/C++ Development Tools. Plugin per a Eclipse IDE

Plugin: Mini-aplicació que es pot afegir a mode de nova característica a programes que accepten aquest tipus d'aplicacions.

Eclipse IDE: Plataforma de programació multi llenguatge caracteritzada per ser open source i amb gran varietat de afegits.

WSN: Xarxa de sensors sense fils.

ADC: Convertidor Analògic a Digital

llei de Moore: Llei que descriu la evolució temporal dels circuits integrats.

UART: Transmisor/Receptor Universal Asincron.

Java: Llenguatge de programació orientat a l'objecte.

Python: Llenguatge de programació multi-paradigma que posa èmfasi en una sintaxi fàcilment llegible

UML: Llenguatge de Modelat Unificat.

ATmega1281: Microcontrolador de la marca ATMEL basat en arquitectura RISC.

MCP9700: Transistor especialment dissenyat per mesurar temperatura (termistor). També es caracteritza pel seu baix consum.

CP2102: Circuit integrat que fa de pont entre un bus USB i un altre UART.

9. Bibliografia

1: Arp@:Embedded Systems Lab@Home http://cv.uoc.es/app/mediawiki14/wiki/P%C3%A0gina_principal

2: University of California, Berkeley 1998 http://www.tinyos.net/

3: http://edison.upc.edu/curs/llum/exterior/vias_p.html

4: http://es.wikipedia.org/wiki/Alumbrado_p%C3%BAblico

5: http://hypertextbook.com/facts/2000/MichaelLevin.shtml

6: http://www.injuryprevention.org/states/la/hotcars/hotcars.htm

7: http://books.google.es/books?id=aAAo9V2ACekC&pg=PA176&lpg=PA176&dq=variaci%C3%B3+de+la+temperatura+al+llarg+del+dia&source=bl&ots=zSUapKQxRi&sig=5Io9Lzvjse59ttVsqiee_dsyKeg&hl=es&ei=Zk-PTcnPB5CZhQfZk-i7Dg&sa=X&oi=book_result&ct=result&resnum=3&ved=0CDEQ6AEwAg#v=onepage&q=variaci%C3%B3%20de%20la%20temperatura%20al%20llarg%20del%20dia&f=false

8: http://www.electricidadlynch.com.ar/lamparasodio.htm

9: VMware Player http://downloads.vmware.com/d/info/desktop_downloads/vmware_player/3_0

10: The Ubuntu Project www.ubuntu.com

11: Eclipse Galileo IDE projects http://www.eclipse.org/galileo/

12: JDT - Java development tools http://www.eclipse.org/projects/project_summary.php?projectid=eclipse.jdt

13: C/C++ Development Tooling (CDT) http://www.eclipse.org/projects/project_summary.php?projectid=tools.cdt

14: Distributed Computing Group Yeti 2 - TinyOS 2 Plugin for Eclipse http://tos-ide.ethz.ch/wiki/index.php

15: TinyOS tree for COU_1_2 24 http://eimtcollab.uoc.edu/softcou24_12/tinyos-2.x_cou24.tar.gz

16: Die Research Studios Austria Forschungsgesellschaft mbH Meshnetics serial programmer for Linux http://pervasive.researchstudio.at/portal/projects/meshprog

17: Wikipedia Wireless Sensor Network http://it.wikipedia.org/wiki/Wireless_sensor_network

18: wikipedia Sensor Node - Motes http://en.wikipedia.org/wiki/Motes

19: UC Berkeley SMART DUST Autonomous sensing and communication in a cubic millimeter http://robotics.eecs.berkeley.edu/~pister/SmartDust/

20: Atmel ZigBit™ 2.4 GHz Wireless Modules- ATZB-24-A2/B0 http://store.atmel.com/Download.ashx/526c265e-ddcf-46da-85d7-6a48c109a0f3

21: Cossbow Energy Monitoring Solution - EcoWizard http://www.xbow.com/pdf/EcoWizard_Introduction_PressRelease.pdf

22: Francesco Corucci, Giuseppe Anastasi, Francesco Marcelloni A WSN-based Testbed for Energy Efficiency in Buildings http://info.iet.unipi.it/~anastasi/papers/iscc11-GreenBuilding.pdf

10. Annexos

10.1. Guia de Instal·lació i funcionament del sistema

La compilació dels programes es senzilla, totes les llibreries que utilitza les tenim amb l'entorn TinyOS 2.x preparat per les cou24, i les llibreries estàndard que no servien, s'han creat de nou a les carpetes de codi de les nostres motes (per exemple TemperatureC i LightC).

Un cop instal·lat l'entorn de TinyOS 2.x preparat per les cou24, es suficient amb entrar a la carpeta destinada a cada mota, i dins de la carpeta /src/ executar la següent ordre:

make cou24

Aquesta ordre generarà el binari que s'ha de carregar a la mota. Aquest binari el trobarem a la carpeta /build/cou24. Per passar-lo a la mota s'ha d'executar l'ordre següent des de la carpeta cou24:

meshprog -t/dev/ttyUSB0 -f./main.srec

on *main.srec* fa referencia al programa de sortida obtingut a la compilació de l'apartat anterior.

Si volem instal·lar més d'una mota Farola, haurem de tornar a executar l'ordre *make* però aquesta vegada indicant que volem un mote_id diferent (si no s'indica rés el mote_id per defecte és 1), per exemple el 4:

make cou24 reinstall,4

L'ordre *make* donarà error en intentar instal·lar el codi a la mota però a nosaltres l'únic que ens interessa és que s'hagi creat l'arxiu *build/cou24/main.srec.out-4* ja que nosaltres farem servir un altre cop l'instal·lador *meshprog*, tal que així:

meshprog -t/dev/ttyUSB0 -f./main.srec.out.4

Un cop tenim les dues motes z_TFCFarola i z_TFCBase carregades, hem de connectar la mota Base al port USB i haurem de llençar l'eina *SerialForwarder:*

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

Aquest *SerialForwarder* permetrà establir connexió a la nostra aplicació Java *FarolesGUI* a través del port TCP9002.

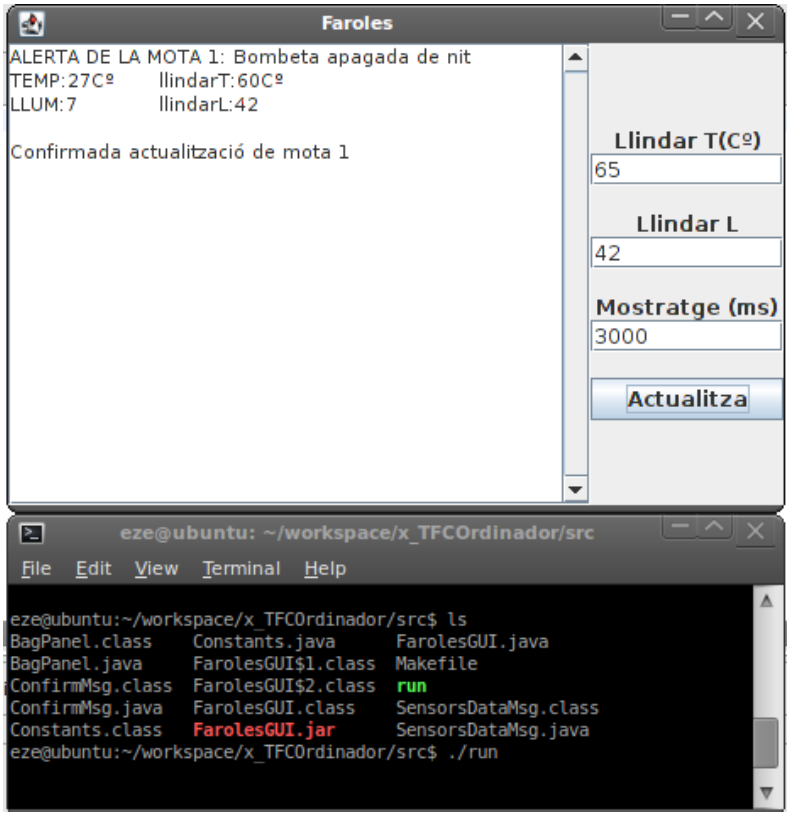
El programa ja està compilat i no és necessari tornar-ho a fer. Si s'hagués de tornar a compilar, s'hauria de seguir el procediment estàndard per Java tenint en compte que la

classe principal és *FarolesGUI.java.*

Per llençar el programa, anirem a la carpeta */x_TFCOrdinador/src* i un cop assegurats de que l'arxiu *run* té permisos d'execució linux, llencem la següent ordre:

./run

I ja tindrem la interfície funcionant i connectada al port 9002 del *SerialForwarder:*



El funcionament es senzill: A l'esquerra tenim el panel d'informació i a la dreta el d'interacció amb les motes. A l'esquerra surten les alertes d'estat erroni de les faroles i des de el panel de la dreta es manipulen i s'envien en Broadcast els llindars i la freqüència de mostratge.

10.2. Propostes inicials

PROPOSTA	DUBTES TECNICS
monitor enllumentat públic (fotodiode comprova necesitat de funcionament, temp comprova funcionament làmpara)	Xarxa? Suficient Distancia 802.15? Llindars Temperatures
sistema d'informació de vehicles (Hall velocitat, fotodiode necesitat d'enllumenat, temp comfort habitacle)	suposo que no es viable perquè es necesita un PC per interpretar les dades...
Monitoratge de cotxes d'slot o RC (hall: crono temps intermitjos, medidor tensió: telemetría motors electrics)	Funcionament Sensor Camp Magnetic, sensor medidor de tensió.
Monitoratge comfort de convois de Metro (llum, temp)	Distancia 802.15?
Monitoratge de sales CPD's (Hall para controlar blowers amb molí, temp)	Es pot fer servir mes estris dels proporcionats? Per exemple un molí de vent per check d'aire

Preseleccionada

Descartada

10.3. Proves realitzades amb els sensors

Per mirar d'establir uns llindars per defecte més o menys acceptables i per fer-nos una idea de les mesures que estabem realitzant es van fer les següents messures:

```
/**
 *
 * Mesures fetes amb el sensor de temperatura (MCP9700) i un termòmetre:
 * ADC=0x121 --> T=20Cº
 * ADC=0x115 --> T=18Cº
 * ADC=0x10A --> T=16Cº
 * ....
 * I tenint en compte la formula del ADCd = Vin*1024/Vref amb
 * una lectura de 20Cº tenim un Vin de 722.5mV i si el nostre sensor té un
 * coeficient de temperatura de 10mV/Cº, un llindar de 65Cº seria aproximadament
 * una lectura de (45Cº*10mV/Cº+722.5mV)*1024/2560= 469d = 0x1D5
 *
```

```
 * Mesures fetes amb el sensor de llum (PDV-P9003-1):
 * Observem que el 24/04 a les 21:01 tenim una llum solar insuficient, i el ADC marca 0x2A
 *
 */
```

10.4. Instal·lació entorn TinyOS

El TinyOS base utiliza motas predefinidas. Nosotros no tenemos una mota estandard (cou24) y es necesario parchear la instalación inicial de TinyOS.

1. Sigue esto http://docs.tinyos.net/index.php/Installing_TinyOS_2.1.1#Two-step_install_on_your_host_OS_with_Debian_packages para instalar TinyOS 2.1.1 en debian based OS. Solo la parte que dice "two-Step..." (o sea Ubuntu)

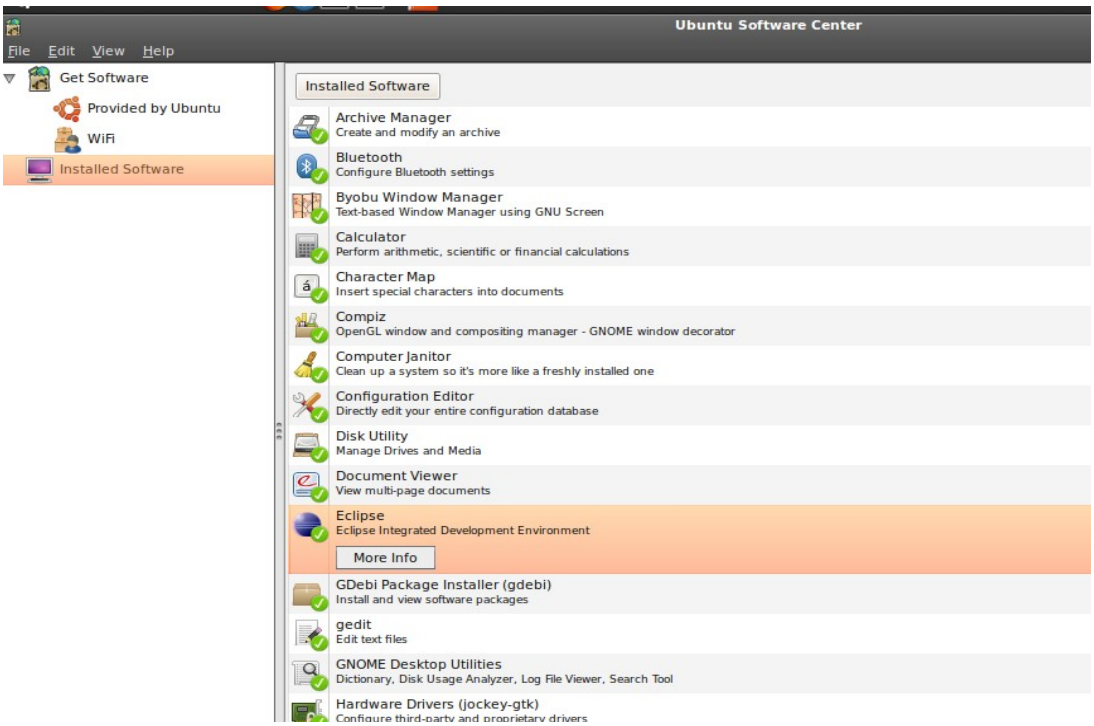
2. Una vez instalado, veras que se ha copiado en /opt/tinyos-2.1.1/. Reemplaza el contenido de la carpeta TinyOS 2.1.1 por el contenido de esto: http://eimtcollab.uoc.edu/softcou24_12/tinyos-2.x_cou24.tar.gz que es el programa tuneado para nuestra mota. Veras que dentro del zip hay una carpeta llamada tinyOS 2.x, pues lo que hay dentro de eso tiene que ir dentro de /opt/tinyos-2.1.1/

3. Descarga e instala http://eimtcollab.uoc.edu/softcou24_12/meshprog-0.1.2.tar.gz que es el programa que utilizamos nosotros que tenemos la mota cou24 para enviar los programas al dispositivo. Los que utilizan dispositivos estandard y por lo tanto en la mayoria de tutoriales de la pagina de TinyOS veras que utilizan el make con baudios para enviar a la mota, olvidate de eso, nosotros enviamos con meshprog con una orden parecida a esta "meshprog -t/dev/ttyUSB0 -f./main.srec"

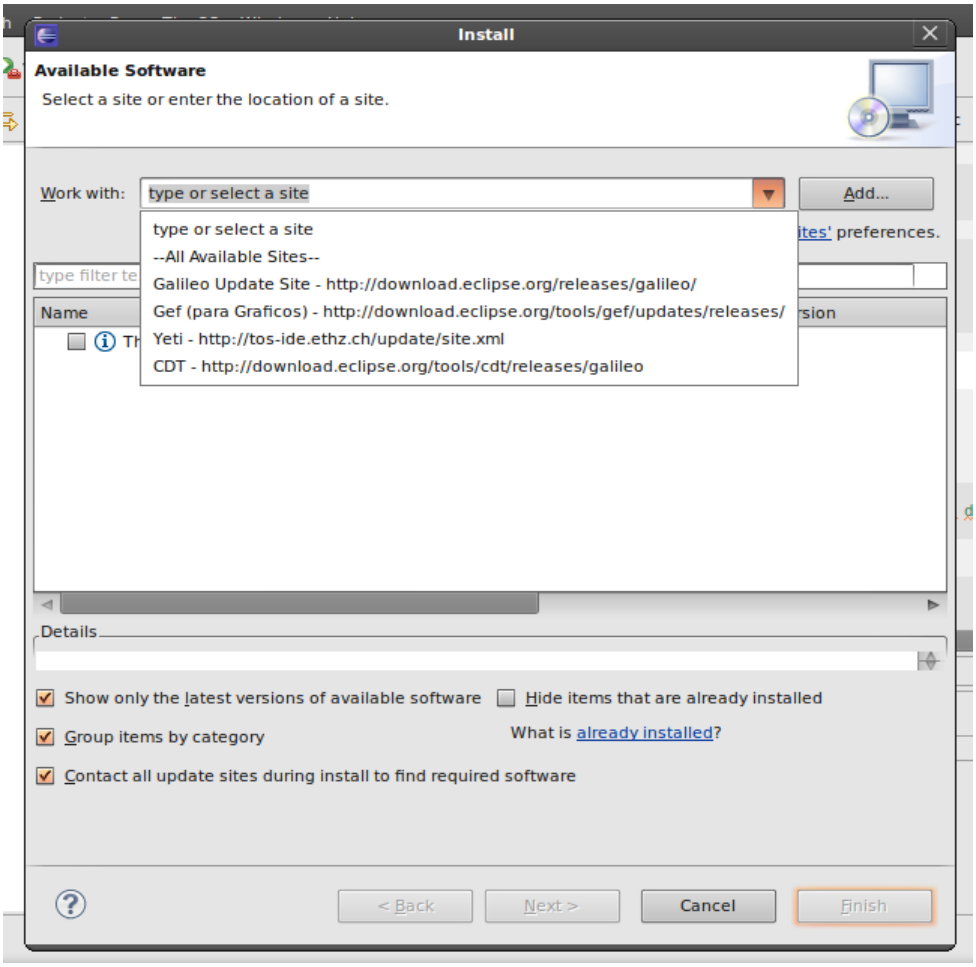
4. Sigue esto http://cv.uoc.es/app/mediawiki14/wiki/Inici24#Learning_TinyOS para empezar a programar.

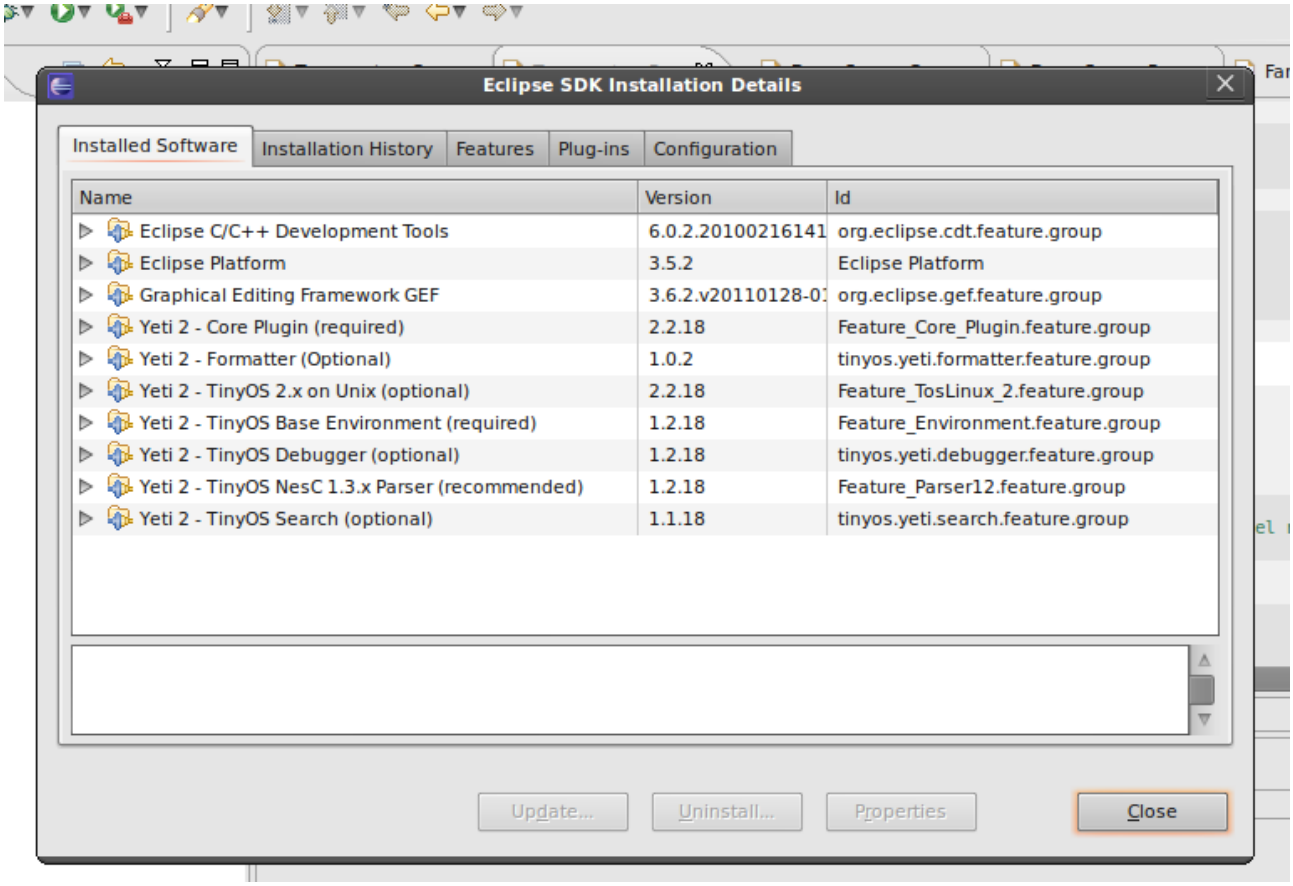
10.5. Instal·lació de Eclipse IDE amb Yeti 2

Primer instal·lem Eclipse desde el manager de Ubuntu:



I despres el modul.




Eclipse SDK Installation Details
Installed Software
Installation History
Features
Plug-ins
Configuration
Name
Version
Id
Eclipse C/C++ Development Tools
6.0.2.20100216141
org.eclipse.cdt.feature.group
Eclipse Platform
3.5.2
Eclipse Platform
Graphical Editing Framework GEF
3.6.2.v20110128-01
org.eclipse.gef.feature.group
Yeti 2 - Core Plugin (required)
2.2.18
Feature_Core_Plugin.feature.group
Yeti 2 - Formatter (Optional)
1.0.2
tinyos.yeti.formatter.feature.group
Yeti 2 - TinyOS 2.x on Unix (optional)
2.2.18
Feature_TosLinux_2.feature.group
Yeti 2 - TinyOS Base Environment (required)
1.2.18
Feature_Environment.feature.group
Yeti 2 - TinyOS Debugger (optional)
1.2.18
tinyos.yeti.debugger.feature.group
Yeti 2 - TinyOS NesC 1.3.x Parser (recommended)
1.2.18
Feature_Parser12.feature.group
Yeti 2 - TinyOS Search (optional)
1.1.18
tinyos.yeti.search.feature.group
Update...
Uninstall...
Properties
Close