Elaboració d'un protocol basat en XML per una xarxa de sensors

Xavier Moldes Ascón
Enginyeria tècnica en informàtica de sistemes - ETIS

Consultor: Jordi Bécares
17 de Gener de 2.011

AGRAÏMENTS

Al consultor, Jordi Bécares, per la seva ajuda i guia durant l'elaboració d'aquest projecte, i a tots els altres consultors que m'han ajudat i ensenyat al llarg de la carrera

Als meus pares, que em van ensenyar a ser persistent entre d'altres moltes coses

I, sobre tot, a la meva família, l'Arnau, el Gerard i la Marta que són els legítims propietaris de bona part del temps que he dedicat a aquesta carrera i sense el qual no me n'hagués sortit

Elaboració d'un protocol basat en XML per una xarxa de sensors

Paraules clau: Xarxa de sensors, sistemes encastats, XML

RESUM

La progressiva reducció de dimensió i cost en els dispositius electrònics, la seva dràstica retallada de consum elèctric i la independència de que això els dota han fet créixer en els últims temps l'interès de les comunitats científiques i tecnològiques per les xarxes sense fils de petits dispositius. Per altra banda, l'XML (*eXtensible Markup Language*) és un metallenguatge extensible que ha esdevingut un standard per l'intercanvi d'informació estructurada entre diferents plataformes.

L'objectiu d'aquest treball és explorar les possibilitats que pot oferir l'introducció del XML en les xarxes de sensors amb l'elaboració d'un protocol de comunicació basat en aquest llenguatge i demostrar la transparència en el canvi de plataforma. Per fer-ho, es disposa de dos dispositius amb capacitat de comunicació sense fils equipats amb detectors de temperatura, lluminositat, efecte Hall i nivell de càrrega de la bateria. El projecte constarà de dues parts: una, més extensa, dedicada al desenvolupament del programari per aquests dispositius, encarregat de obtenir les lectures dels diferents sensors i emetre-les per la xarxa utilitzant el llenguatge XML, i una altra, per recollir aquesta informació present a la xarxa, interpretar-la, salvar-la en una base de dades i exposar-la al món en una plana web.

El programari dels dispositius sensors s'escriurà en llenguatge nesC dins el sistema tinyOS que és el sistema operatiu que equipen. La part d'explotació de les dades es desenvoluparà sota la plataforma .NET de Microsoft.

ÍNDEX

Capítol 1. Introducció 7
1.1 Xarxes de sensors 7
1.1.1 definició 7
1.1.2 organització 8
1.1.3 Protocols de MAC (control d'accés al medi) 8
1.1.4 Models d'encaminament 8
Encaminament en estrella 8
Encaminament en arbre o multi-salt 9
Encaminament basat en clústers 9
1.2 Motes 10
Microcontrolador 10
Transceptor 10
Memòria externa 10
Font d'alimentació 10
Sensors 10
1.3 Tinyos i nesc 11
1.3.1 Components 11
1.3.2 Commands, Events i Tasks 11
1.3.3 nesC 12
1.3.4 Serial forwarder 12
1.4 XML 13

Capítol 2. Descripció funcional del projecte 15
2.1 Definició 15
2.2 Objectius 16
2.3 Node sensor o Mota 17
2.3.1 Mota base 18
2.3.2 Mota sensor 18
2.4 Protocol de comunicació 18
2.5 Sistema d'explotació 20
2.5.1 Serial forwarder 20
2.5.2 Servidor d'objectes remots 20
2.5.3 Servidor web 20

Capítol 3. Desenvolupament 21
3.1 Recursos 21
3.2 Programari de les motes 21
3.2.1 Codi comú 21
Header tfcNet.h 21
Component *XMLparser* 21
3.2.2 Projecte tfcStation 23
Component *tfcStation* 23
Component SensorsCache 23
Component StationRadio 24
Components BatterySender, TemperatureSender, PhotoSender 25
Component *HallSender* 26
3.2.3 Projecte tfcHost 26
Component *tfcHost* 26
Component HostRadio 26
Component *HostSerial* 27
3.3 Programari de la banda PC 28
3.3.1 Serial Forwarder i remote object server (TFC Sockets) 28
3.3.2 Projecte web 30
3.4 Compilació i Instal·lació 31

3.4.1 Compilació i Instal·lació a les motes 31
tfcStation 32
tfcHost 35
3.4.2 Compilació i instal·lació al PC 36
Compilació i instal·lació del servidor d'objectes 37
Compilació i instal·lació de la web 38
3.5 Planificació 40
3.6 Desenvolupament futur 41

ÍNDEX D'IL·LUSTRACIONS

1. Encaminament en estrella 8
3. Encaminament basat en clústers 9
2. Encaminament en arbre o multi-salt 9
4. Arquitectura típica d'una mota 10
5. Exemple d'aplicació tinyOS 11
6. Visió general del projecte 15
7. Imatge de la mota i detall de la ubicació dels seus elements 17
8. Detall de la CPU 17
9. Memòria present a la mota 18
10. Esquema del Handshake 19
11. Esquema tfcStation 23
12. Diagrama de fluxe de *SensorsCache* 24
13. Diagrama de fluxe de *StationRadio* 25
14. Diagrama de fluxe de *BatterySender*, *TemperatureSender* i *PhotoSender* 25
15. Esquema de tfcHost 26
16. Diagrama de fluxe de *HostRadio* 27
17. Diagrama de flux de HostSerial 27
18. Aspecte del formulari de control del cTFCmanager 29
19. Aspecte de la plana WEB 30
20. Detall d'espera de confirmació d'acció 30
21. Detall carpeta Projects 31
22. Detall carpeta *tfcCommon* 32
23. Detall carpeta *tfcStation* 32
24. Detall compilació *tfcStation* 33
25. Detall programació de la mota 34
26. Detall carpeta *tfcHost* 35
27. Detall compilació tfcHost 35
28. Detall programació mota base 36
29. Vista carpeta projectes 36
30. Detall compilació tfcSocket 37
31. Detall resultat compilació tfcSoket 37
32. Detall compilació tfcWeb 38
33. Detall publicació web 38
34. Opcions per la publicació de la web 39

Capítol 1. Introducció

A l'última dècada del segle XX les xarxes van revolucionar la manera en què persones i organitzacions intercanviaven informació i coordinaven les seves activitats. En els propers anys una nova generació de xarxes tornarà a revolucionar el món tecnològic; els darrers avenços han fet realitat el desenvolupament de dispositius petits, econòmics i de baix consum capaços de captar informació i processar-la i també de comunicar-se sense necessitat de fils. La capacitat de comunicar-se sense fils no és nova, però si que ho és que ho facin dispositius tan petits i que, gràcies al seus baixos cost i consum, puguin constituir grans xarxes autònomes de captació d'informació amb finalitats molt diverses.

Les arquitectures tradicionals de xarxa basades en protocols de comunicació amb moltes capes no són aplicables en aquestes; els dispositius no disposen de suficient memòria per implementar-los i, per altra banda, el consum energètic necessari per suportar-los els faria inviables. Cal desenvolupar nous models físics i organitzatius basats en la dispersió, l'aleatorietat i el mínim consum.

Per altra banda, un llenguatge de marques anomenat XML (*eXtensible Markup Language*) ha esdevingut en els últims anys l'standard en l'intercanvi d'informació estructurada entre plataformes diferents, i últimament la majoria dels sistemes B2B el fan servir per intercanviar dades.

En aquest primer capítol es defineixen les característiques de les xarxes de sensors i les diferents modalitats d'organització existents, els equips electrònics que les formen, el sistema operatiu que les governa i la seva metodologia de programació; a continuació es descriu el metallenguatge XML i les seves principals característiques. El segon i tercer capítol estan dedicats a la presentació del projecte i els seus objectius i a la descripció del programari desenvolupat. Finalment es troben les conclusions, la bibliografia i els annexes.

1.1 Xarxes de sensors

1.1.1 definició

Una xarxa de sensors «*Wireless Sensor Network, WSN*» pot ser descrita com un grup de dispositius capaços d'interactuar amb el seu entorn mesurant o controlant paràmetres físics, que es comuniquen i es coordinen per a dur a terme una aplicació específica vinculada a un espai o medi físic.

Les xarxes de sensors son un repte per a la investigació i l'enginyeria gràcies a la gran flexibilitat que presenten. No hi ha un únic conjunt de regles que classifiqui clarament totes les xarxes de sensor i no hi ha tampoc una única solució tècnica que satisfagui totes les necessitats.

1.1.2 organització

Inicialment aquestes xarxes estaven composades per un nombre relativament petit de sensors connectats per cable a una unitat central on es processaven les dades. Avui, en canvi, les xarxes estan formades per nombres més elevats de sensors que incorporen petites fonts d'energia i que abasten àrees molt més àmplies gràcies a la independència que atorga el fet de no necessitar cables per comunicar-se. De fet, en algunes topologies no és necessari que la unitat central comuniqui directament amb tots els nodes de la xarxa, sinó que són els mateixos nodes els responsables de propagar la informació més enllà de l'abast de la base, augmentant així la cobertura de la xarxa fins a cobrir àrees molt extenses.

1.1.3 Protocols de MAC (control d'accés al medi)

Com a tota xarxa, les xarxes de sensors implementen les capes física, la capa d'accés al medi i la capa d'aplicació. Dins de la capa d'accés al medi trobem diferents protocols, molts d'ells desenvolupats a partir dels altres, i molt orientats a l'estalvi d'energia.

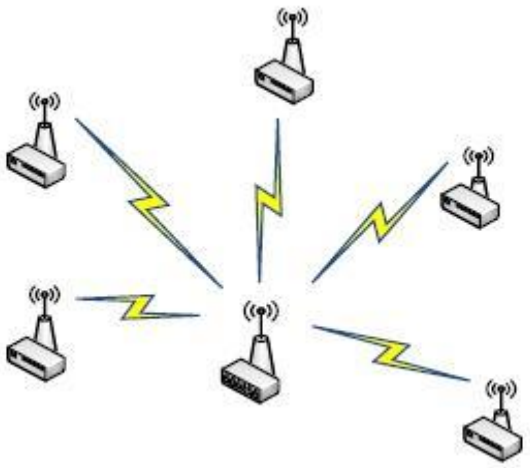
1.1.4 Models d'encaminament

En general, tots els models de xarxa preveuen l'existència d'un element, anomenat base, que fa d'enllaç entre la xarxa i els sistemes que l'han d'explotar fent-hi arribar tota la informació que circula. Els nodes no tenen un coneixement previ de la topologia de la xarxa, l'han de descobrir. Quan un nou element s'hi afegeix, s'identifica amb un missatge *broadcast*[1] i es queda escoltant la xarxa, llegint els missatges, fins que té una idea clara de cap a quin node ha de dirigir la seva comunicació.

Els models d'encaminament determinen com la informació circula des dels sensors emissors cap a la unitat central. Principalment podem distingir dos:

Encaminament en estrella

L'encaminament en estrella és el model més simple, aquell en el que tots els nodes de la xarxa comuniquen directament amb el node base.
Limita la grandària de la xarxa a l'abast de transmissió dels nodes (Si tots han de poder comunicar amb la base, aquesta s'ha de trobar dins de l'abast de tots els nodes); és només apta per petites i poc poblades xarxes.

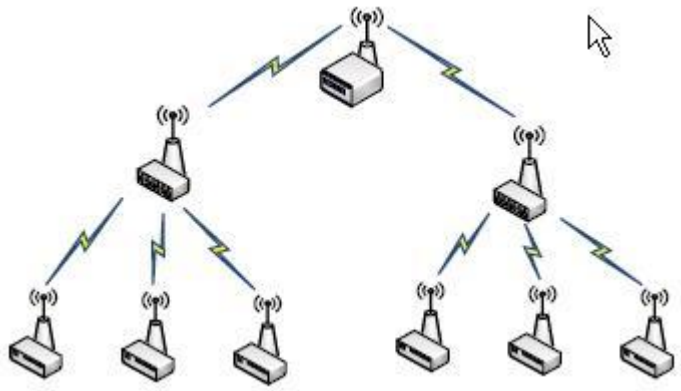


1. Encaminament en estrella

[1] Un missatge *broadcast* és aquell que no va dirigit a un element concret, sinó a tots aquells que el rebin.

Encaminament en arbre o multi-salt

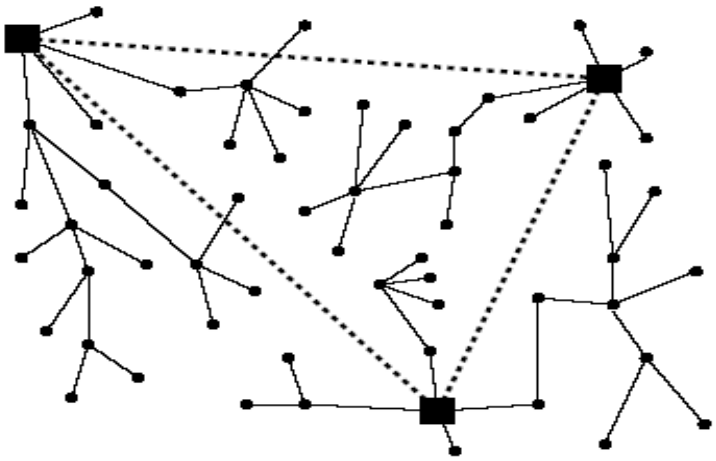
En aquest model els nodes s'organitzen en forma de capes al voltant del node base. Els missatges viatgen de capa en capa des de les posicions més llunyanes fins a la mota base. Els nodes, en ser inserits a la xarxa, han de triar entre tots els seus veïns, un dels que es trobin en la capa més inferior. La grandària de la xarxa, en aquest model, és il·limitada, sempre i quan les distàncies entre els nodes respectin l'abast individual d'aquests.



2. Encaminament en arbre o multi-salt

Encaminament basat en clústers

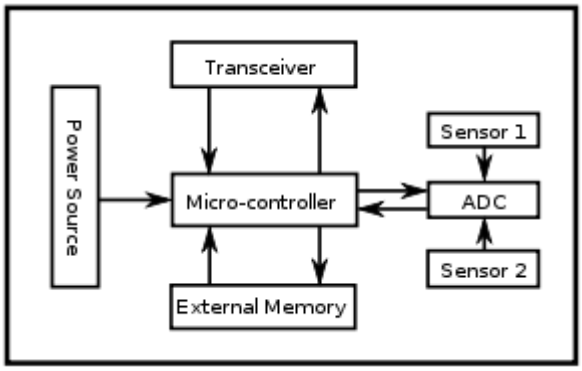
Hi ha models d'encaminament més complexos basats en l'anterior, en els que la xarxa s'organizta en grups de sensors, o *clusters*, cada un dels quals té un node base; tots els nodes base dels diferents *clusters* conformen una xarxa de nivell superior.



3. Encaminament basat en clústers

1.2 Motes

Una mota és un node en una xarxa de sensors sense fils amb una mínima capacitat de processament, equipada amb un o més sensors dels que capta informació i capaç de comunicar amb altres elements.



4. Arquitectura típica d'una mota

Els components principals d'una mota són el microcontrolador, el transceptor, la memòria externa, la font d'alimentació i un o més sensors.

Microcontrolador

El microcontrolador és l'encarregat d'executar els processos, processar les dades i controlar el funcionament dels altres components del node.

Transceptor

El transceptor és el component que habilitat la comunicació amb els altres nodes de la xarxa. Normalment aquesta es realitza dins la banda de ràdio ISM (*Industrial, Scientific and Medical radio band*) que pot ser utilitzada sense llicència. Altres possibles vies de comunicació són la radio freqüència, comunicació òptica (làser) i infrarojos.

Memòria externa

La memòria pròpia del microcontrolador (integrada al propi xip) i la memòria Flash són els tipus de memòria mes freqüents en aquests dispositius.

Font d'alimentació

Captar i processar informació i comunicar-se són tasques que consumeixen energia. LA comunicació és, amb diferència, el que més energia consumeix. L'energia es emmagatzemada en bateries, que són la principal font dels nodes sensors. Alguns sensors poden també utilitzar energia solar per abastir-se.

Sensors

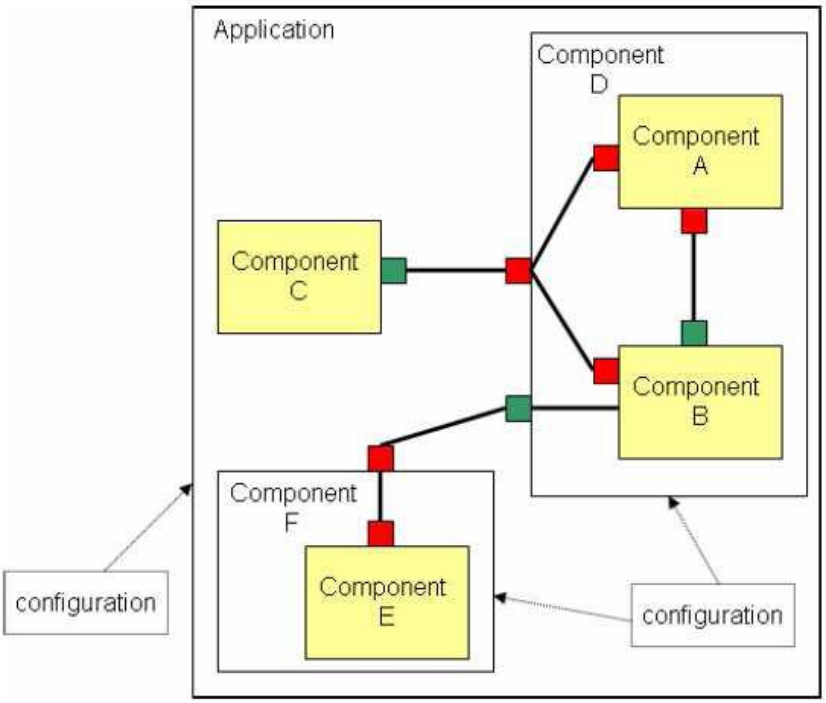
El sensors són dispositius que produeixen una resposta mesurable davant d'un canvi en una condició física, com la temperatura o la pressió. Els sensors mesuren dades físiques el paràmetre objectiu. El senyal analògic continu obtingut dels sensors és digitalitzat per un convertidor analògic-digital i enviat al microcontrolador per el seu tractament.

1.3 Tinyos i nesc

tinyOS «*Tiny microthreading Operative System*» es un sistema operatiu de codi obert basat en components per xarxes de sensors sense fils. Està escrit en el llenguatge de programació nesC com un conjunt de tasques i processos que col·laboren entre ells. Està preparat per funcionar sota les importants restriccions de memòria que presenten els nodes de les xarxes de sensors. TinyOS està desenvolupat per un consorci liderat per La universitat de California a Berkeley, en cooperació amb Intel Research.

1.3.1 Components

El tinyOS és un sistema operatiu orientat a components. En aquest tipus d'arquitectura les aplicacions es creen enllaçant components afavorint la reutilització de codi mitjançant les seves interfícies i l'abstracció del hardware amb el que es treballa.



5. Exemple d'aplicació tinyOS

Les aplicacions van sempre unides de forma implícita al sistema operatiu, de manera que no existeix diferència entre sistema operatiu i aplicació, com és freqüent a d'altres sistemes operatius, sinó que aplicació i sistema operatiu formen un únic bloc que es carrega al dispositiu.

1.3.2 Commands, Events i Tasks

La dinàmica dels programes realitzats sota tinyOS està conduïda per *tasques*, que poden ser interrompudes per *events* que gaudeixen de major prioritat per ser atesos. El sistema manté una cua amb les tasques pendents d'executar i entra en mode *standby* per estalviar bateria.

Una de les tècniques de programació més estesa és l'ús d'operacions *split-phase,* que consisteixen en substituir una crida directa per realitzar una operació per una crida que inicia la operació però torna immediatament i un event que notifica la finalització.

El tinyOS està basat, doncs, en tres classes d'abstracció la comprensió de les quals és la clau per programar correctament:

- Els ***commands*** són crides cap avall, d'un component cap a d'altres de capes inferiors
- Els ***events*** són la inversa als commands, una crida cap amunt que el component de capes inferiors utilitza per notificar als components de capes superiors que ha succeït alguna cosa.
- Les ***tasks*** són seccions de codi que s'executen de forma asíncrona, sempre i quan la CPU no hagi de tractar cap event. S'executen per ordre de crida (cua FIFO).

1.3.3 nesC

El llenguatge de programació propi del tinyOS és el nesC, una extensió del llenguatge C utilitzada per la programació d'aplicacions basades en components. El nesC permet el desenvolupament amb molt poc espai de memòria, un dels requeriments més importants dels sistemes encastats.

El nesC contempla tres tipus d'elements, els mòduls, les configuracions i les interfícies. Els component implementen i consumeixen interfícies, que defineixen quin *commands* i *events* accepta i emet un component. Les configuracions defineixen els enllaços entre els components o *wiring* i els mòduls contenen el codi del programa.

El nesC incorpora un joc de tipus de dades independents de la plataforma que facilita la programació per a microcontroladors de diferents estructures.

1.3.4 Serial forwarder

tinyOS, a més de constituir el sistema operatiu que s'instal·larà en les motes i el compilador per generar el paquet de l'aplicatiu, proveeix també una sèrie d'eines orientades a facilitar les tasques de depuració i proves dels sistemes.

Una d'aquestes eines, escrita en java, és el Serial forwarder. Pensada per actuar com a pont entre la xarxa de sensors i el món exterior a aquesta, el Serial Forwarder rep tot allò que la mota base transmet per el port sèrie i ho ofereix a tot aquell que es connecti amb ella mitjançant un sòcket.

1.4 XML

XML «*eXtensible Markup Language*» és un metallenguatge extensible de marques desenvolupat pel W3C «*World Wide Web Consortium*». És una simplificació i adaptació del SGML «*Standard Generalized Markup Language*» basat alhora en el GML «*Generalized Markup Language*» creat per IBM als anys 70.

La tecnologia XML pretén donar solució al problema d'expressar informació estructurada de la manera més abstracta i reutilitzable possible. Que la informació sigui estructurada vol dir que es composa de parts ben definides, i que aquestes parts es composen alhora d'altres parts. Això vol dir que es té un arbre de peces d'informació. Cada una de les parts es diuen elements i són identificats mitjançant etiquetes. Una etiqueta és, doncs, una marca feta al document i que identifica una secció d'aquest com un element; una peça d'informació amb sentit clar i definit.

Els objectius que el W3C va definir per l'XML són:

- XML s'ha de poder fer servir directament des de internet
- XML ha d'admetre una gran varietat d'aplicacions
- XML ha de ser compatible amb SGML
- Ha de ser fàcil crear programes que processin documents XML
- El nombre de funcionalitats opcionals de XML s'ha de mantenir en un mínim absolut, preferiblement zero.
- Els documents XML han de ser intel·ligibles pels humans i raonablement clars.
- Un disseny de XML s'ha de poder preparar ràpidament.
- Un disseny de XML ha de ser formal i concís
- Els documents XML seran fàcils de generar
- La concisió en els marcadors XML té una importància mínima

XML ha esdevingut un estàndard en l'estructuració i intercanvi de informació, sobretot en transaccions entre sistemes diferents. La introducció de XML ha significat un canvi en la manera de gestionar processos en diferents camps:

- XML permet una major estructuració dels documents, repercutint en una major integració amb altres dades quan es tracta l'edició de bases de dades. Fins ara, quan es necessitava compartir informació entre diverses bases de dades es recorria a formats simples, com ara fitxers de text amb delimitadors de camp. XML facilita l'intercanvi d'estructures de dades, garantint que mai es perdran objectes (ni els seus atributs i herències) al llarg del procés d'integració de dades noves. Empreses com Oracle i Informix han obert seccions per l'estudi del XML.

- La integració del XML dins d'un sistema de comerç electrònic permet la normalització de gran part de les transaccions que es produeixen a la cadena de comercialització. Al mateix temps facilita la sortida final a internet i permet la utilització de sistemes que verifiquen l'integritat de les dades (servidors segurs). La unió Europea s'ha adonat de la facilitat amb que EDI «*Electronic Data Interchange*» es pot integrar amb XML i està

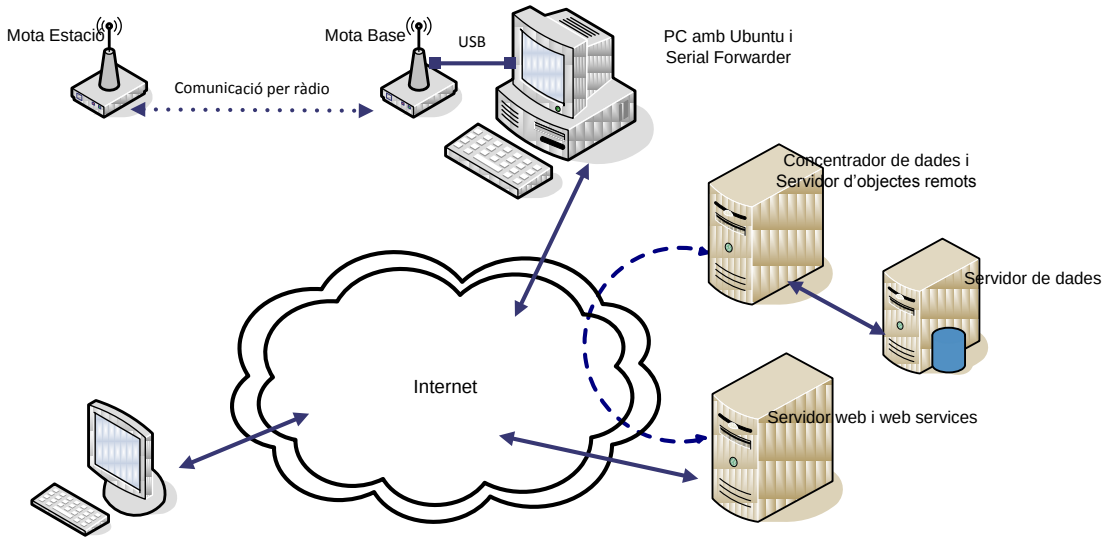
finançant diversos projectes (entre els que destaca el European XML/EDI pilot project) que pretenen realitzar una DTD de XML per EDI.

- Fent servir un sistema de metadades provat i estable (com ara RDF) per descriure el contingut d'un document que, alhora, ha estat realitzat en XML seguint una DTD ben formada, s'obté un sistema de informació robust que facilitarà les tasques clàssiques en la gestió de la informació, edició i recuperació especialment. És unir la potència en la recuperació de les metadades amb la versatilitat i flexibilitat en la representació que ofereix XML.

Capítol 2. Descripció funcional del projecte

2.1 Definició

El projecte que es descriu en aquest document té com objectiu el disseny d'una xarxa bàsica de sensors, la definició d'un protocol basat en XML que els elements de la xarxa faran servir per a comunicar-se entre ells i el desenvolupament d'un dispositiu pont que permeti la recepció, interpretació i distribució de la informació que circula per la xarxa cap al mon exterior.



6. Visió general del projecte

La xarxa bàsica està formada per dues motes idèntiques a nivell hardware, però que realitzen tasques molt diferents l'una de l'altra: la primera actua com a node base, enllaçant la xarxa de sensors amb els sistemes externs d'explotació i l'altra ho fa com a estació de captació de dades, atenent els seus sensors i transmetent la informació obtinguda a la mota base.

El protocol que regeix la comunicació està basat en XML; això vol dir que tots els missatges que circulen per la xarxa ho fan formalitzats en XML. El model d'encaminament triat és en estrella, però el protocol inclou un mecanisme de *handshake* inicial per tal d'identificar la mota base.

Les dades que circulen a la xarxa són transferides al sistema d'explotació per la mota base. Allà són processades, inserides en una base de dades per la seva futura consulta i exposades al món mitjançant un servidor d'objectes i una plana web.

2.2 Objectius

L'objectiu principal del projecte és demostrar la viabilitat d'una xarxa de sensors amb un protocol de comunicacions basat en XML. Atenent als diferents elements que hi formen part, podem fer el següent desglossament:

- **Permetre la convivència de diferents tipus de sensors**
  El protocol ha de permetre la presència de diferents tipus de motes i també l'aparició de nous tipus de sensors sense que això afecti a les funcionalitats ja implementades.

- **Admetre la lliure introducció de nous elements dins la xarxa**
  L'aparició de nous nodes a la xarxa ha d'estar plenament contemplada. Els nous nodes s'han de presentar i esperar a rebre el missatge de benvinguda de la mota base. Un cop identificats el node i la base ja es pot engegar la transmissió de dades.

- **Codificar la informació seguint un format públic i obert, com ara XML**
  Un de les principals metes d'aquest projecte és l'adopció del XML com a format pels missatges que es generen a la xarxa. L'ús d'aquest metallenguatge facilitarà la interpretació dels missatges en el sistema d'explotació i permetrà futurs intercanvis de dades cap a d'altres sistemes.

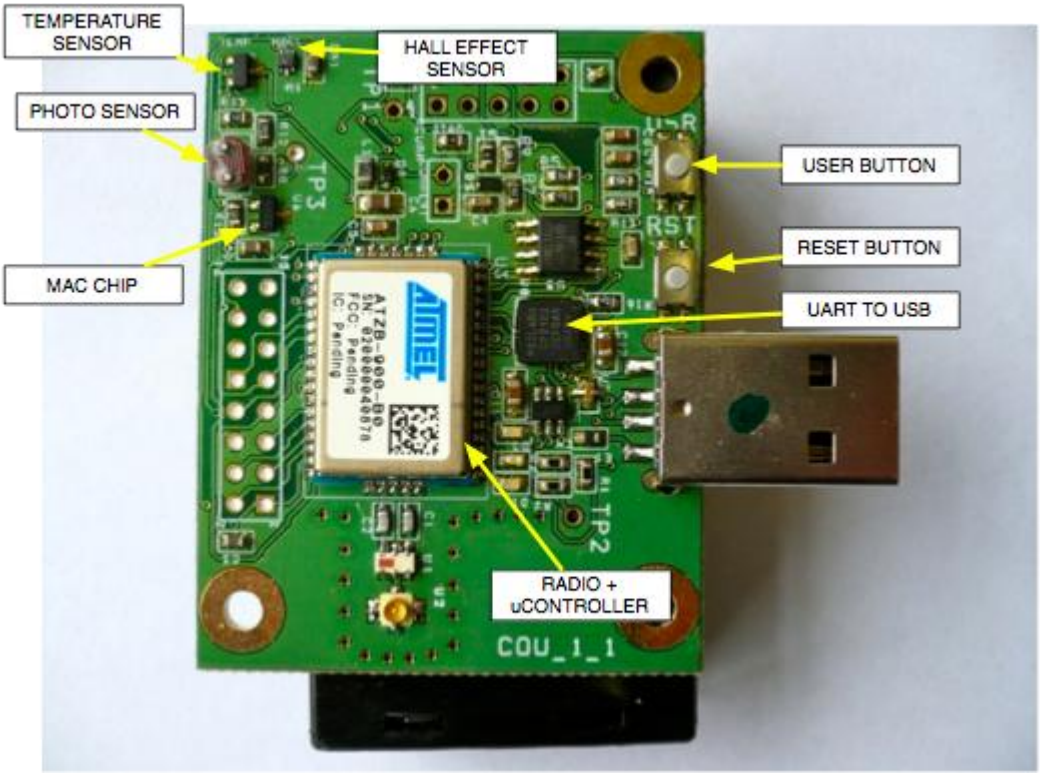
- **Preveure l'ús en el futur de l'encaminament multi-salt**
  Si bé el mètode d`'encaminament implementat és en estrella, cal que tant el protocol com la xarxa prevegin la utilització en un futur d'un encaminament multi-salt.

- **Demostrar que la informació que circula per la xarxa es pot emmagatzemar i publicar a internet**
  És indispensable que la informació provinent de la xarxa surti d'aquesta per ser tractada. El que es pretén és establir un mètode basat en la transparència, on no calgui transformar les dades provinents de la xarxa per poder ser tractades a l'exterior.

- **Demostrar que és possible interactuar amb la xarxa des del món exterior**
  La xarxa no pot ser vista només com una font d'informació, també ha de permetre la recepció d'ordres provinents de l'exterior.

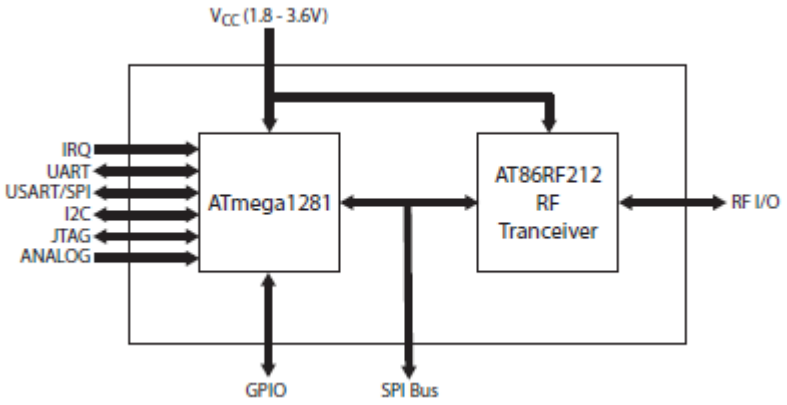
2.3 Node sensor o *Mota*

Per conformar la xarxa la UOC ha proporcionat un kit de desenvolupament format per una parella de dispositius basats en el xip ATZB-900-B0 d'Atmel i equipats amb sensor de temperatura, sensor de lluminositat i sensor d'efecte hal.



7. Imatge de la mota i detall de la ubicació dels seus elements

El xip ATZB-900-B0 incorpora en una sola càpsula el microcontrolador ATmega1281 i el transceptor AT86RF212. La càpsula conté també tota la memòria present a l'equip, i que consisteix en 128 Kb de memòria Flash pel sistema i el codi i 4 Kb de memòria RAM pel programa en execució



8. Detall de la CPU

Parameters	Condition	Range	Unit
On-chip Flash Memory size		128	Kbytes
On-chip RAM size		8	Kbytes
On-chip EEPROM size		4	Kbytes
Operation Frequency		4	MHz

9. Memòria present a la mota

2.3.1 Mota base

La mota base és l'encarregada de transferir els missatges presents a la xarxa cap al sistema d'explotació extern. També rebrà les ordres provinents de l'exterior i les adreçarà al destinatari adequat.

Per tant, la mota base manté dues tasques funcionant: una per atendre el canal de ràdio i un altre per atendre la comunicació amb el pc al que està connectat.

2.3.2 Mota sensor

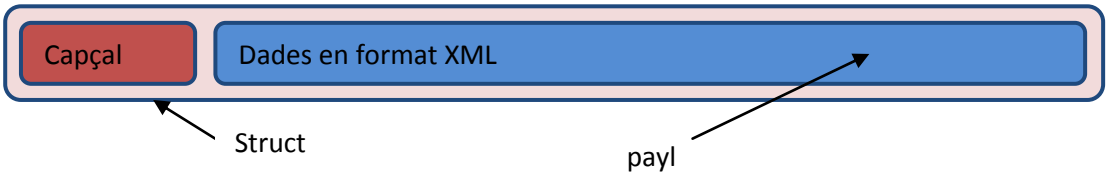
La mota sensor és l'encarregada de llegir periòdicament els seus sensors i fer arribar els respectius valors al sistema d'explotació en forma de missatges. El tractament de cada sensor es fa de forma independent, permeten així establir intervals de temps diferents per cada sensor; també es facilita l'aparició en el futur de nous sensors.

La mota sensor manté 4 processos funcionant, un per cada sensor a llegir (temperatura, lluminositat i nivell de bateria) i un altre per mantenir la comunicació amb la mota base. El sensor d'efecte hall és asíncron i no cal un procés per consultar-lo cíclicament; ell provocarà quan hi hagi detecció.

2.4 Protocol de comunicació

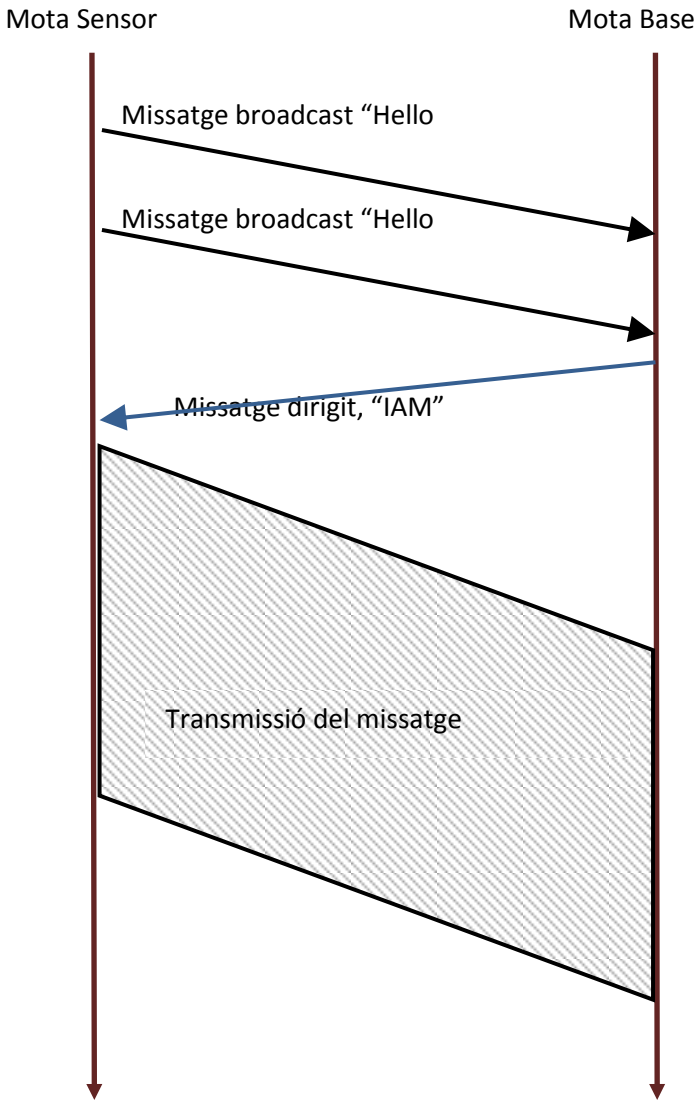
El protocol de comunicació estableix una descripció formal dels formats que han de presentar els missatges per poder ser intercanviats entre els diferents equips. Defineix també les regles que s'han de seguir per aconseguir-ho.

Tots els missatges que circulen per la xarxa de sensors, ho fan dins l'estructura ***message_t*** proporcionada pel tinyOS i necessària si es fan servir els components que el tinyOS ofereix per establir la comunicació, com ara l' ***ActiveMessageC***. Aquesta estructura preveu un espai per acollir les dades a transmetre. És dins d'aquest espai reservat, anomenat *payload* on es copiarà la informació que s'ha d'enviar.



El protocol de comunicació preveu que cada element present a la xarxa tingui un identificador únic, que és assignat en temps de compilació i càrrega[2] i especifica que tots els missatges han d'anar dirigits a un element específic, tret d'aquells missatges destinats, precisament a obtenir els identificadors .

El primer que ha de fer un node de la xarxa és obtenir l'identificador de la seva mota base; sense aquest identificador no pot realitzar cap transmissió de dades. Així, quan arrenca el seu sistema de comunicació envia un missatge de presentació, codificat com "HELLO" indicant el seu identificador i queda esperant resposta de la mota base, que respon amb un missatge "IAM" incloent el seu identificador. Un cop rebut el missatge "IAM" el node ja pot començar a transmetre. Si el node sensor no rep el missatge "IAM", repetirà cíclicament el "HELLO" fins a obtenir-ne resposta.



10. Esquema del Handshake

[2] En capítols posteriors s'explica com s'efectua aquesta assignació.

2.5 Sistema d'explotació

El sistema d'explotació és el mecanisme situat fora de la xarxa encarregat de obtenir els missatges provinents d'aquesta, processar-los, enregistrar-los a una base de dades per futurs tractaments i exposar-los al món mitjançant un servidor d'objectes remots i una plana web.

2.5.1 Serial forwarder

El primer element d'aquest mecanisme és el Serial Forwarder. És una eina proveïda pel propi tinyOS i desenvolupada en java, que corres sobre un sistema linux i s'encarrega de rebre tots els paquets provinents de la mota base.

El serial forwarder espera que d'altres programes, executant-se a la mateixa o a una altra màquina, es connectin mitjançant TCP-IP a un socket que manté obert. Tots els missatges rebuts són retransmesos a totes les connexions entrants en aquest socket.

La raó per incloure aquest primer pas en el sistema d'explotació és facilitar la transició de la informació cap el sistema Windows, que és el sistema triat per desenvolupar el gruix del sistema d'explotació.

2.5.2 Servidor d'objectes remots

S'ha definit un objecte software per encapsular les funcions de comunicació i gestió de la base de dades, que són les tasques corresponents a aquesta capa.

Per una banda, l'objecte llença un procés que atendrà la comunicació amb el *SerialForwarder* i rebrà la informació present a la xarxa; per l'altra, manté una estructura de dades que replica l'estructura de la xarxa i conserva els últims 100 missatges rebuts de cada node. Això permet la consulta en temps real tant dels valors actuals dels sensors com dels valors de moments anteriors.

2.5.3 Servidor web

El servidor web és l'última etapa del projecte. La seva raó de ser és demostrar la viabilitat de tot el desenvolupat prèviament. Consisteix en una senzilla plana web que mostra els últims valors llegits de cada sensor i l'últim missatge rebut de la mota i permet interactuar amb ella amb dos botons que representen ordres a enviar.

La plana web és accessible en proves a:

http://xm.selfip.net:6789/tfc.aspx

Capítol 3. Desenvolupament

3.1 Recursos

Per tal de desenvolupar el projecte, es disposa dels següents recursos:

a) **Desenvolupament de la xarxa i el firmware dels sensors**
El desenvolupament del programa dels sensors es realitzarà en llenguatge nesC per tinyOS. La plataforma per dur-ho a terme és el sistema operatiu Ubuntu.

- Dos equips sensors lliurats per la UOC
- Ordinador de sobretaula amb Ubuntu 10.04 LTS i tinyOS-2.1.1
- Ordinador portàtil amb Ubuntu 10.04 LTS i tinyOS-2.1.1

b) **Desenvolupament de la web i *web services***
El desenvolupament de la web i els web services es farà en llenguatge C# i ASP.NET en una plataforma Windows.

- Ordinador de sobretaula amb Windows XP i Visual Studio .NET

3.2 Programari de les motes

El programari de les motes està organitzat en dos projectes, un per la mota que actua de pont entre la xarxa de sensors i un PC al que es connecta per USB i un altre per la mota que actua com estació de sensors.

3.2.1 Codi comú

Cadascun dels projectes té el seu propi codi diferenciat, però també és necessari que comparteixin informació i recursos. La informació compartida, amb les definicions dels canals de comunicacions, els objectes utilitzats i la codificació dels missatges entre d'altres dades, es troben en un header compartit, el *tfcNet.h*. Per altra banda, tots dos projectes comparteixen un únic protocol que exigeix la codificació i descodificació dels missatges en XML. Per realitzar aquestes tasques, s'ha dissenyat un component capaç de codificar i descodificar XML.

Header tfcNet.h

Header global del projecte on hi ha les definicions dels codis de missatge, constants generals i la definició dels objectes comuns.

Component *XMLparser*

Una de les premisses fonamentals d'aquest projecte és que la informació que circula per la xarxa de sensors ho fa en format XML. Era necessari, per tant, dissenyar un codificador/descodificador que encaixés en les limitacions de memòria que imposen el hardware i el tinyOS.

Els missatges que circulen per la xarxa han de poder ser presentats tant en format XML per ser transmesos, com en format binari per ser tractats. La transició entre els dos formats està encapsulada dins el component *XMLparser*. Cada una de les possibles presentacions té

associada una struct c per tal de ser manipulada des del codi. Quan es presenta en format string XML, ho fa dins l'struct **xmlEnvelopeMsg** mentre que quan ho fa en format binari, és presenta com una struct **tfcMessage**.

Estructura **tfcMessage**:

```
typedef nx_struct
tfcMessage
{
    nx_uint16_t senderId;
    nx_uint16_t
receiverId;
    nx_uint16_t infoType;
    nx_uint16_t infoData;


} tfcMessage;
```

Estructura **xmlEnvelopeMsg**:

```
typedef nx_struct xmlEnvelopeMsg
{
    nx_uint8_t
bufferXML[XML_BUFFER_LENGTH];

} xmlEnvelopeMsg;
```

L'XML parser ofereix dues funcions:

```
command error_t XMLparser.codeXML(tfcMessage *tfcmsg, xmlEnvelopeMsg *ptr)
command error_t XMLparser.decodeXML(xmlEnvelopeMsg *ptr, tfcMessage
*tfcmsg)
```

Que ens serviran per passar d'un format a l'altre. El pas de binari (**codeXML**) a string XML és senzill, doncs només cal construir un string amb el format correcte i fent servir les funcions de llibreria standard del compilador c (l'sprintf) es resol amb una línia de codi. En canvi, el pas contrari (**decodeXML**) és molt més complex, ja que l'XML ofereix diferents sintaxis per a la mateixa informació i no fixa cap ordre en la presentació dels diferents atributs de l'objecte. Així, ha calgut escriure un intèrpret capaç d'entendre totes aquestes sintaxis i de trobar els valors cercats sigui quina sigui la seva posició dins de la cadena XML.

Degut a que el tinyOS limita el manegament de memòria complicant encara més la descodificació, el parser resultant té algunes limitacions. Tot ell està basat en una funció (find_value) que recorre la cadena XML identificant els nodes i els seus fills per trobar el valor d'un atribut donat dins un node donat. El valor retornat indica si hi hagut èxit en la cerca, i el valor de l'atribut demanat es posa a la variable **str_valor**.

Diferents sintaxis XML acceptades:

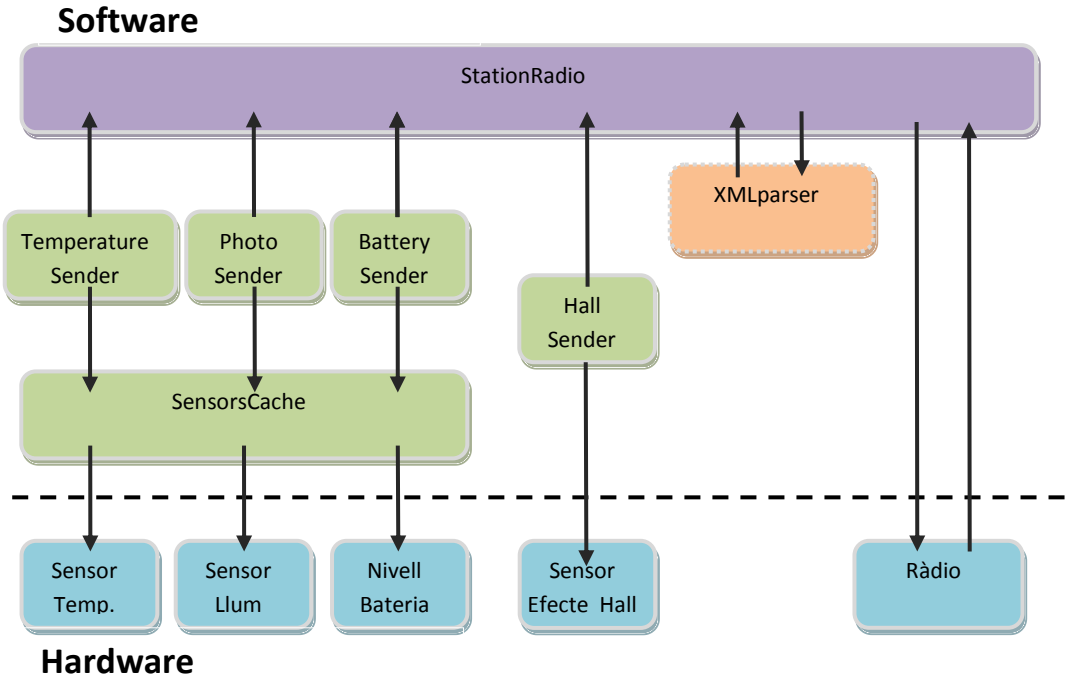
```
<TFCMSG>
    <From>senderId</From>
    <To>receveiderId</To>
    <Type>infoType</Type>
    <Data>infoData</Data>
</TFCMSG>

<TFCMSG From=senderId To=receveiderId Type=infoType Data=infoData />

<TFCMSG From=senderId>
    <To>receveiderId</To>
    <Type>infoType</Type>
    <Data>infoData</Data>
</TFCMSG>
```

3.2.2 Projecte tfcStation

El projecte tfcStation és el corresponent a la mota d'obtenció de dades. Està dividit en dos grans blocs, un encarregat de llegir els diferents sensors presents a la mota i un altre per establir i mantenir la comunicació amb la mota base.



11. Esquema tfcStation

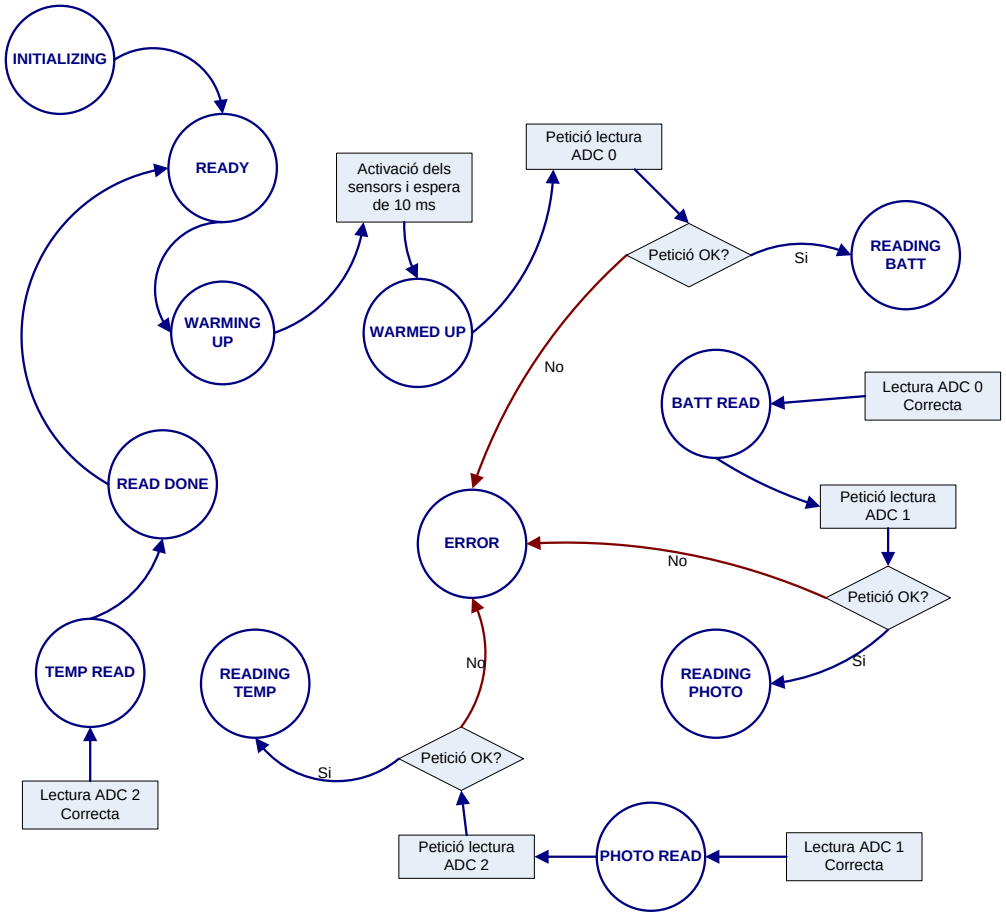
Component *tfcStation*

Definició del projecte i dels components continguts.

Component SensorsCache

És l'encarregat de llegir periòdicament els diferents sensors presents a la mota i emmagatzemar-ne l'última lectura de cadascun. Actua com a memòria intermitja entre el hardware i els components que envien les lectures per la xarxa i garanteix la sincronia entre els diferents accessos al hardware seqüenciant les interrupcions de lectura. Ofereix valors amb una antiguitat màxima d'un segon.

El seu funcionament està basat en un Timer i una màquina d'estats. La màquina d'estats arma el sistema de comunicació i després llegeix els tres sensors disponibles: nivell de bateria, temperatura i lluminositat de forma seqüencial. La màquina d'estats s'atura en cada petició de lectura del hardware fins que aquest retorna el valor; l'event de retorn provoca el canvi al següent estat. Aquesta seqüència d'estats es fa sense esperar el timer principal del component. Un cop llegits tots els sensors, es torna a l'estat preparat i s'espera fins al següent cicle de temps.



12. Diagrama de fluxe de *SensorsCache*

Component StationRadio

La seva missió és atendre la comunicació amb la mota base. Així, la seva primera tasca és trobar la base llençant un missatge broadcast demanant-li que s'identifiqui. Un cop localitzat l'interlocutor, ja pot començar a enviar-li missatges.

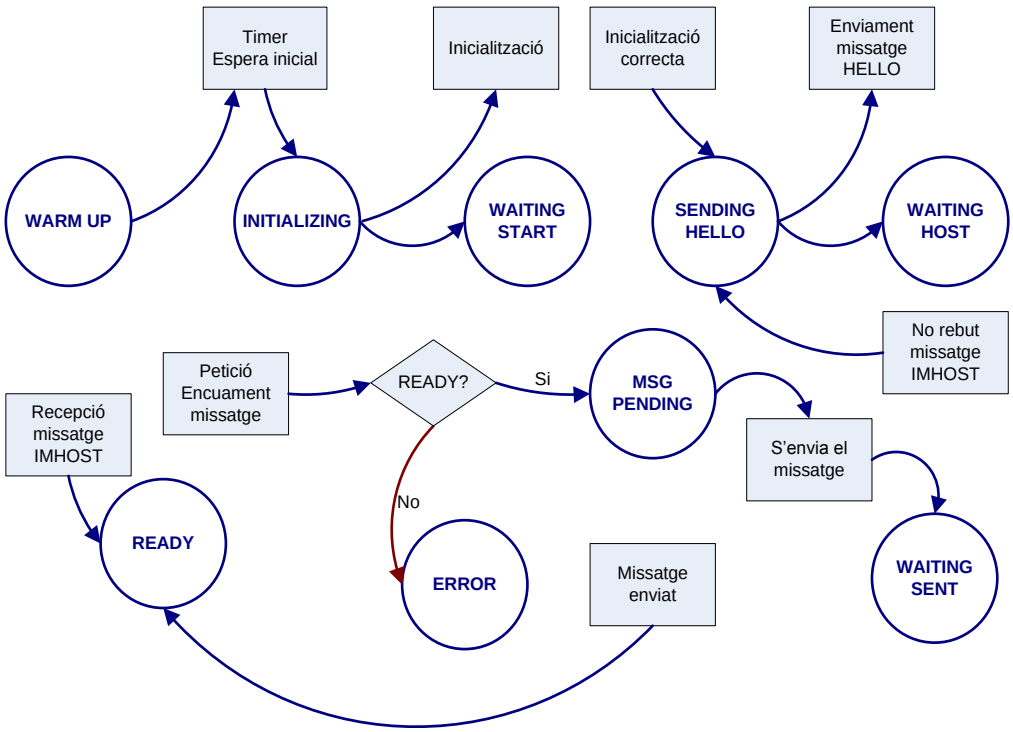
Com s'ha descrit en l'apartat del *XMLparser*, cal disposar de dos tipus d'objectes per completar una transmissió:

```
// punters a buffer XML, un per la recepció i un altre per l'enviament
xmlEnvelopeMsg *xmlPtrTX;
xmlEnvelopeMsg *xmlPtrRX;
// structs per serialitzar/deserialitzar l'XML
tfcMessage tfcMsgTX;
tfcMessage tfcMsgRX;
```

Per evitar col·lisions, es defineix un parell d'objectes per cada sentit de la comunicació, un per enviar i un altre per rebre.

L'enviament es gestiona amb una petita cua (d'un sol missatge) on s'emmagatzema temporalment el missatge pendent. Els altres components sol·liciten l'enviament d'un missatge mitjançant una funció i continuen la seva execució sense esperar la finalització de l'enviament. Quan aquesta es produeix, són notificats mitjançant un event.

Les etapes d'execució d'aquest component estan guiades per una màquina d'estats.

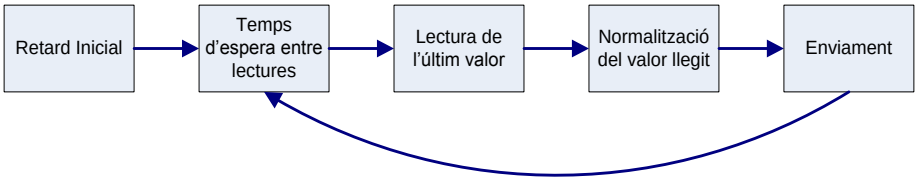


13. Diagrama de fluxe de *StationRadio*

La transmissió de missatges es realitza transformant un objecte **tfcMessage** en un **xmlEnvelopeMsg** i omplint el *payload* de l'struct **message_t** del tinyOS.

Components BatterySender, TemperatureSender, PhotoSender

Són els encarregats d'enviar a la mota base les lectures dels sensors de Nivell de bateria, temperatura i lluminositat respectivament. Ja s'ha explicat que el component *SensorsCache* és l'encarregat d'accedir al sensors i conservar l'última lectura. Aquest components accedeixen periòdicament a aquests valors, els converteixen a les unitats adequades i els lliuren a l'StationRadio per tal d'enviar-les cap a la mota base.



14. Diagrama de fluxe de *BatterySender*, *TemperatureSender* i *PhotoSender*

La conversió a les unitats adequades es realitza transformant el valor obtingut en un voltatge i convertint aquest a volts, temperatura, o lux[3] seguint les fórmules de cada sensor.

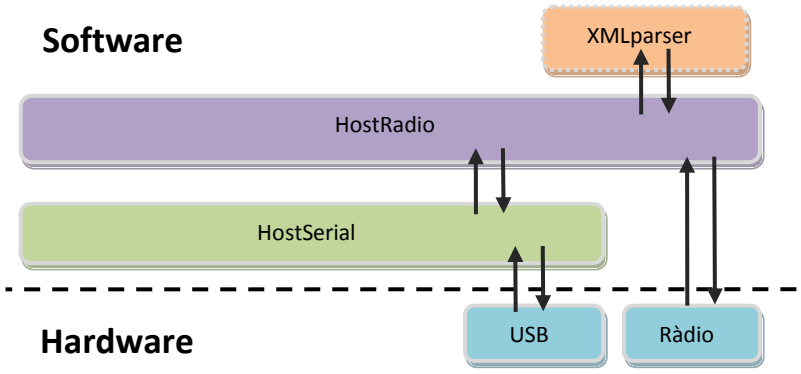
Cada un dels components introdueix un temps d'espera diferent (definits a *tfcNet.h*) abans de llençar el temporitzador principal amb la finalitat de separar en el temps els enviaments de missatges i evitar així col·lisions per enviaments simultanis.

Component *HallSender*

El detector d'efecte Hall és un detector asíncron. Aquest component s'encarrega d'inicialitzar el hardware i esperar que es produeixi un event de detecció; llavors, llença un missatge cap a la mota base fent servir l'*StationRadio*.

3.2.3 Projecte tfcHost

Aquest és el projecte que corre a la mota base. La seva funció és conduir el tràfic present a la xarxa cap al PC al que està connectat per USB i a l'inrevés, introduir a la xarxa els missatges provinents del PC.



15. Esquema de tfcHost

Component *tfcHost*

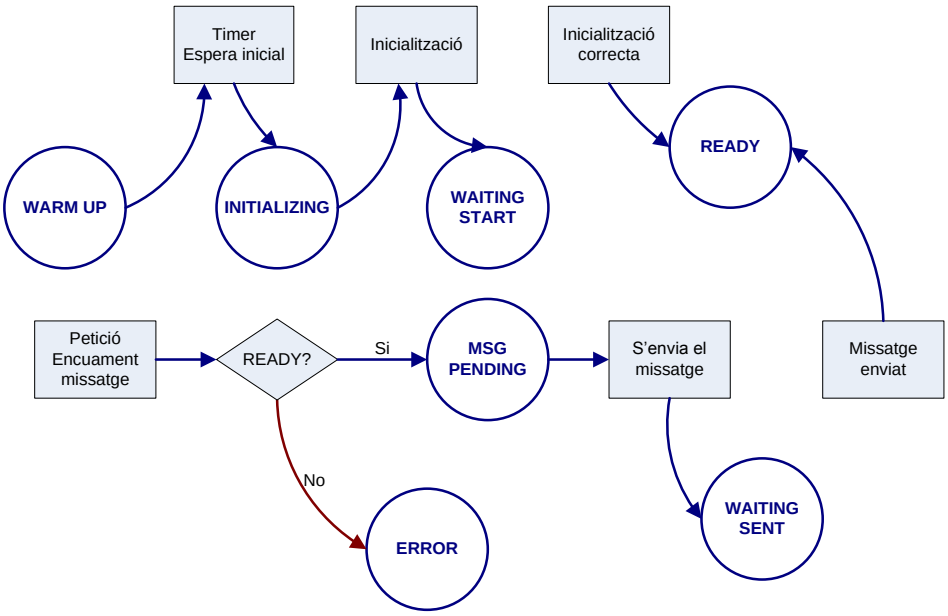
Definició del projecte i dels components continguts.

Component HostRadio

Encarregat de manegar les comunicacions per ràdio amb la xarxa de sensors. La seva missió principal és replicar tot el tràfic de la xarxa de sensors cap al pc, però també ha de respondre als missatges de handshake que envien les motes per identificar la base. Per això li cal descodificar els missatges rebuts mitjançant l'*XMLparser* abans de reenviar-los.

El funcionament d'aquest component està regit per una màquina d'estats.

[3] En el cas del sensor de llum, finalment s'ha optat per enviar un valor de referència entre 0 i 4 per indicar el grau de lluminositat detectada, essent 0 lluminositat nul·la i 4 lluminositat màxima

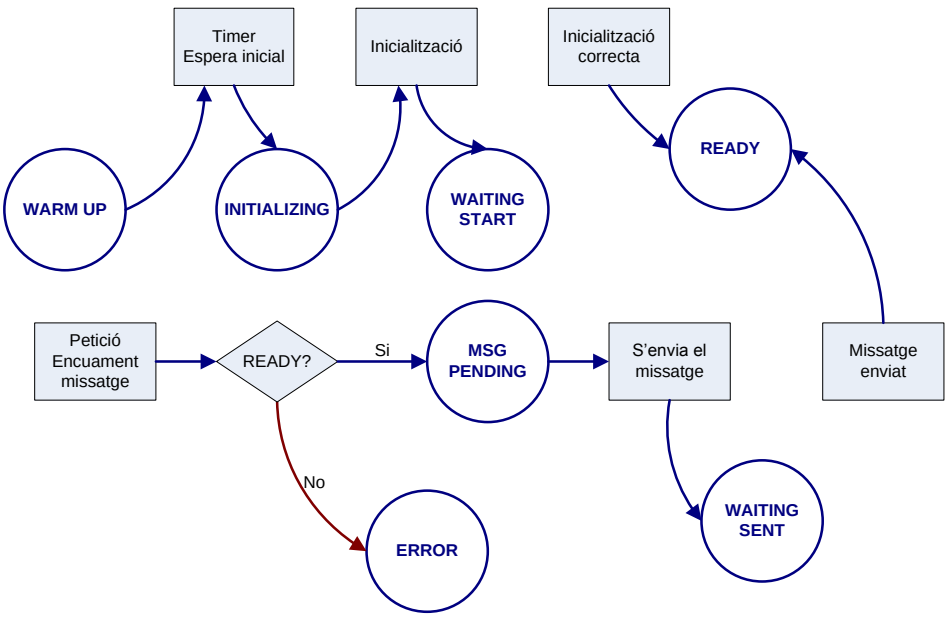


16. Diagrama de fluxe de *HostRadio*

També atén les peticions que des del PC s'envien a la xarxa. Per l'enviament implementa un mecanisme semblant al ja descrit a mòduls anteriors, amb un petita cua d'un missatge i una funció per demanar l'enviament d'un missatge.

Component *HostSerial*

Es el responsable de controlar les comunicacions amb el PC, al que es connecta mitjançant el port USB. Igual que en altres mòduls descrits anteriorment, disposa d'una cua d'un missatge per emmagatzemar els missatges pendents. L'adició de missatges es fa de forma diferida fent servir una funció de retorn immediat.



17. Diagrama de flux de HostSerial

3.3 Programari de la banda PC

La xarxa de sensors presentada en aquest projecte es complementa amb dues aplicacions sobre PC: un servidor d'objectes remots i una pàgina web. El servidor d'objectes remots és l'encarregat de obtenir els missatges que circulen a la xarxa de motes, enregistrar-los a una base de dades i exposar-los per què d'altres aplicatius en facin us. Un exemple d'això és la pàgina web, que es connecta al servidor d'objectes, ofereix l'estat dels sensors presents en temps real i permet interactuar amb ells.

3.3.1 Serial Forwarder i remote object server (TFC Sockets)

El Serial Forwarder és el punt d'unió entre el programari de la banda PC i la xarxa de motes. És una utilitat que actua com un proxy de la xarxa, permetent que d'altres aplicacions es connectin i interactuïn amb les motes. En aquest projecte, a més, el Serial Forwarder permet fer el salt de l'UBUNTU on corre el tinyOS, al Windows, on s'executen els projectes que es descriuen a continuació.

El remote object server és un servidor d'objectes basat en la tecnologia Remote Objects de l'entorn .NET de Microsoft. De fet, exposa un únic objecte, anomenat **cTFCmanager.**

El servidor s'associa al port 12500 i el servei remot s'arrenca en mode *Singleton*, el que vol dir que tots els clients que es connectin compartiran una sola instància de l'objecte; l'alternativa és el mode *singlecall* en el cada client posseeix una instància pròpia de l'objecte.

La classe **cTFCmanager** té dues finalitats:

- Rebre i registrar a la base de dades tots els missatges provinents de la xarxa de motes
- Mantenir en memòria una llista de les motes existents a la xarxa amb els últims 100 missatges rebuts de cadascuna.

El **cTFCmanager** es connecta al SerialForwarder mitjançant una funció que rep com a paràmetres la IP i el port on trobar-lo. Un cop iniciada la comunicació, es llença un thread encarregat de gestionar la comunicació. El SerialForwarder exigeix un intercanvi inicial de bytes, una mena de handshake per començar a transmetre missatges, així que abans de començar a rebre informació cal satisfer aquest requisit.

Establerta definitivament la connexió, s'inicia el cos del procés, on s'aprofita els forats que deixa la recepció per enviar els possibles missatges pendents.

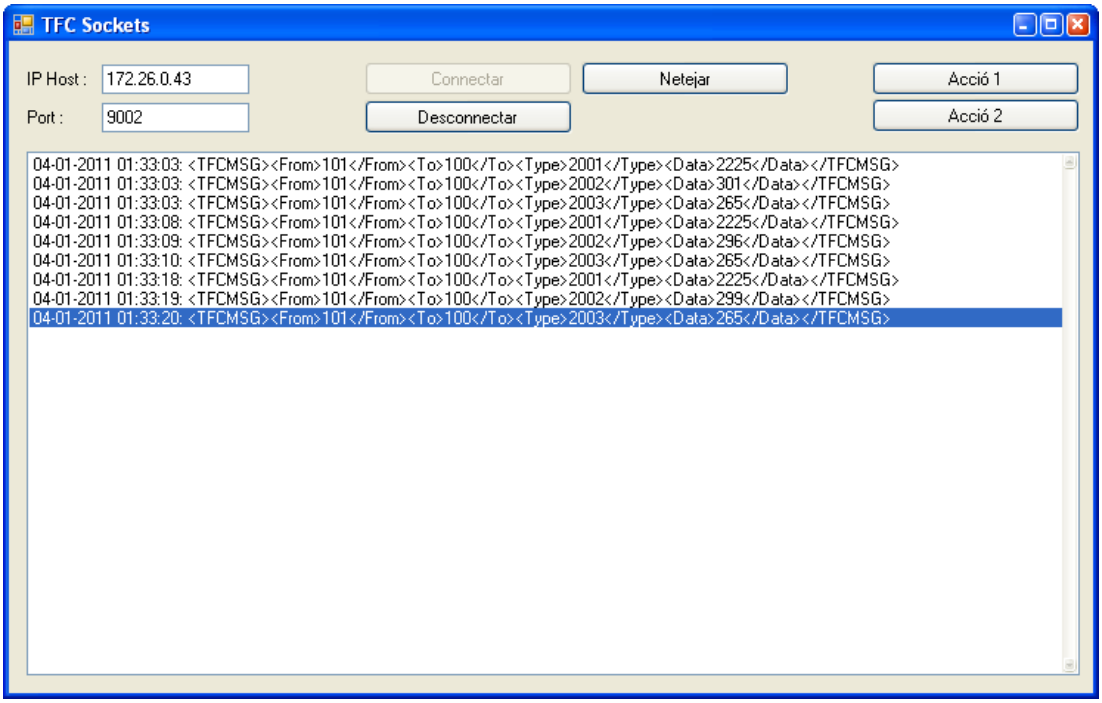
Quan, es rep un missatge de les motes, es converteix l'array de bytes rebuts en un string del que a continuació, fent servir el serialitzador XML incorporat al .NET, obtenim un objecte del tipus **TFCMSG** (és el mateix nom del node principal que envien les motes en XML).
L'objecte **TFCMSG** a més dels atributs presents en el format XML implementa una funció per gravar el missatge rebut a una base de dades. L'objecte **cDB** és una petita classe que encapsula la connexió a les dades.

La ubicació de la base de dades s'especifica mitjançant l'arxiu de configuració del programa, que es descriurà a l'apartat de compilació i instal·lació.

Per finalitzar el procés del missatge rebut, es crida a la funció processMessage, encarregada de registrar el missatge dins la llista de motes que manté el **cTFCmanager**. Aquesta llista està basada en dos classes, **cStation** que representa una mota i **cLectura** que representa un missatge rebut.

La classe **cStation** manté una llista amb els últims 100 missatges rebuts de la seva mota. La funció processMessage busca la **cStation** corresponent a l'emissor del missatge i l'afegeix a la seva llista.

El servidor d'objectes està acompanyat d'un formulari que permet interactuar amb la xarxa i explotar les prestacions que ofereix el **cTFCmanager**

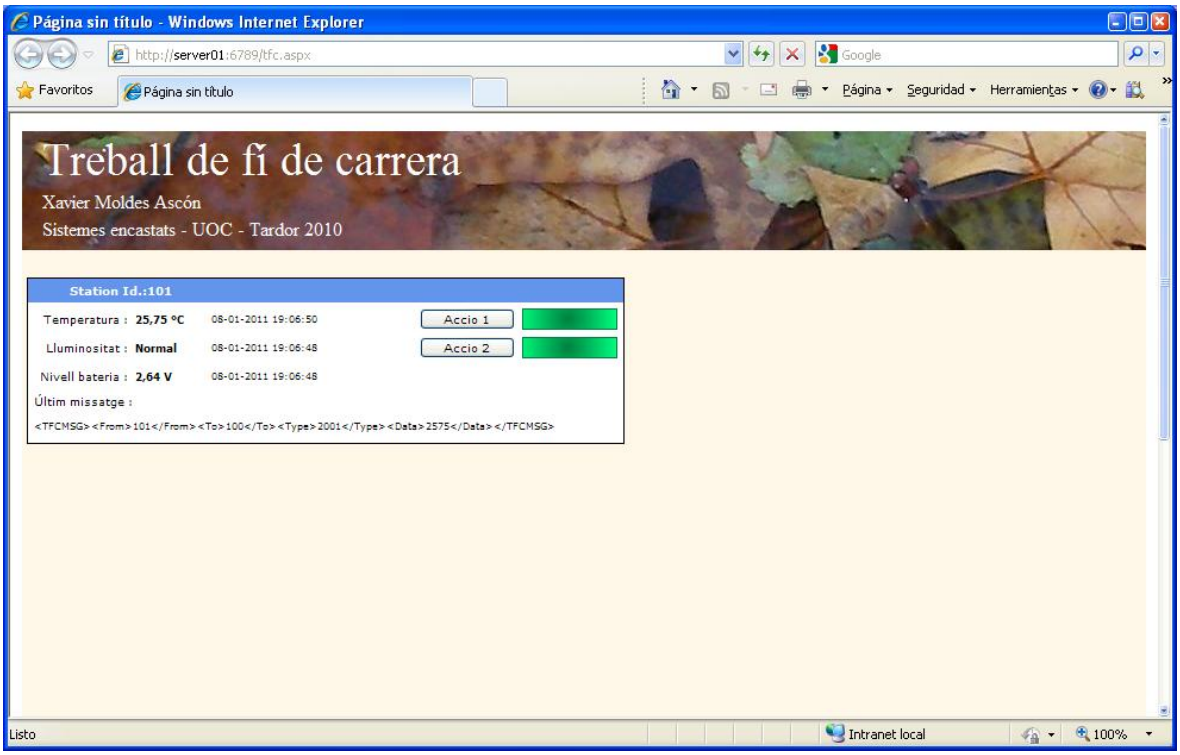


18. Aspecte del formulari de control del cTFCmanager

Els botons Connectar i Desconnectar activen i desactiven la comunicació amb el SerialForwarder, la listbox inferior mostra els últims missatges rebuts, el botó Netejar buida la listbox i els botons d'acció envien un missatge a la mota per executar les accions associades (alternar els leds 1 i 2 respectivament).

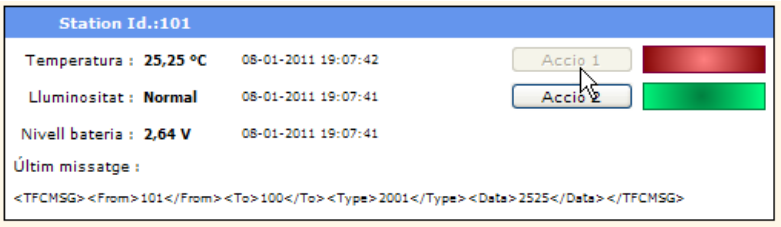
3.3.2 Projecte web

La pàgina web és un altre exemple de l'ús dels objectes remots. Consisteix en un user control del ASPx (que en el fons no és més que una porció de codi HTML amb codi C# associat que s'inclou en una pàgina principal) que mostra la última lectura dels sensors de la única mota present a la xarxa.



19. Aspecte de la plana WEB

També permet la interacció amb les motes de la xarxa mitjançant els botons d'acció que queden inhabilitats mentre no es rep la confirmació per part de la mota de que l'acció ha estat executada (la "llum" verda passa també a vermella quan s'espera la confirmació de l'acció):



20. Detall d'espera de confirmació d'acció

L'actualització periòdica de la plana web és responsabilitat d'un timer que força el refresc cada segon. En l'event del timer es connecta amb el servidor d'objectes remots i s'obté la instància del **cTFCmanager**.

Després es demanen les últimes lectures dels diferents sensors i es mostren. També es mostra l'últim missatge rebut en format XML.

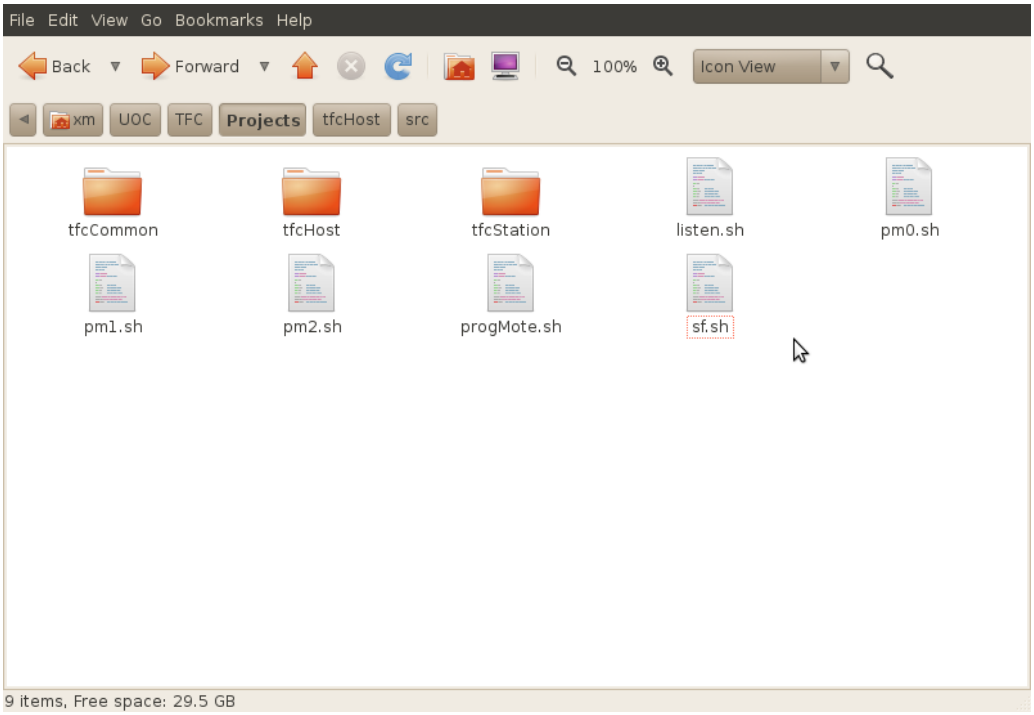
Els botons d'acció actuen de forma similar: obtenen la instància remota del **cTFCmanager** i li demanen l'execució de l'acció.

3.4 Compilació i Instal·lació

Juntament amb aquest document s'ha lliurat dos arxius ZIP que contenen tot el que cal per poder compilar, instal·lar i executar tots els projectes descrits. Caldrà, però, disposar dels entorns de compilació i execució adequats a cada un d'ells.

3.4.1 Compilació i Instal·lació a les motes

El projecte de les motes es troba en el fitxer **tfc_motes.zip** s'ha de desempaquetar en la carpeta que destinem a allotjar el projecte; es crearà dins d'aquesta una nova carpeta anomenada *Projects* que conté tot el necessari per compilar i instal·lar els projectes. Concretament hi ha tres carpetes que contenen els fonts de cada un dels dos projectes (*tfcStation* i *tfcHost*) i els arxius comuns a tots dos (*tfcCommon*). També hi ha una sèrie de fitxers shell script (*.sh) que es descriuran més endavant en aquest mateix capítol.



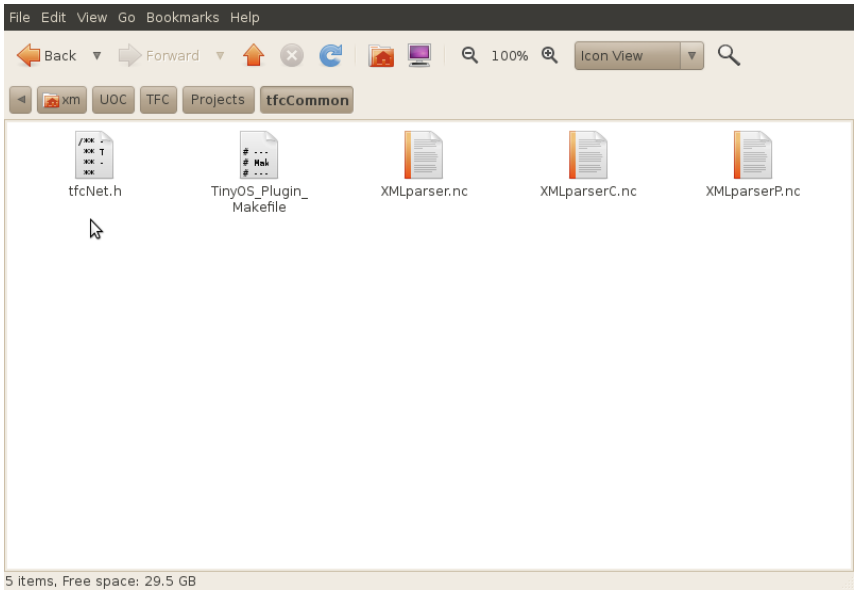
21. Detall carpeta Projects

L'edició del codi s'ha dut a terme amb l'entorn de desenvolupament Eclipse i la perspectiva per tinyOS recomanada a l'enunciat del projecte. En canvi, la compilació i la programació de les motes s'ha fet des d'una consola del sistema, amb les comandes pròpies del sistema (*make*) i el programa de transmissió *meshprog* també lliurat amb l'enunciat de la pràctica.

S'assumeix que el PC on es desempaquetarà i es compilarà el codi té correctament instal·lat el tinyOS adequat per les motes descrites en aquest document i el programa *meshprog*. De

no ser així, les instruccions per obtenir i instal·lar les versions adequades es troben a la plana web: http://cv.uoc.edu/app/mediawiki14/wiki/

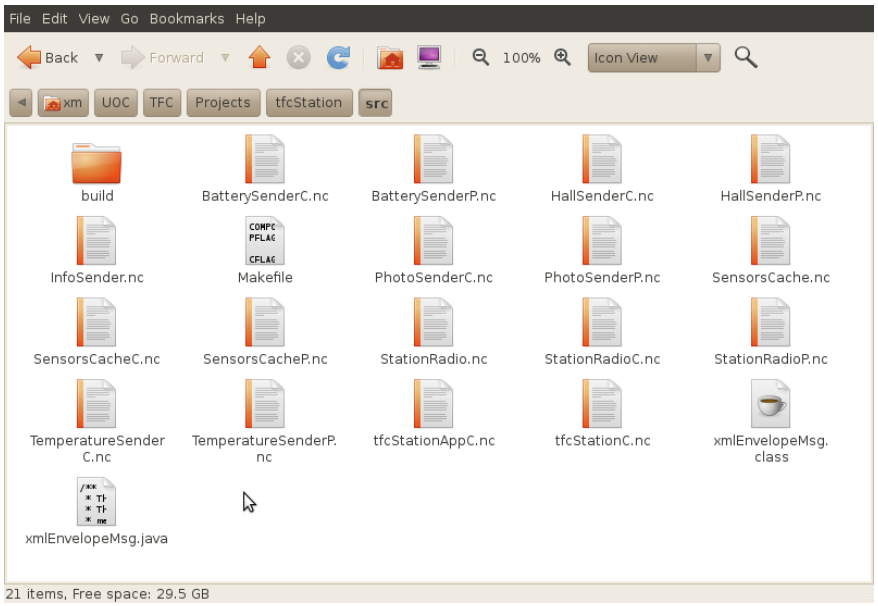
La carpeta *tfcCommon* conté codi que s'inclou als dos projectes, però no representa cap projecte en sí mateixa; és simplement un contenidor per al codi compartit que facilita la no duplicació de fitxers i els errors que això comporta.



22. Detall carpeta *tfcCommon*

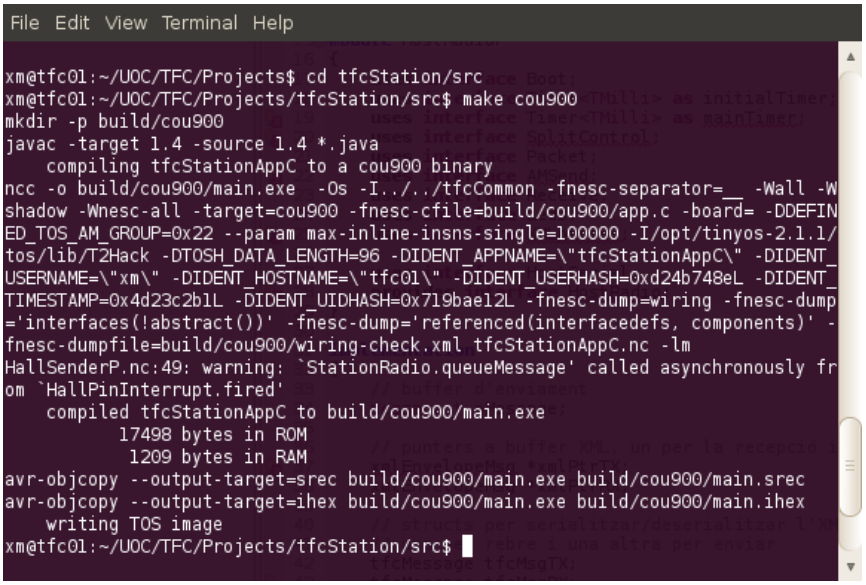
tfcStation

Per compilar el projecte *tfcStation*, que es correspon amb la mota que actuarà com a punt de recollida de dades, cal situar-se en una consola del sistema a la carpeta src dins de la carpeta *tfcStation*.



23. Detall carpeta *tfcStation*

La compilació es llença amb la comanda `make cou900` :



24. Detall compilació *tfcStation*

La comanda *make* pressuposa l'existència d'un arxiu Makefile dins la carpeta. Aquest fitxer conté les instruccions i condicions del compilador.

```
COMPONENT=tfcStationAppC
PFLAGS+=-I../../tfcCommon

CFLAGS += -I$(TOSDIR)/lib/T2Hack
CFLAGS += -DTOSH_DATA_LENGTH=96

BUILD_EXTRA_DEPS += xmlEnvelope.class
CLEAN_EXTRA = *.class xmlEnvelopeMsg.java

xmlEnvelope.class: $(wildcard *.java) xmlEnvelopeMsg.java
    javac -target 1.4 -source 1.4 *.java

xmlEnvelopeMsg.java:
    mig java -target=null $(CFLAGS) -java-classname=xmlEnvelopeMsg
../../tfcCommon/tfcNet.h xmlEnvelopeMsg -o $@

include $(MAKERULES)
```

Cal destacar la inclusió d'algunes directives addicionals:

```
PFLAGS+=-I../../tfcCommon
```

per indicar al compilador que alguns fitxers necessaris es troben en una altra ubicació, concretament a la carpeta tfcCommon, on ja hem indicat que hi ha el codi comú.

```
CFLAGS += -I$(TOSDIR)/lib/T2Hack
```

per indicar al compilador que necessitarà llibreries ubicades en la ruta indicada. Es tracta de llibreries de C necessàries per compilar el *XMLparser*

```
CFLAGS += -DTOSH_DATA_LENGTH=96
```

Per establir un nou valor per a la variable `DTOSH_DATA_LENGTH` que conté la grandària màxima del payload del missatge que circula per la xarxa.

Si la compilació acaba correctament, com es mostra la imatge 24, cal assignar-li un identificador únic a la mota abans de programar-la. Això es fa amb la instrucció:

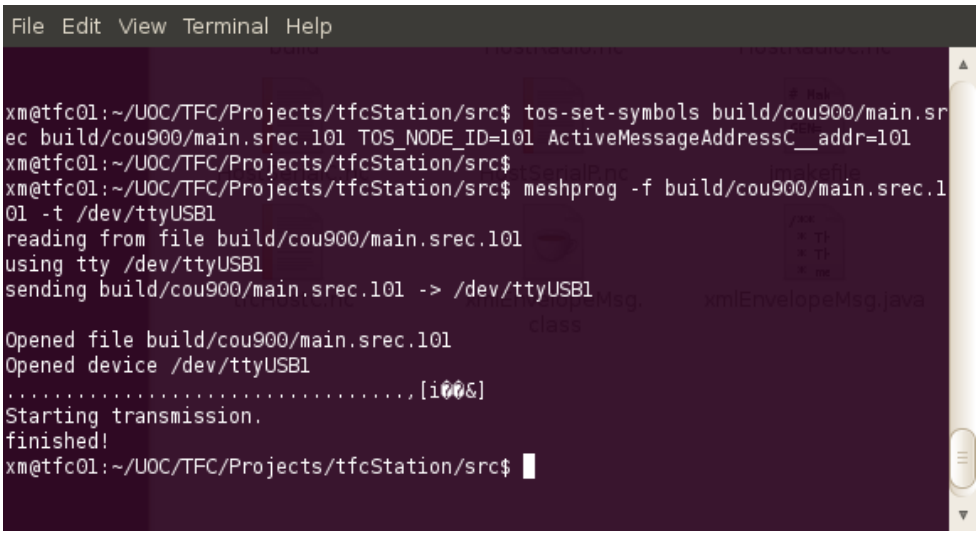
tos-set-symbols build/cou900/main.srec build/cou900/main.srec.101 TOS_NODE_ID=101 ActiveMessageAddressC__addr=101

que modifica el programa obtingut (main.srec) assignant-li un nou identificador de node per la mota.
Finalment cal enviar el nou programa a la mota. El procediment per fer això requereix que la mota es trobi connectada a un dels ports USB del PC i que aquest estigui identificat[4].

Per fer-ho, cal escriure:

```
meshprog -f build/cou900/main.srec.101 -t /dev/ttyUSB0
```

Que enviarà el programa amb el nou identificador (main.srec.101) a la mota connectada al USB 0. Un cop aparegui a pantalla el missatge Opened device /dev/ttyUSBX i uns punts vagin avançant per la línia, cal prémer el botó de reset de la mota i esperar la finalització de l'enviament. La mota ja està programada. En uns segons el nou programa s'iniciarà.

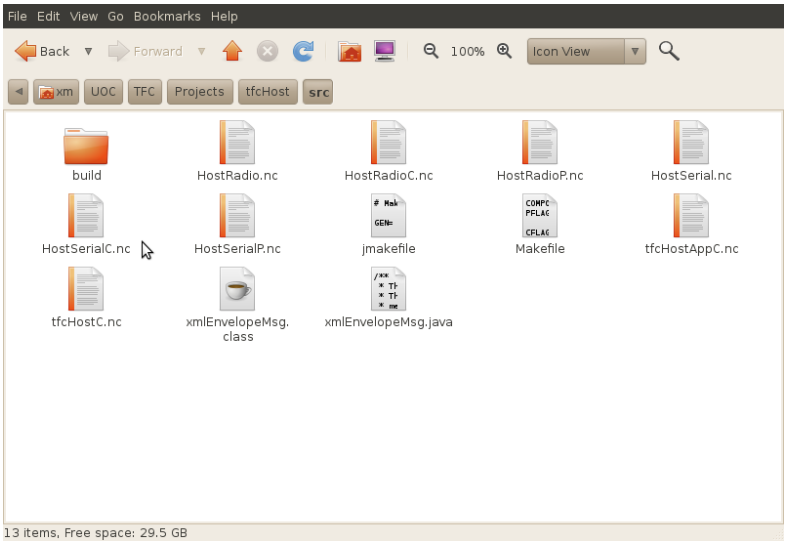


25. Detall programació de la mota

[4] Per identificar el port USB on es troba connectada la mota, cal executar la comanda **`ls /dev/ttyUSB*`** que ens retornarà la llista de ports utilitzats (/dev/ttyUSB0, /dev/ttyUSB1, etc) si n'hi ha més d'un caldrà desconnectar altres dispositius USB fins que només trobem el corresponent a la mota.

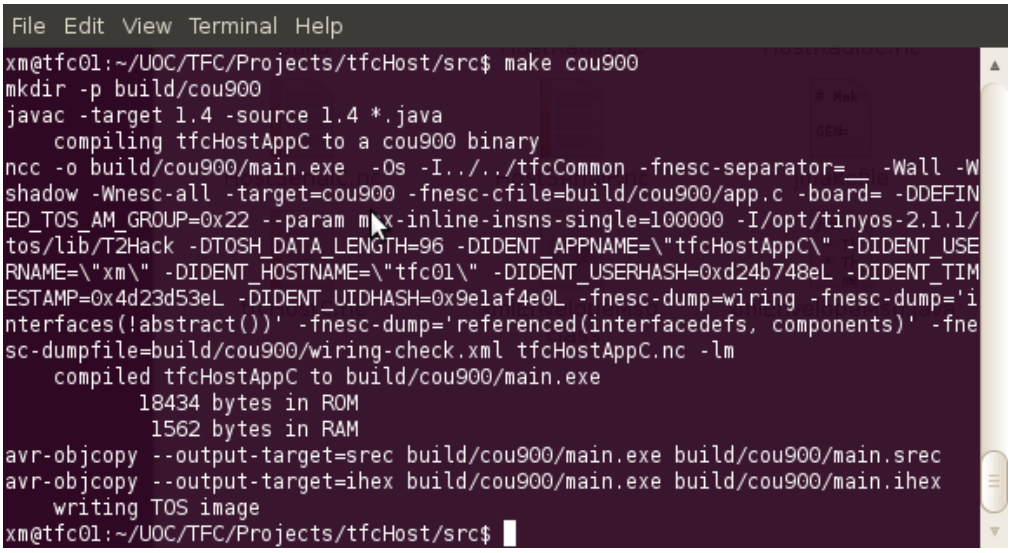
tfcHost

El procediment a realitzar al projecte de la mota base, el *tfcHost*, és gaire bé idèntic al descrit per el *tfcStation*, només caldrà canviar els noms de les carpetes i els identificadors de la mota.



26. Detall carpeta *tfcHost*

Situats en aquest cas a la ruta Projects/tfcHost/src, La compilació es llença també amb la comanda `make cou900` (el Makefile inclou les mateixes directives que l'anterior) :



27. Detall compilació tfcHost

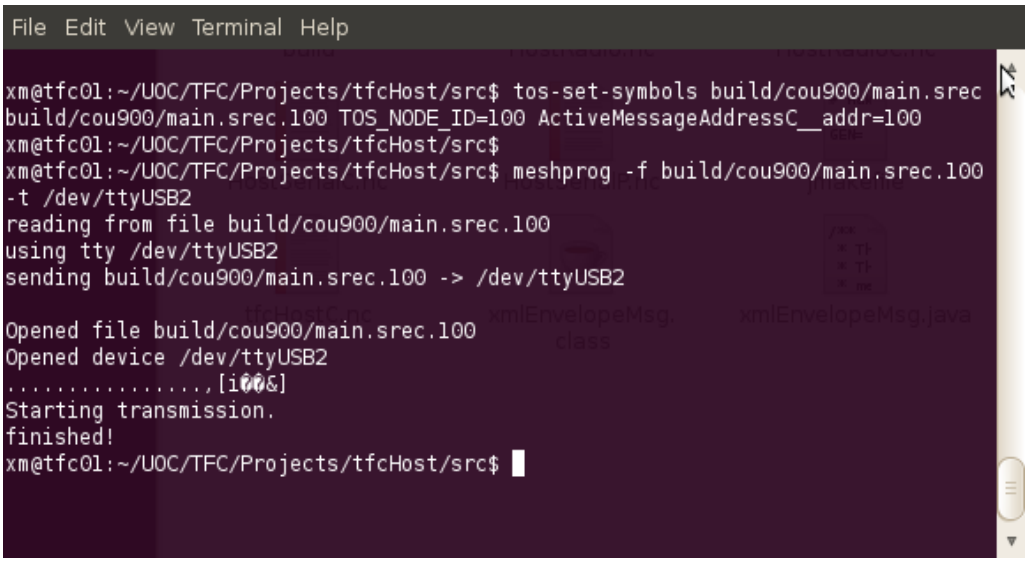
Si la compilació acaba correctament, com es mostra la imatge 27, cal assignar-li un identificador únic a la mota abans de programar-la. Això es fa amb la instrucció:

```
tos-set-symbols build/cou900/main.srec build/cou900/main.srec.100
TOS_NODE_ID=100 ActiveMessageAddressC__addr=100
```

En aquest cas l'identificador assignat a la mota és el 100.

Finalment es programa la mota amb la comanda:

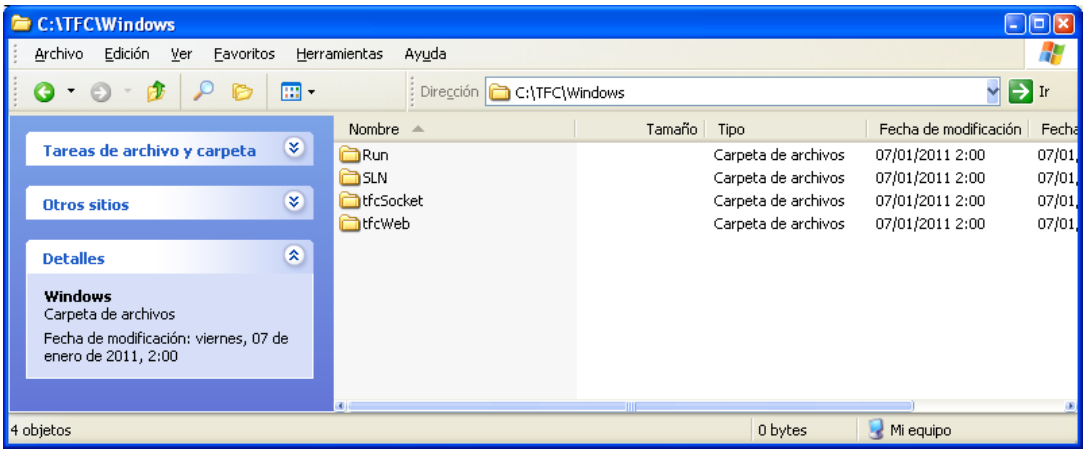
```
meshprog -f build/cou900/main.srec.100 -t /dev/ttyUSB0
```



28. Detall programació mota base

3.4.2 Compilació i instal·lació al PC

El fitxer ZIP que conté el codi font dels projectes sobre Windows es diu **tfc_windows.zip**. Un cop desempaquetat en una carpeta buida, generarà un arbre de directoris on hi ha arxivats els diferents arxius.



29. Vista carpeta projectes

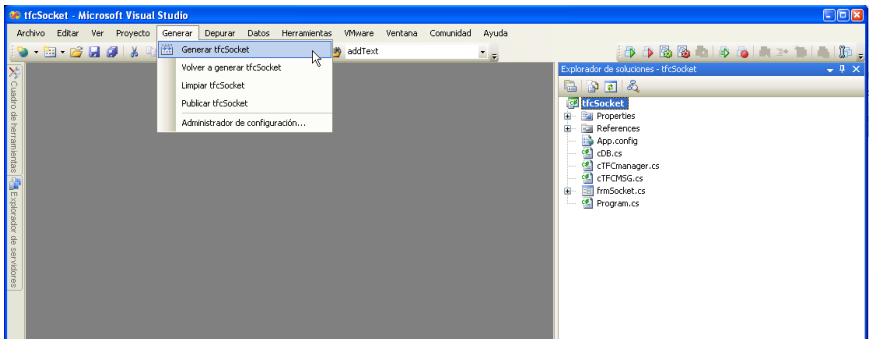
La carpeta SLN conté els arxius de solució per VisualStudio 2005[5]. Com ja s'ha esmentat en capítols anteriors, s'ha de disposar d'aquesta plataforma per poder generar els projectes Windows. La carpeta tfcSocket conté els fonts per el projecte del servidor d'objectes i la carpeta tfcWeb conté els arxius necessaris per obtenir la pàgina web. Finalment, la carpeta

[5] probablement funciona amb versions posteriors, però el desenvolupament s'ha fet en aquesta plataforma

Run contindrà l'executable del projecte tfcSocket i els arxius necessaris per el seu correcte funcionament.

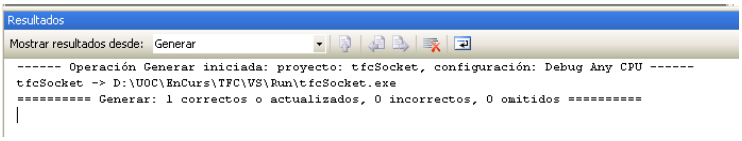
Compilació i instal·lació del servidor d'objectes

Per compilar el programa només cal carregar la solució tfcSocket.sln i ordenar la generació del projecte mitjançant la opció *Generar tfcSoket* del menú *Generar* :



30. Detall compilació tfcSocket

Un cop generada l'aplicació obtindrem un missatge confirmant que la compilació ha anat bé.



31. Detall resultat compilació tfcSoket

El resultat de la compilació és desat a la carpeta Run, dins de la carpeta principal dels projectes Windows. Allà es troba, junt a l'executable acabat de generar, l'arxiu de configuració que ha d'acompanyar-lo, el tfcSocket.exe.config:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
        <appSettings>
        <add key="RemoteObjectServer" value="tcp://localhost:12500/cTFCmanager.soap"/>
        <add key ="DBconnection" value="server=SERVER01;uid=tfc;pwd=tfc;database=TFC"/>
        </appSettings>
</configuration>
```

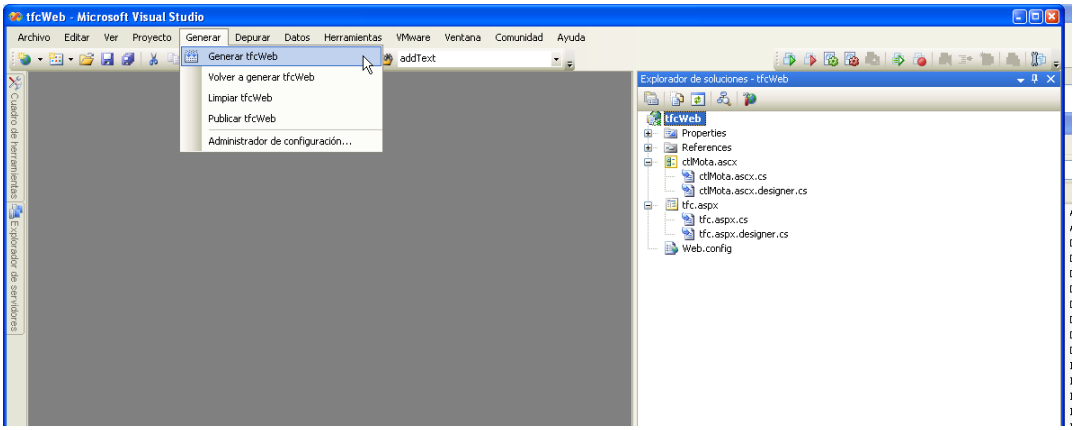
La variable **RemoteObjectServer** conté l'adreça del servidor d'objectes en el format especificat i la variable **DBconnection** l'string de connexió al servidor SQL Server que conté la base de dades del TFC.

La base de dades ha de contenir una taula, anomenada **Missatges** que respon a la següent estructura:

Nom	Tipus	Observacions
Id	Int	Identitat, clau primària
DataHora	Char(14)	Data/Hora en format YYYYMMDDHHMMSS
Origen	Int	
Desti	Int	
Tipus	Int	
Dades	Varchar(250)	

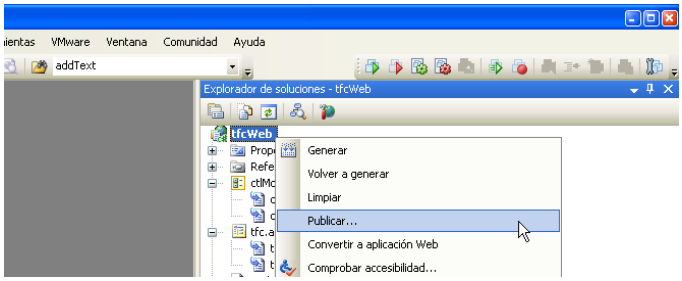
Compilació i instal·lació de la web

La compilació de la web s'inicia de forma idèntica a la del servidor d'objectes. La solució, en aquest cas, es diu tfcWeb.sln i s'ha d'obrir i generar.



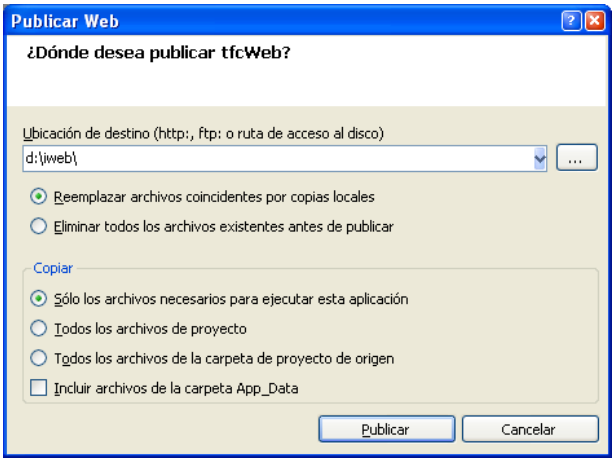
32. Detall compilació tfcWeb

A continuació s'ha de generar el conjunt de fitxers que composaran la plana web. Això s'aconsegueix amb l'opció publicar del menú contextual del projecte.



33. Detall publicació web

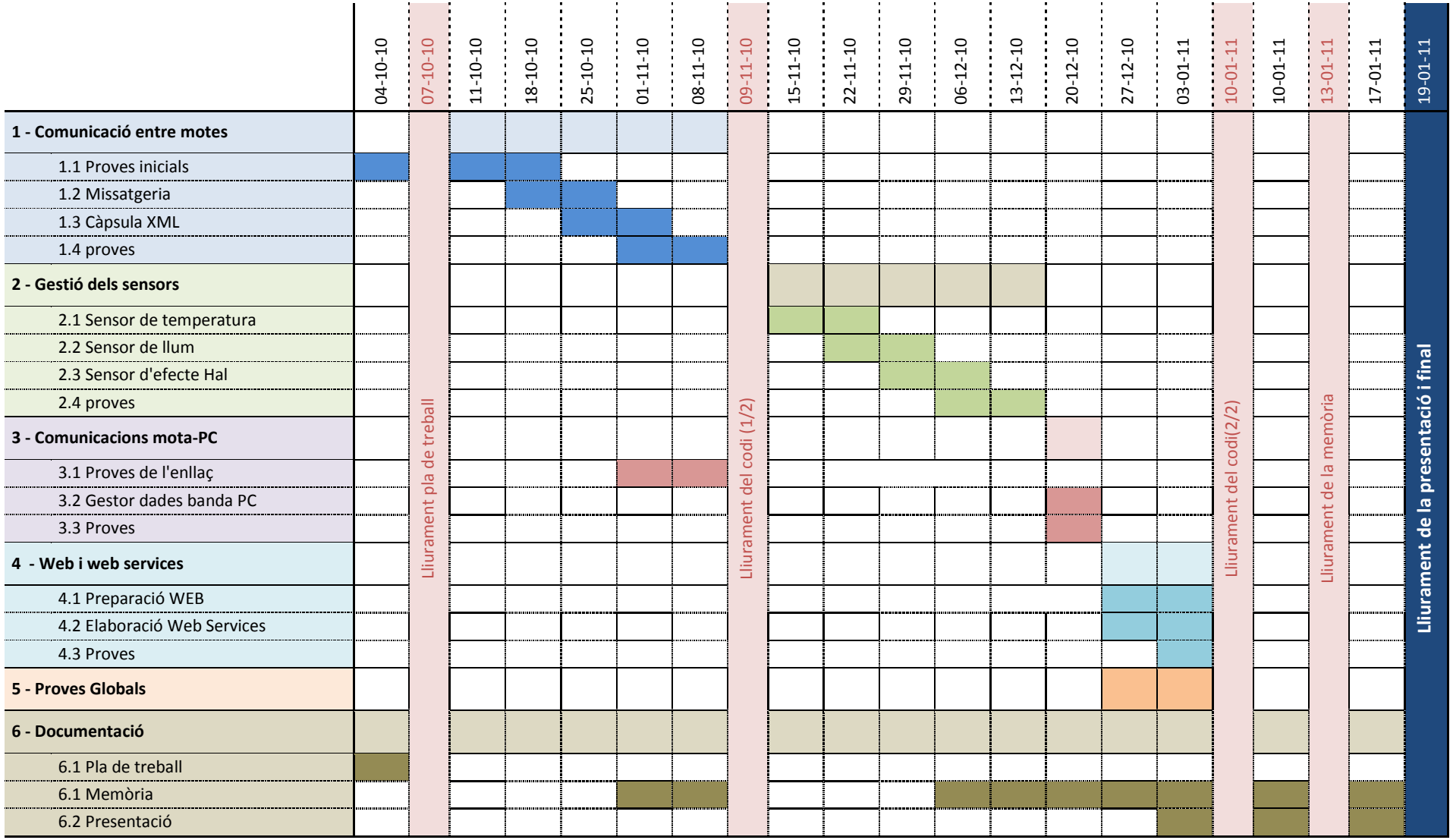
Aquesta opció obre un formulari que demana la carpeta on es copiaran els arxius i el comportament que volem que tingui el visual studio en fer-ho. Com a norma general, deixarem les opcions com es mostra a la il·lustració 34.



34. Opcions per la publicació de la web

El contingut de la carpeta indicada s'ha de traslladar a la carpeta de la plana web al IIS (Internet Information Server)

3.5 Planificació



3.6 Desenvolupament futur

El projecte desenvolupament no deixa de ser, gaire bé, una prova de laboratori. Queda molta feina per fer abans de poder considerar-lo com un projecte acabat. A continuació s'enumeren els objectius més importants a complir en un futur:

- **Proves del protocol en el seu estat actual en una xarxa molt més poblada.**
  Les proves amb només dos motes conformant la xarxa no són suficients per prendre decisions sobre la viabilitat del protocol. Cal provar situacions de molt més tràfic, amb més risc de col·lisions i major càrrega de feina en els punts on convergeixen tots els missatges: la mota base i els servidors exteriors.

- **Evolucionar el protocol cap a un encaminament multi-salt.**
  L'encaminament multi-salt permetrà dimensions de xarxa molt més àmplies que l'actual i, per tant, un àrea d'abast molt més gran.

- **Potenciar les eines d'explotació de la informació.**
  Les possibilitats d'explotació són gaire bé infinites. Amb un magatzem de dades amb prou volum es poden fer estadístiques i treballs de mineria de dades per crear models de predicció i aportar intel·ligència al sistema.

- **Inserció d'altres models de motes**
  Un dels objectius esmentats al inici d'aquest document és el de permetre la inserció de nous elements a la xarxa sense que això suposi modificar el programari dels elements preexistents.

Bibliografía

TinyOS Home Page
http://www.tinyos.net/

National Instruments Developer Zone: *Qué es una red de sensores inalámbrica (WSN)?*
http://zone.ni.com/devzone/cda/tut/p/id/9507

Berkeley, University of California: *The tinyOS help archives*
https://www.millennium.berkeley.edu/pipermail/tinyos-help

Proyecto RIMSI: *Qué es WSN?*
http://www.motas.es/rimsi/index.php?option=com_content&task=view&id=2&Itemid=3

TELEMATICA, Revista electrónica de estudios telemáticos: *Revisión del proceso de identificación de nodos en las wireless sensor networks*
http://www.urbe.edu/publicaciones/telematica/indice/html-vol8-1/articulo1.html

F.L. Lewis: *Wireless Sensor Networks*
http://arri.uta.edu/acs/networks/WirelessSensorNetChap04.pdf

Niels Aakvaag, Jan-Erik Frey: *Redes de sensores inalámbricas*
http://library.abb.com/GLOBAL/SCOT/scot271.nsf/VerityDisplay/A019E9833DCF2819C1257199004E5DD2/$File/39-42%202M631_SPA72dpi.pdf

MSDN: *Serialización de SOAP y XML*
http://msdn.microsoft.com/es-es/library/90c86ass.aspx

Archana Bharathidasan, Vijay Anand Sai Ponduru: *Sensor Networks: An Overview*
http://wwwcsif.cs.ucdavis.edu/~bharathi/sensor/survey.pdf

W3C: *Extensible Markup Language (XML) 1.0*
http://www.w3.org/TR/REC-xml/#sec-origin-goals

Antonio de la Rosa, José A. Senso: *XML como medio de normalización y desarrollo documental*
http://www.ugr.es/~jsenso/curriculum/xml.pdf