Desenvolupament d'una xarxa de sensors i actuadors sense fils de propòsit industrial

Memòria del Projecte Fi de Carrera
d'Enginyeria en Electrònica
realitzat per
Xavier López Puig
i dirigit per
Antonio Portero Trujillo
Bellaterra, 16 de Juny de 2008

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Antonio Portero Trujillo,
professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per Xavier López Puig.

I, perquè així consti, signa aquest certificat.

Signat: .....

Bellaterra (Cerdanyola del Vallès), 16 de Juny de 2008

Taula de continguts

Article.
Desenvolupament d’una Xarxa de Sensors i Actuadors Sense Fils de Propòsit Industrial

Desenvolupament d'una xarxa de sensors i actuadors sense fils de propòsit industrial

Xavier López Puig, estudiant d'Enginyeria Electrònica de l'ETSE

***Abstracte*—Aquest document és un estudi i implementació bàsica de l'estat actual en quant a les tecnologies/protocols sense fils existents, en particular ZigBee sobre la plataforma CC2430 de Texas Instruments, per a una aplicació amb una taxa de transferència de dades baixa, que no presenta un alt grau de complexitat, però que requereix gran versatilitat i fiabilitat.**

***Índex de Termes*—Wireless LAN (Xarxa d'Àrea Local Sense Fils), sistema SCADA (Control Supervisor i Adquisició de Dades) digital, baix consum, ZigBee, CC2430.**

Es recomana la consulta de l'apèndix "Conceptes previs i abreviatures" on es troben detallats conceptes i abreviatures utilitzats en el redactat d'aquesta memòria.

I. INTRODUCCIÓ

La banda de freqüència ISM (Industrial, Científica i Mèdica) de 2'4 GHz és de llicència lliure i ha estat crucial per el creixement del mercat de les tecnologies sense fils encastades, esdevenint particularment popular en els darrers anys.

La tendència actual, en qualsevol mercat (ja sigui per aplicacions industrials, electrònica de consum, automatització de la llar, gestió d'energia...) en el que es requereixi una certa interacció amb l'entorn entre les diferents parts que formen la instal·lació és de fer les comunicacions sense fils sempre que sigui possible i viable.

Així, ens trobem en un mercat que viu un dels millors moments, en constant evolució, amb multitud d'aplicacions disponibles, gran oferta d'estàndards i protocols on elegir el més adequat per la nostra aplicació, moltes plataformes de diferents fabricants per desenvolupar aquests estàndards i protocols, gran acceptació de la població, etc...

A. Aplicació

Hidràulica 2000 és una empresa especialitzada en el sector hidràulic, dedicada al subministrament i servei de components i sistemes hidràulics. Hidràulica 2000 ofereix una amplia oferta de serveis adaptats a les necessitats de cada client: instal·lació i posada en funcionament de maquinària, manteniment i reparació (bombes, motors i maquinària hidràulica en general), assistència en diagnosi, recanvis, construcció de grups hidràulics, disseny de sistemes de control, etc...

Per tant, seria interessant disposar d'un sistema de mesura i control a curta distància (uns 30 metres) que facilites les tasques de manteniment, posta a punt, diagnosi, etc... de diferents sistemes, així es tracta de desenvolupar una xarxa de sensors i actuadors sense fils que a part d'implementar la funcionalitat de lectura i control a distància també funcioni com a sistema de registre, consulta i tractament de dades ja sigui des d'una estació en planta o a través d'Internet. Per tant, l'objectiu és l'obtenció un sistema de control, supervisió i adquisició de dades (SCADA).

Aquestes serien algunes de les principals característiques / prestacions que hauria de complir el sistema:

- Flexible, que es pugui instal·lar en tot tipus d'equipament (ja sigui mòbil o estàtic), que permeti el seu redimensionament (afegir i treure sensors i actuadors) amb facilitat i sigui senzill treballar amb diferents tipus de sensors i actuadors (pressió, temperatura, caudal, revolucions, finals de carrera, alarmes, controls tot o res, controls proporcionals... tant analògics com digitals)
- Que es puguin implementar controls utilitzant PWM (Pulse-Width Modulation), que és una tècnica en la que es modifica el cicle de treball d'una senyal periòdica (ja sigui sinusoïdal o quadrada). Dita tècnica és molt utilitzada en el sector hidràulic per el control de vàlvules proporcionals, motors, etc...
- Que sigui robust, és a dir, que suporti condicions "extremes" (pols, "humitat", vibracions, altes i baixes temperatures...) i presenti immunitat en front a interferències electromagnètiques habituals en els ambients industrial
- Que puguem supervisar els valors del sensors des de qualsevol aparell electrònic al que se li pugui incorporar un receptor (laptop, PDA, telèfon mòbil...) a una distància de 30 metres aproximadament, per tant el sistema ha de permetre comunicació amb l'exterior i certa interacció a través d'una interfície home-màquina senzilla
- Seguretat en les comunicacions, encriptació de les dades
- Registre, consulta i tractament de dades des d'una estació instal·lada en planta o a través d'Internet
- Que el sistema sigui estable i de confiança, és a dir, que no es pengi, que hi hagi elements de control que verifiquin que les comunicacions i el sistema en

general està treballant correctament i si es detecta qualsevol anomalia s'activi un sistema de seguretat
- Baix consum (al ser un sistema sense fils ens interessa que presenti una gran autonomia i no requereixi gaire manteniment) i baix cost

També seria interessant que el sistema fos "modular", és a dir, que sigui fàcil de seleccionar quines d'aquestes prestacions ens interessen per a cada aplicació, per exemple hi ha aplicacions en les que potser no ens interessa fer el registre de dades o que es pugui accedir des d'Internet, i que seleccionar i descartar les prestacions requerides fos el més senzill possible.

El que s'obtindrà és un sistema de supervisió i control a distància. Un dels aspectes claus d'aquest projecte és que al ser un sistema obert permet la seva aplicació en tot tipus d'instal·lacions i amb finalitats diferents encara que en una primera instància estigui orientat a aplicacions industrials. L'objectiu de la supervisió de sistemes és el seu control, obtenir valors indicadors del seu estat. En moltes ocasions la supervisió també s'utilitza alhora de fer manteniments i tests de funcionament (ja siguin controls rutinaris o en cas de mal funcionament) i en alguns casos es fa treballar el sistema en condicions extremes per estudiar la seva resposta. Per tant, el fet de tenir un sistema de supervisió sense fils facilita que en els casos en que es faci treballar l'equipament en situacions de risc, el personal encarregat en realitzar dita prova, puguin dur a terme les seves tasques a una distància prudencial. El fet de que sigui un sistema sense fils també facilita treballar amb equipaments mòbils.

B. Objectius

Aquest projecte està basat en la transmissió de informació sense fils, al principi d'aquest projecte s'han plantejat els següents objectius:

- Fer un estudi dels estàndards i protocols sense fils existents, avaluar-los i seleccionar-ne un, el que més s'acoti a les nostres necessitats.
- Estudiar a fons l'estàndard elegit, entenent el seu funcionament i les seves prestacions i característiques pròpies
- Estudiar les tecnologies i plataformes que hi ha en el mercat per implementar l'estàndard i seleccionar la solució més adient (tenint en compte tant factors que conformen el maquinari de la plataforma com el software de desenvolupament i altres eines de suport)
- Estudiar a fons la solució elegida: funcionament, prestacions, característiques, eines de desenvolupament...
- Desenvolupar l'aplicació: definir en detall els requeriments del sistema, quins recursos necessitem, desenvolupar programari, dissenyar interfícies entre sensors/actuadors i la nostra plataforma en el cas que siguin necessàries...

II. Elecció de la tecnologia/protocol

En primera instància s'hauria d'avaluar les diferents solucions tecnològiques existents en el mercat per a comunicacions sense fils de baix consum, curta distància i baix cost (Bluetooth, ZigBee, tinyOS, Wibree...) fent una breu descripció d'algunes d'elles, les que ens han semblat més interessants i adients per als nostres requeriments.

BLUETOOTH [1]: Està orientat a la connectivitat entre laptops, PDA's...

ZIGBEE [2]: Està orientat a control i automatització.

TINYOS [3]: És un entorn operatiu basat en esdeveniments dissenyat per ser utilitzat en xarxes de sensors encastats de manera que suporta les operacions intensives concurrents que es requereixen amb els mínims requeriments de hardware. El llenguatge de programació és stylized C que utilitza un compilador a mida "NesC".

ULTRA LOW ENERGY BLUETOOTH [4]: És una tecnologia de ràdio digital dissenyada per a molt baix consum i poc rang (10 metres), basat en microchips transceptors de baix cost. Està dissenyat per tenir un consum energètic 10 vegades inferior a Bluetooth 2.1, i és 50 vegades més ràpid en les transferències de dades. L'inconvenient és que tal i com està definit no és compatible amb els estàndards actuals de Bluetooth. Algunes de les solucions que apareixeran inclouran un mòdul dual (ULE Bluetooth / Bluetooth 2.1), incloent ambdues especificacions en un mòdul, d'aquesta manera es garantirà que tant els dispositius vells com els nous puguin ser capaços de mantenir-se en contacte. No estarà disponible fins avançat el 2008.

ALTRES SOLUCIONS: No s'ha de descartar l'opció de dissenyar i realitzar les comunicacions per radio freqüència sense utilitzar cap de les tecnologies esmentades.

A. Introducció Bluetooth i ZigBee

Per elegir la tecnologia que utilitzarem per desenvolupar el projecte haurem de comparar les diferents tecnologies mencionades amb anterioritat. Ens centrarem bàsicament en comparar ZigBee i Bluetooth, descartant les altres possibles solucions ja que aquestes dues tecnologies tenen molt bones prestacions, compten amb gran suport per part de les principals companyies multinacionals, per tant se'ls augura un bon present i futur, i compten amb un perfil que s'adapta a les nostres necessitats. A més a més, ULE Buetooth encara no està disponible, i la solució de dissenyar un sistema de ràdio el descartem ja que, tal i com demostra el "White paper on decision of make vs buy of ISM RF module" [5], els costos de la inversió inicial (tant econòmics com temporals) són molt elevats i a més a més es necessiten unes expectatives de mercat molt grans per recuperar dita inversió.

Tant ZigBee com Bluetooth segueixen el protocol IEEE 802 [6] (802.15.4 [7] i 802.15.1 [8] respectivament) i operen en les bandes lliures dels 2'4 GHz.

B. Comparativa Bluetooth i ZigBee

Ambdues tecnologies són molt similars, en l'apèndix 2 "Bluetooth vs ZigBee" trobareu una breu descripció de les principals característiques dels dos estàndards, remarcant les seves especificacions exclusives.

Resumim algunes de les principals característiques d'ambdues tecnologies:

TAULA I
COMPARACIÓ BLUETOOTH I ZIGBEE

Característicques	Bluetooth	ZigBee
Consum (TX, RX i dormint)	25 mA 37 mA 30 uA	27 mA 27 mA 0'3 uA
Velocitat de transferència en RF	3 Mbps	250 Kbps
Densitat de nodes	8 (en un futur 256)	65535 (distribuits en xarxes de 255 nodes)
Distància amb antena de baix guany	100 metres	30 metres
Temps de connexió	1 s	30 ms

ZigBee i Bluetooth són molt similars però amb algunes diferències:

- Una xarxa ZigBee pot constar d'un màxim de 65535 nodes distribuïts en subxarxes de 255 nodes, enfront als 8 màxims d'una subxarxa (Piconet) Bluetooth.
- ZigBee consumeix menys que Bluetooth. En termes exactes, ZigBee té un consum de 27 mA transmetent i de 3 µA en repòs, enfront als 25 mA transmetent i 30 uA en repòs que té Blutooth. Aquest menor consum és degut a que el sistema ZigBee està la major part del temps en estat de repòs, mentre que en una comunicació Bluetooth no pot ser així, i sempre s'està transmetent i/o rebent.
- ZigBee té una velocitat de fins a 250 Kbps, mentre que en Bluetooth és de fins a 3 Mbps.
- La tecnologia ZigBee pretén ser més simple i econòmica que altres WPANs (Xarxa Sense Fils d'Àrea Personal) tal com pot ser Bluetooth. És d'esperar que el node ZigBee que més capacitat té requereixi només el 10% del software que es necessitaria per a un node de Bluetooth o d'Internet sense fils, mentre que el més simple al voltant del 2%. Com sempre, la mida dels codis ha experimentat un augment de complexitat i per tant, també de mida, tot i així la mida d'un codi ZigBee es troba al voltant del 50% de la mida per Bluetooth.
- El preu dels dispositius ZigBee és aproximadament un 30% inferior que el de Bluetooth

C. Conclusió

Per l'aplicació a la que hem de fer front és més adequat utilitzar ZigBee ja que té més autonomia, permet més connexió de dispositius, la velocitat de transferència és suficient ja que el volum de dades que hem d'enviar és petit i no hi ha requeriments de temps real, el volum de codi és inferior, és més econòmic.

ZigBee és especialment indicat per a situacions en les que el consum energètic i/o els costos d'implementació són crítics. Per això, és apropiat per aplicacions de domòtica en les que no es desitja o no és possible crear un entramat de fils de comunicació. També és propici per aquells casos en els que els elements controlats o els sensors són mòbils. Algunes de les aplicacions serien: infraestructures avançades de mesura, detectors de fum i monòxid de carboni, control de les electro-bombes i els sensors d'humitat d'un hivernacle, ZigBee és idoni, per tant és perfecte per a la nostra aplicació.

III. IEEE 802.15.4 I L'ESPECIFICACIÓ ZIGBEE

ZigBee és el nom d'una especificació per a protocols de comunicacions d'alt nivell utilitzant ràdios digitals petites i de baixa potència basat en l'estàndard IEEE 802.15.4 per a xarxes d'àrea personal sense fils (WPANs), per tant, compleix tots els requeriments de l'estàndard IEEE 802.15.4 i afegeix altres característiques. L'objectiu de ZigBee són aplicacions RF que requereixen baixa taxa de dades, llarga vida de bateria, i treball en xarxa segur.

Ressaltem les característiques ZigBee que s'adapten a les nostres necessitats:

- Permet la coexistència de diferents usuaris en freqüència
- Transmissions evitant col·lisions
- Mecanisme de reenviament de missatges

L'apèndix 3 "Especificació ZigBee" és un resum de l'estàndard IEEE 802.15.4 i l'especificació ZigBee (modulació, canals, les capes, tipus de dispositius, topologies de xarxes, procés de comunicació, encaminament, perfils, etc...).

IV. PLATAFORMA DE DESENVOLUPAMENT

Un cop seleccionat el protocol que utilitzarem fem un estudi de les plataformes disponibles al mercat que utilitzen ZigBee.

Com sempre, darrera el desenvolupament d'un estàndard com pot ser ZigBee, hi trobem el suport d'un gran nombre d'empreses del sector tecnològic que han contribuït, o no, en la seva creació i que faciliten eines per treballar-hi, tant en format hardware com software. Algunes d'aquestes empreses que disposen de plataformes de desenvolupament que compleixen les especificacions de ZigBee són: Ember, Freescales, MeshNetics, Microchip, Nec, OKI, Renesas, Texas Instruments... [2] . Dins l'apèndix 3 "Especificació ZigBee", concretament en l'apartat 3.5 trobareu els requeriments hardware mínims per poder desenvolupar aplicacions ZigBee.

De totes les plataformes disponibles en el mercat compararem la plataforma de Freescale (MC1321x [9]) i Texas Instruments (CC2430 [10] i CC2431 [11]). Hem elegit aquestes dues dins de tota l'oferta ja que es tracta de dues empreses molt fortes en el sector, per tant, de confiança, i de les que s'han utilitzat altres productes, obtenint bons resultats, i per tant coneixem les eines que utilitzen.

A. *Solució adoptada*

A l'apèndix 4 "Comparativa MC1321x (*Freescale*) vs CC2430 i CC2431 (*Texas Instruments*) trobareu una breu descripció de les característiques de cada plataforma.

La solució de *Freescale* i *Texas Instruments* són molt similars, a més a més, tant l'un com l'altre compten amb l'stack (implementació software) del protocol ZigBee, però elegim el model CC2430 de Texas Instruments ja que és la més complert i ofereix més prestacions (disposa de més memòria, és compatible amb el model que disposa de motor de localització...)

B. *Principals característiques del CC2430*

Repassem algunes de les prestacions del CC2430:

- El microcontrolador consisteix en un nucli del 8051 optimitzat amb un rendiment vuit vegades el d'un model estàndard del 8051
- Suporta depuració interactiva
- Bloqueig programable d'escriptura i lectura de la memòria flash
- Quatre timers (un de 16 bits, dos de 8 i un MAC (timer del control d'accés a memòria))
- Generació de PWM
- Quatre modes de potència (gestió de consum)
- Estàndard d'encriptació de missatges avançat AES
- Requereix missatges de resposta com a reconeixement de recepció de missatge

A l'apèndix 5 "CC2430" trobareu un resum de les principals característiques d'aquest dispositiu (GPIO, ADC, Timers, com generar senyals PWM, etc...), així com també el Z-Stack, HAL API i l'IAR IDE [18].

V. DESENVOLUPAMENT DEL PROJECTE

En aquest apartat descriurem en detall tot el procés de desenvolupament del projecte, els passos que hem seguit, des de l'adquisició del kit de desenvolupament fins als resultats i conclusions.

Consultar l'apèndix 7 "Planificació del treball i valoració econòmica" per tal de fer-se una idea de com ha estat l'evolució del treball, el temps invertit i com de rentable és treballar amb ZigBee sobre la plataforma de Texas Instruments.

A. *CC2430DK [17] vs CC2430ZDK [19]*

Texas Instruments compte amb dos kits de desenvolupament per CC2430, el CC2430DK i el CC2430ZDK. La única diferència entre els dos kits és que el CC2430ZDK està totalment orientat al desenvolupament de sistemes ZigBee i inclou més material tant hardware com software (plaques de demostració, el Z-Stack, etc...). Hem optat per el CC2430DK ja que és més econòmic (cosa important, ja que en aquests moments estem avaluant el producte) i l'stack ZigBee és de lliure distribució, per tant el podem descarregar i utilitzar-lo igualment.

B. *CC2430DK*

El CC2430DK és una eina de Texas Instruments per desenvolupar aplicacions basades en l'estàndard de xarxes 802.15.4. El kit inclou tot el hardware i el software requerit per avaluar, demostrar, prototipar i desenvolupar varies aplicacions basades en l'estàndard de xarxes 802.15.4 i ZigBee.

El CC2430DK pot ser utilitzat per desenvolupar qualsevol cosa, des de un simple interruptor de llum a nodes avançats amb molts perifèrics. El software inclou una llibreria d'interfície a hardware.

A l'apèndix 6 "CC2430DK" trobareu un resum de les principals característiques d'aquest kit (descripció del hardware inclòs en el kit, aplicacions, aspectes a tenir en compte, etc...)

C. *Consideracions prèvies respecte l'stack ZigBee*

A pesar de les bondats de l'stack ZigBee, hem de tenir molta cura alhora de dissenyar una possible aplicació i tenir en compte certes premisses.

Sabem que els ED són els que permeten un reduït consum energètic, però també sabem que aquests només es poden comunicar amb el seu coordinador i que la latència de les comunicacions és inversament proporcional al consum dels dispositius. Això és degut a que quan un ED no està transmetent o rebent, es desactiva i entra en un estat de baix consum o sleep. Només despertarà d'aquest estat cada cert temps, preprogramat, o quan una interrupció externa el desperti. Per això, tots els missatges que han de ser enviats a aquest ED han de dirigir-se al seu coordinador, que els mantindrà en una cache fins que l'ED pugui rebre'ls. Això augmenta la latència i la càrrega del coordinador, però permet a l'ED funcionar amb bateries degut al seu baix consum. Per suposat, un ED RFD no pot desenvolupar funcions d'encaminament, però ja que és possible tenir ED de tipus FFD, tècnicament si és possible tenir dispositius finals que encaminin paquets, tot i que això va en contra de la definició de "dispositiu final". De totes maneres, si el dispositiu ha d'encaminar el tràfic dels seus veïns, aquest ha de ser FFD i no pot restar en mode sleep, ja que les rutes que comptessin amb ell no serien útils.

Per tot això, si desitgem crear una xarxa extensa amb molts dispositius, hem d'elegir bé quins podran encaminar paquets i quins podran estalviar energia. Si a més a més volem fer això de manera transparent i independent del dispositiu, és a dir, que no volem preocupar-nos d'on estan els dispositius i de quin tipus són, sinó que la pròpia xarxa, per la seva estructura física, lògica i espacial ha de determinar el rol de cada dispositiu, haurem de proporcionar a cada node de la xarxa de funcionalitat complerta, de la possibilitat d'alimentar-se amb fonts "perdurables" d'energia i dels algoritmes necessaris per determinar el rol de cada node. Quelcom bastant complex.

D. *Consideracions prèvies CC2430DK*

Quan treballem amb el CC2430 sobre l'SmartRF04EB podem accedir als pins/ports I/O del CC2430 a través dels connectors A (P10) i B (P11). Hem de tenir en compte que

podem accedir des del CC2430 als perifèrics de l'SmartRF04EB (joystick, display LCD, potenciòmetre, etc.), i ho fem a través dels pins, per tant, en el cas que utilitzem algun d'aquests perifèrics no podrem utilitzar els corresponents pins. Així que s'ha de tenir present quins pins són utilitzats en cada cas. A l'apèndix 6 "CC2430DK" concretament en l'apartat 6.4.6 trobareu la taula on es descriuen ambdós connectors i quins pins són utilitzats per controlar els perifèrics de l'SmartRF04EB.

E. Aplicació inicial

Ja que aquest projecte permet moltes aplicacions possibles, degut a que té un plantejament molt obert, hem d'acotar-lo i dividir-lo en parts.

Com a primera aplicació es realitzarà el control d'una electrovàlvula proporcional. Dita vàlvula té dues sortides (A i B), però hidràulicament, tal i com està muntada en l'equipament, només és funcional la sortida A. Aquesta vàlvula és controlada mitjançant un senyal PWM de 100Hz. El control consisteix en que en funció del cicle de treball la sortida activa és una o altra, del 25% al 50% del cicle de treball s'activa la sortida A, i del 50% al 75% la sortida B.

La funció d'aquesta vàlvula és d'aixeta (com totes les vàlvules) controlant així la velocitat de gir d'un motor.

Abans de començar a programar s'han de definir certs paràmetres vitals per a l'aplicació com són els clústers, el format de les dades, etc... Que és el que farem a continuació.

F. Clústers

Els utilitzem quan enviem dades per l'aire amb la finalitat d'especificar que és el que estem enviant (configuracions, petició o resposta de dades), és la manera d'identificar quin tipus de dades s'estan transmetent. Per tant, és una de les parts en les que s'ha de tenir més cura al dissenyar, per tal d'obtenir un sistema que permeti ser manipulat, modificat, ampliat, etc.. amb facilitat.

Tot i que ens centrarem en el control mitjançant PWM també definirem clústers per treballar amb sensors (i les funcions de processament de dades associades). Els clústers sempre són ampliables, ja que només són identificadors d'informació, llavors el que faci cada dispositiu quan rep un clúster concret depèn de les funcions que associem a aquell clúster per processar-lo.

Aquests són els clústers que s'han definit:

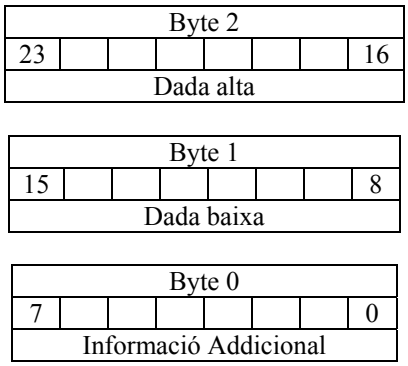
NOM DEL CLUSTER	CLUSTER ID
XSensors_request_temperatura_CLUSTERID	1
XSensors_receive_temperatura_CLUSTERID	2
XSensors_request_pressio_CLUSTERID	3
XSensors_receive_pressio_CLUSTERID	4
XSensors_request_caudal_CLUSTERID	5
XSensors_receive_caudal_CLUSTERID	6
XSensors_request_revolucions_CLUSTERID	7
XSensors_receive_revolucions_CLUSTERID	8
XSensors_request_tot_CLUSTERID	9
XSensors_send_PWM_CLUSTERID	10

Tots els clústers que contenen la paraula "request" fan referència a que és una petició d'enviament de dades, per tant, quan rebem un missatge amb aquest clúster sabem que se'ns està demanant que enviem aquelles dades (per exemple una petició de sensor de temperatura). Tots els clústers que contenen la paraula "recive" indiquen que les dades que inclou aquest missatge són la resposta a una petició realitzada (per exemple rebem el valor del sensor de temperatura). El clúster XSensors_send_PWM_CLUSTERID indica que enviem el cicle de treball d'un senyal PWM que s'està generant en el dispositiu destí, com que en aquest cas no es necessita cap resposta del dispositiu destí no necessitem cap altre clúster que faci referència a PWM .

G. Format de les dades que s'envien per aire

Aquest és un dels punts importants del projecte, ja que l'estructura de les dades serà comuna a tots els dispositius i és el mitjà utilitzat per les comunicacions conjuntament amb els clústers. Per tant, ha de permetre certa flexibilitat, ja que hem d'assegurar que les modificacions que puguem fer en un futur puguin mantenir aquest format sent compatible amb versions anteriors.

Hem definit un format de dades de 3 bytes, de manera que cada cop que s'envia un missatge s'han d'omplir els 3 bytes següents:



S'ha fet en 3 bytes perquè l''aplicació no requereix un volum de dades gran, ja que dels perifèrics que tenim (ADC, Timers, etc...) el que més capacitat té és de 16 bits, així que rarament necessitarem enviar més de 16 bits d'informació. De totes maneres, hem inclòs un altre byte. El byte d' "informació addicional" s'utilitzarà per incloure, com el seu nom indica, informació addicional (per exemple, mitjançant els clústers podem identificar de quin tipus de sensor ens arriben les dades, però en la mateixa xarxa poden haver-hi repetits sensors del mateix tipus, llavors, utilitzant aquest byte podem identificar-los, o també podem utilitzar aquest byte per fer configuracions, enviar factors de conversió, etc... i també pot ser útil en futures modificacions per fer compatible versions noves i velles), els dos bytes de dades (baix i alt) contenen els valors (dades) que es volen transmetre (tenir en compte que tot i que es tracta de 2 bytes, per exemple els convertidors AD, que tenen resolució configurable, treballen amb 14 bits com a màxim.).

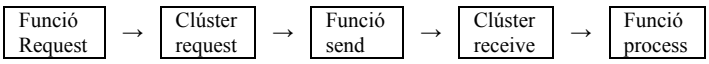
H. Comunicació

En general, les comunicacions es produiran de la següent manera; a partir d'un esdeveniment, qualsevol estímul intern o extern (ja sigui un timer, una interrupció...), un dels dispositius executa una funció request i s'envia missatge amb el clúster request. El/els dispositius receptors identifiquen el clúster, i en el cas que aquell clúster requereixi d'una resposta, s'executarà una funció send que processa el missatge ja que associem funcions a executar com a resposta als clústers que rebem. Aquestes funcions tant poden ser la lectura d'un port AD (per exemple degut a una consulta d'una sensor), o una entrada digital... Un cop processat el clúster s'enviarà un missatge resposta amb el clúster recieve incloent les dades requerides. L'emissor original rep el missatge i executa una funció process per treballar les dades rebudes (tractar-les en el cas que sigui necessari i actuant sobre sortides si es requereix).

A continuació definim els tres tipus diferents de funcions que hem dissenyat:

- Request: Fem petició perquè ens enviï les dades sol·licitades
- Send: Enviem les dades sol·licitades
- Process: Processem les dades que ens arriben

Aquest seria, d'una manera esquematitzada, el procés de comunicació:



En el cas del clúster PWM, el procés serà similar al descrit anteriorment però amb petites variacions. Un dispositiu ZigBee, degut a l'execució d'una funció (com a resposta al valor d'algun dels sensors) o per una interrupció, executarà la funció PWM, que enviarà un missatge amb el clúster PWM que inclourà el valor de cicle de treball (aquest valor, tant pot ser calculat com a resultat d'un algorisme com ser la representació de la lectura d'una entrada analògica). El/els dispositius receptors reconeixeran el clúster i processaran el missatge executant una funció process, variant el cicle de treball del senyal PWM que estaran generant sense retornar cap missatge resposta al dispositiu origen.

Aquest és el tràfic de dades que inclou la informació que nosaltres generem, però s'ha de tenir en compte que cada cop que enviem un missatge també s'envien altres camps (adreça destí, identificador de clúster, etc...). Trobareu més informació a l'apèndix 8 "Premisses, programació i algorismes comentats"

I. Seguretat

S'ha dotat de varis sistemes de seguretat al sistema:

- Cada cop que enviem un missatge se'ns notifica si ha pogut ser enviat o no, és a dir, si el nostre dispositiu ha estat capaç d'enviar-lo o no, en el cas que no s'hagi enviat s'intenta reenviar fins a 5 vegades, si no ha estat possible l'enviament, es notifica a l'usuari
- Per cada missatge que enviem esperem un missatge de resposta de reconeixement. Si no el rebem reenviem el missatge fins a 5 vegades, en el cas en que seguim sense rebre resposta mostrem missatge d'alarma
- Quan treballem en mode control electrovàlvula els dispositius implicats no poden entrar en estat d'estalvi d'energia, ja que el control ha de ser exhaustiu

Per més informació sobre la implementació dels modes de seguretat consultar l'apèndix 8 "Premisses, programació i algorismes comentats"

J. Amplificació

Per poder actuar sobre la vàlvula necessitem amplificar el senyal PWM, per el que s'ha optat per adquirir un mòdul amplificador industrial Iso2 Flex [20].

K. Resolució

A l'apèndix "Premisses, programació i algorismes comentats" trobareu el codi per aquesta aplicació.

El codi escrit genera quatre modes d'operació diferents. Bàsicament, el que es fa és que un dispositiu s'encarrega de generar un senyal PWM i un altre dispositiu s'encarrega d'enviar-li les variacions que s'han d'efectuar sobre el cicle de treball. També s'ha implementat una simulació de lectura de sensor, en la que es fa petició de lectura i es rep la lectura d'un port analògic. Els modes de treball es seleccionen a través del joystick que incorpora com a perifèric l'SmartRF04EB.

Aquests són els modes de treball:

- Amunt: Cada cop que premem el joystick en direcció ascendent incrementem el cicle de treball del PWM i l'enviem
- Avall: Cada cop que premem el joystick en direcció descendent disminuïm el cicle de treball del PWM i l'enviem.
- Dreta: Cada cop que premem el joystick en direcció dreta realitzem una petició de lectura del "sensor de pressió". Com que no tenim sensors, el dispositiu que rep aquesta petició realitza una lectura d'un port analògic on podem tenir-hi qualsevol valor.
- Esquerra: Quan premem el joystick en direcció esquerra executem un esdeveniment per timer cada 0'3 segons que provoca una lectura d'un port analògic on tenim connectat un potenciòmetre. Es captura el valor del potenciòmetre i s'envia com a cicle de treball d'un senyal PWM.

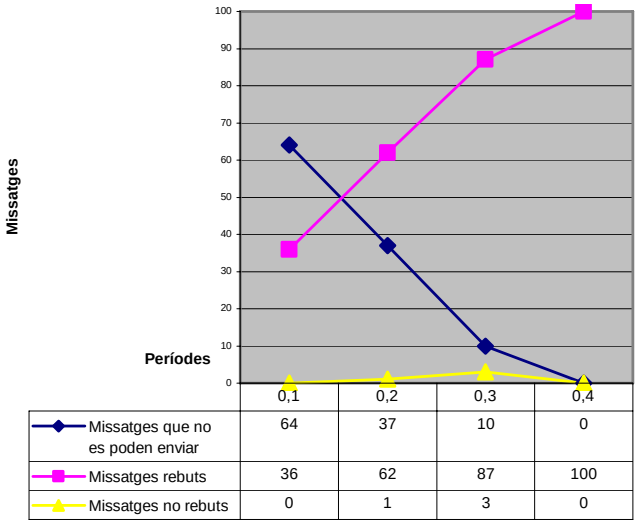
L. Resultats

S'ha realitzat l'aplicació descrita, aquesta funciona correctament, un dispositiu genera el senyal PWM, i l'altre controla el seu cicle de treball.

A part, s'ha realitzat una prova de test per tal de valorar la qualitat de la comunicació. Aquesta prova consisteix en comparar la distància entre dispositius i l'èxit en l'enviament/recepció de missatges. Hem agafat l'aplicació que hem desenvolupat en l'apèndix 8 "Premisses, programació i algorismes comentats" en el mode de treball d'enviament de cicle de treball com a resultat d'un esdeveniment provocat per un timer, i hem variat el període de l'esdeveniment i la distància entre dispositius. La prova consisteix en enviar 100

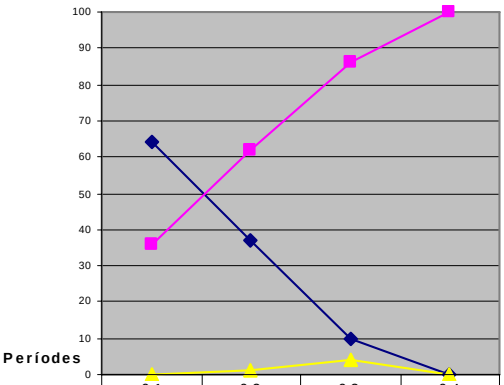
missatges de cicle de treball (és a dir, lectura del port ADC, enviament de missatge, recepció de missatge, processar missatge i modificar el cicle de treball del PWM) i fer un enregistrament dels paquets que no s'han pogut enviar i els paquets que hem rebut. Els resultats els trobareu en l'apèndix 9 "Test de velocitat de transferència". Com a resultat d'aquesta experiència hem elegit el període de 0'3 segons com a òptim, ja que mostra una bona relació entre els missatges que es volen enviar, els que s'aconsegueixen enviar, els que es reben, i el temps entre missatges és més que suficient. A continuació es mostren dos gràfics, un amb la relació de missatges de 0 a 20 metres, i l'altre de 25 a 45 metres. No s'han fet transmissions més enllà dels 45 metres perquè no estava dins del plantejament inicial del projecte, ja que per la maquinaria a la que va orientat rarament ens meurem més enllà d'una superfície de 30 x 30 metres.

De 0 a 20 metres



	0,1	0,2	0,3	0,4
Missatges que no es poden enviar	64	37	10	0
Missatges rebuts	36	62	87	100
Missatges no rebuts	0	1	3	0

De 25 a 45 metres



	0,1	0,2	0,3	0,4
Missatges que no es poden enviar	64	37	10	0
Missatges rebuts	36	62	86	100
Missatges no rebuts	0	1	4	0

M. Conclusions

El projecte s'ha finalitzat amb èxit assolint les expectatives inicials. A continuació un llistat de conclusions resultants del desenvolupament del projecte:

- S'ha aconseguit realitzar l'aplicació, i funciona correctament
- S'ha contestat, que tot i que hi ha la possibilitat programar directament sobre els registres, el fet de poder utilitzar eines de suport com el Z-Stack API, Z-Stack OSAL API i el HAL API faciliten molt les coses alhora de voler implementar aplicacions ZigBee i utilitzar els perifèrics del dispositiu.
- Com s'observa en l'apèndix 9 "Test de velocitat de transferència", la taxa de transferència queda limitada per la capacitat que té el CC2430 en processar els missatges que s'han d'enviar, no pas per la capacitat que té per rebre missatges ni per la distància. Així ens trobem que amb un període de 0'1 segons el 64% dels missatges no es poden enviar, en canvi es reben el 100 % dels missatges que si s'aconsegueixen enviar. Tal com s'observa en els gràfics de l'apartat anterior, a mesura que augmentem el període es compensa la quantitat de missatges que no es poden enviar, amb els que si que s'envien, els que es reben i els que es perden. En referència a la distància de separació entre dispositius, cal dir que fins als 40 metres de separació no hi ha hagut cap variació en les taxes de missatges, en excepció del període de 0'3 segons (que a partir dels 25 metres es perd un paquet més que en distàncies inferiors). El que si que hem detectat és que al treballar amb 45 metres de distància i sense visibilitat (ja que un dels dispositius es trobava situat fora de les instal·lacions) en algun moment costava que els dispositius es detectessin,

però un cop detectats es mantenen les taxes de comunicació. Veient els resultats, s'ha d'elegir un període de transferència que permeti un bon resultat en quant a la relació de missatges que s'intenten enviar, els que realment s'envien, els que es reben i que, encara que no hagi de ser en temps real, tampoc s'hagi d'agafar un període excessivament llarg. Així que veient el que hem obtingut en l'experiència realitzada optem per elegir com a període d'enviament de missatges de correcció del cicle de treball del PWM 0'3 segons (s'aconsegueixen enviar el 90% dels missatges, i dels que s'envien es reben el 95'5%).

- Tant el temps invertit com el cost material (consultar apèndix 7 "Planificació del treball i valoració econòmica) no són cap exageració, a més, si tenim en compte que els coneixements sobre la matèria en un inici eren zero. A part, la plataforma de desenvolupament presenta gran facilitat en la manipulació del codi, tant per fer petites modificacions com per generar projectes totalment nous, és molt versàtil i permet crear implementacions ZigBee en un període de temps curt.

Tot això fa que el projecte hagi estat totalment viable i es pugui tenir un preu de mercat competitiu.

El dia de la presentació es mostraran dos vídeos. En un es podrà veure com funciona el control de l'electrovàlvula i l'altre mostrarà l'espai on s'han realitzat les proves descrites en l'apèndix 9 "Test de velocitat de transferència".

N. Futur

Han quedat moltes coses per fer, aquest llistat mostra les ampliacions ha fer:

- Implementar la xarxa de sensors i actuadors. S'han fet els clústers (per els sensors), ara falta definir els actuadors, els sensors, crear interfícies (en el cas que faci falta, com per exemple l'amplificador per el control PWM), dissenyar les funcions que es requereixin (depenent de l'aplicació) i coordinar-ho tot
- Un cop tinguem la xarxa de sensors i actuadors funcionant s'hauria de fer tot el referent al enregistrament de les dades, processat i consulta
- Ampliar la seguretat aplicant un control del nivell de les bateries. En el cas que aquest nivell sigui inferior a cert llindar avisar al coordinador
- Ampliar seguretat enviant missatges periòdics per saber quins dispositius de la xarxa estan actius
- Utilitzar el perifèric que implementa estàndard d'encriptació AES [13]

REFERÈNCIES

[1] Especificació Bluetooth disponible a través de la web: http://www.bluetooth.com

[2] Especificació ZigBee disponible a través de la web: http://www.zigbee.org

[3] Especificació Tinyos disponible a través de la web: http://www.tinyos.net

[4] Especificació Ultra Low Energy Bluetooth disponible a través de la web: http:// www.bluetooth.com/bluetooth/Products/lowpower.htm

[5] White paper on decision of make vs buy of ISM RF module disponible a través de la web: http://focus.it.com.cn/cn/lit/wp/sliy002/sliy002.pdf

[6] Més informació sobre l'estàndard IEEE 802 disponible a través de la web: http://www.ieee802.org

[7] Més informació sobre l'estàndard IEEE 802.15.4 disponible a través de la web: http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

[8] Més informació sobre l'estàndard IEEE 802.15.1 disponible a través de la web: http://www.ieee802.org/15/pub/TG1.html

[9] Més informació sobre el MC1321x i tots els documents tècnics disponibles a través de la web: http://www.freescale.com/files/rf_if/doc/data_sheet/MC1321x.pdf

[10] Més informació sobre el CC2430 i tots els documents tècnics disponibles a través de la web: http://focus.ti.com/docs/prod/folders/print/cc2430.html

[11] Més informació sobre el CC2431 i tots els documents tècnics disponibles a través de la web: http://focus.ti.com/docs/prod/folders/print/cc2431.html

[12] Més informació sobre el HCS08 i tots els documents tècnics disponibles a través de la web: http://www.freescale.com/webapp/sps/site/taxonomy.jsp?nodeId=01624684491437

[13] Més informació sobre l'AES disponible a través de la web: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[14] Més informació sobre el CC2420 i tots els documents tècnics disponibles a través de la web: http://focus.ti.com/docs/prod/folders/print/cc2420.html

[15] Més informació sobre el MCU 8051 disponible a través de la web: http://en.wikipedia.org/wiki/Intel_8051

[16] Més informació sobre el Z-Stack i tots els documents tècnics disponibles a través de la web: http://focus.it.com/docs/toolsw/folders/print/z-stack.html

[17] Més informació sobre el CC2430DK i tots els documents tècnics disponibles a través de la web: http://focus.ti.com/docs/toolsw/folders/print/cc2430dk.html

[18] Més informació i manual d'instruccions de l'IAR IDE a través de la web: http://www.iar.com

[19] Més informació sobre el CC2430ZDK i tots els documents tècnics disponibles a través de la web: http://focus.ti.com/docs/toolsw/folders/print/cc2430zdk.html

[20] Més informació sobre el mòdul Iso2 Flex disponible a: http://www.guemisa.com/conver/docus/Iso2%20flexweb.pdf

Apèndix 1. Conceptes previs i abreviatures

- ADC: Convertidor de senyal analògic a digital
- ADREÇA DE XARXA: L'adreça assignada a un dispositiu per el per la capa de xarxa i utilitzada per la capa de xarxa per encaminar missatges entre dispositius.
- ADREÇAMENT INDIRECTE: L'habilitat d'alguns dispositius per comunicar-se sense saber l'adreça del destí desitjat. Les transmissions indirectes només hauran d'incloure el camp de l'adreça del punt final font amb el bit de Adreçament Indirecte activant el APDU i seran direccionats cap el Coordinador ZigBee per la font. El que s'espera que passi és que el Coordinador ZigBee busqui l'adreça / punt final / clúster ID de la font dins de la seva taula de vincles i reenviï el missatge a cada adreça / punt final destí corresponent.
- AES: Estàndard d'encriptació avançat
- AF: Framework d'aplicació
- API: Aplicació d'interfície de programació
- APL: Capa d'aplicació
- APS: Sub-capa de suport a aplicació
- ASSOCIACIÓ: El servei subministrat per la sub-capa MAC IEEE 802.15.4-2003 que és utilitzat per establir relacions d'agrupament en una xarxa
- ATRIBUT: Una entitat de dada que representa una quantitat física o estat. Aquesta dada és comunicada a altres dispositius utilitzant comandes.
- BEACON-ENABLED PAN: Una Personal Area Network que conté com a mínim un dispositiu que transmet beacon frames a un interval regular.
- BPSK: Binary phase-shift keying
- BROADCAST: La transmissió d'un missatge a tots els dispositius dins d'una xarxa.
- BOD: Brown out detector, protegeix els continguts de la memòria durant les variacions de la tensió d'alimentació que provoquen que els voltatges regulats d'1'8V caiguin per sota del nivell mínim requerit per la memòria flash i la SRAM.
- CLÚSTER: Un contenidor per a un o més atributs
- COORDINADOR ZIGBEE: El controlador principal d'una xarxa basada en IEEE 802.15.4-2003 que és responsable de la formació i el manteniment de la xarxa (associar i desassociar dispositius...). El coordinador ZigBee ha de ser un full function device (FFD).
- CRC: Cyclic Redundancy Check
- CSMA-CA: Accés múltiple per detecció de portadora amb evasió de col·lisió

- DESASSOCIACIÓ: El servei subministrat per la sub-capa MAC IEEE 802.15.4-2003 que és utilitzat per discontinuar la relació en un grup d'un dispositiu en una xarxa.
- DISPOSITIU: Qualsevol entitat que conté una implementació de la pila del protocol ZigBee.
- DISPOSITIU FINAL ZIGBEE: Un RFD o FFD IEEE 802.15.4-2003 participant en una xarxa ZigBee, el qual no és ni un coordinador ZigBee ni un router ZigBee.
- DMA: Accés directe a memòria
- DSSS: Direct-sequence spread spectrum
- FULL-FUNCTION DEVICE (FFD): Un dispositiu amb capacitat d'operar com a coordinador o router i implementa el conjunt complert del protocol.
- HAL: Capa d'abstracció hardware
- IDE: Ambient de desenvolupament integrat
- IDENTIFICADOR DE CLÚSTER: Una referència a l'única enumeració de clústers dins d'un perfil específic. L'identificador de clúster és un únic número de 8 bits dins de l'àmbit del segment de camp de l'aplicació i identifica un clúster específic. Els identificadors de clúster són identificats com a entrades o sortides en el descriptor simple per utilitzar en la creació d'una "binding table" (taula de vinclament)
- IF: Freqüència intermitja
- LNA: Amplificador de baix soroll
- MAC: Control d'accés al medi
- MCU: Unitat microcontroladora
- MISO: Master in slave out
- MOSI: Master out slave in
- NODE: Una col·lecció de descripcions i aplicacions d'un dispositiu independent residint en una sola unitat i compartint en comú una ràdio 802.15.4.
- NWK: Network, xarxa
- OSAL: Capa d'abstracció de sistema operatiu
- PHY: Capa física
- PUNT FINAL: Un component particular dins d'una unitat. Cada dispositiu ZigBee pot suportar fins a 240 d'aquests components.
- PERFIL: Una col·lecció de descripcions de dispositiu, les quals totes juntes formen una aplicació cooperativa. Per exemple, un termòstat en un node es comunica amb una caldera en un altre node. Tots junts, formen de manera cooperativa un perfil d'aplicació de calefacció.
- POR: Power on reset. Proporciona una correcta inicialització durant la posta en marxa del dispositiu.
- REDUCED FUNCTION DEVICE (RFD): Un dispositiu operant amb una implementació mínima del protocol IEEE 802.15.4.
- RF: Ràdio freqüència
- ROUTER ZIGBEE: Un FFD IEEE 802.15.4-2003 participant en una xarxa ZigBee, el qual no és un coordinador ZigBee però pot actuar com un coordinador IEEE 802.15.4-2003 dins del seu propi espai d'operació, que té capacitat d'encaminar missatges entre dispositius i donar suport.
- QPSK: Quadrature phase-shift keying
- SFR: Registre de funció especial

- TRANSMISSIÓ DIRECTE: Transmissió de frame utilitzant adreçament directe.
- UNICAST: La transmissió d'un missatge a un sol dispositiu en una xarxa.
- USART: Receptor/transmissor síncron/asíncron universal
- UART: Receptor/transmissor asíncron universal
- XARXA MESH: Una xarxa en la qual l'encaminament de missatges es realitza com un procés descentralitzat i cooperatiu que inclou varis encaminaments de parells de dispositius de part d'altres dispositius.
- ZIGBEE DEVICE OBJECT (ZDO): La porció de la capa d'aplicació responsable de definir el rol del dispositiu dins d'una xarxa (per exemple, coordinador ZigBee o dispositiu final), iniciant i/o responent a consultes de vinculació i descobriment i establint relacions segures entre dispositius de xarxa.

Apèndix 2. Bluetooth vs. ZigBee

En aquest apartat realitzarem una breu descripció d'algunes de les característiques de Bluetooth i de ZigBee que ens serviran alhora de seleccionar quin dels dos estàndards seguir. Per a més informació sobre l'especificació Bluetooth i ZigBee consultar les referències [1] i [2] respectivament.

2.1 Bluetooth

Es tracta d'un estàndard sense fils basat en l'estàndard IEEE 802.15.1 utilitzat per connectar entre si telèfons mòbils, ordinadors portàtils, mans lliures, reproductors de MP3... Existeixen quatre versions d'aquest estàndard, que segueix en constat desenvolupament, reforçant els seus punts forts: baix consum, mínim cost, seguretat integrada, fiabilitat, facilitat d'us i possibilitat de connexió en xarxa ad hoc (xarxa en la que no existeix un node central si no que tots els nodes estan en igual de condicions, per tant no es necessita una estructura fixa).

Resum de característiques:

- La tecnologia Bluetooth està orientada a aplicacions de veu i dades
- Funciona a la banda de freqüència de 2'4 GHz la qual no requereix cap tipus de llicència
- El radi d'acció és de 10 a 100 metres en funció de la classe del dispositiu Bluetooth. La màxima velocitat de transmissió és de 3 Mbps.
- Els objectes sòlids no suposen cap tipus d'obstacle
- No és necessari que els dispositius estiguin en la mateixa línea de visió ja que transmeten en totes direccions
- Existeixen modes de seguretat

2.2 ZigBee

Es tracta d'un estàndard sense fils basat en l'estàndard IEEE 802.15.4 utilitzat per aplicacions de control i xarxes de sensors, especialment on es requereix baixa transmissió de dades, llarga vida de bateria i seguretat.

Resum de característiques:

- El radi d'acció és de 10 a 100 metres
- La velocitat de transmissió és de 250 Kbits en la banda de 2'4 GHz, 40 Kbps en la banda de 915 MHz i 20 Kbps en la banda de 868 MHz
- L'objectiu d'aquesta tecnologia és convertir-se en un estàndard sense fils en el sector del control remot
- Es centra en aquelles aplicacions de control en les que no es requereix una velocitat de transmissió molt alta però que requereixen un consum i un preu baix, així com facilitat d'us, com controls remots, domòtica...
- Ofereix nivells de seguretat
- Màxim de 240 end-points per dispositiu ZigBee

Apèndix 3. Especificació ZigBee

En aquest apartat es realitzarà un resum del que és l'especificació ZigBee, incloent l'estàndard IEEE 802.15.4. Per a una informació més detallada, consultar les referències [7] (tot sobre IEEE 802.15.4) i [2] (tot sobre l'aliança ZigBee)

3.1 Introducció

ZigBee és un estàndard de comunicacions sense fils dissenyada per la ZigBee Alliance. No és una tecnologia, si no un conjunt estandarditzat de solucions que poden ser implementades per qualsevol que ho desitgi.

A l'actualitat existeix una gran quantitat d'estàndards per a comunicacions sense fils. Permeten grans taxes de transferència per a aplicacions tals com la transmissió d'àudio, vídeo, dades, etc... Malgrat tot, aquests estàndards no són adequats per situacions en les que el consum energètic o la complexitat del dispositiu són vitals. Per això s'ha dissenyat ZigBee.

Tant els sensors com els actuadors o altres dispositius petits de mesura o control no requereixen un gran ampla de banda, però si un mínim consum energètic i una baixa latència. ZigBee va ser creat per la comunicació d'aquests dispositius.

3.1.1 Què és ZigBee?

ZigBee és el nom d'una especificació per a protocols de comunicacions d'alt nivell utilitzant ràdios digitals petites i de baixa potència basat en l'estàndard IEEE 802.15.4 per a xarxes d'àrea personal sense fils (WPANs). L'objectiu de ZigBee són aplicacions RF que requereixen baixa taxa de dades, llarga vida de bateria, i treball en xarxa segur.

L'especificació ZigBee 1.0 va ser editada el 14 de Desembre de 2004, la última versió, ZigBee 2006, va ser editada durant el Desembre del 2006.

ZigBee opera en les bandes de ràdio industrial, científica i mèdica (ISM); 868 MHz a Europa, 915 MHz a USA i 2'4 GHz a la major part del món.

La última versió de l'stack (la segona), editada el 26 de Juny de 2007, anomenada 2006, principalment substitueix l'estructura MSG/KVP per una llibreria "clúster". L'aliança ZigBee ha començat a treballar en el que serà la versió 2007.

També hi ha nous perfils d'aplicacions com la lectura automàtica de mesures, automatització d'edificis comercials i automatització de cases basat en el "clúster library principe".

A nivell de xarxa ZigBee 2007 no és compatible amb ZigBee 2004/2006, tot i que el node ZigBee 2004/2006 RFD pot adherir-se a una xarxa del 2007 i visa-versa. El que no és possible és barrejar routers 2004/2006 amb routers/coordinadors del 2007.

Així, l'aliança ZigBee està desenvolupant un estàndard de comunicacions sense fils a molt baix cost, molt baix consum de potència i de dos vies. Les solucions que adoptin l'estàndard ZigBee seran encastats en electrònica de consum, automatització de casa i edifici, control industrial, perifèrics PC, aplicacions de sensors mèdics, joguines...

L'actual focus de ZigBee és definir una xarxa mesh de propòsit general, econòmica, auto-organitzada, que pugui ser utilitzada per control industrial, sensors encastats, recol·lecció de dades mèdiques, avisos d'intrusió i fum, automatització d'edificis, de cases, domòtica... La xarxa resultant utilitzarà molt poca potència de manera que els dispositius individuals poden funcionar durant anys sense canviar les bateries.

3.2 ZigBee i la coexistència de freqüència sense fils

La banda de freqüència ISM (Industrial, Científica i Mèdica) és de llicència lliure i ha estat crucial per el creixement del mercat de les tecnologies sense fils encastades però, com a recurs compartit, és crucial que tots els usuaris de la banda actuïn com a "bons ciutadans". En particular, els dissenyadors i implementadors de plataformes i productes han d'assumir que, en un cas normal, compartiran el medi RF amb una varietat d'altres emissors, ja siguin intencionadament o no.

La banda ISM de 2'4 GHz ha esdevingut particularment popular en els últims anys. Una petita llista dels possibles usuaris i possibles interferències inclou:

- Xarxes 802.11b
- Xarxes 802.11g
- Xarxes 802.11n
- Bluetooth Pico-Nets
- PAN basada en 802.15.4
- Telèfons sense fils
- Càmeres de vigilància
- Forns microones
- Auriculars sense fils
- Xarxes WiMax

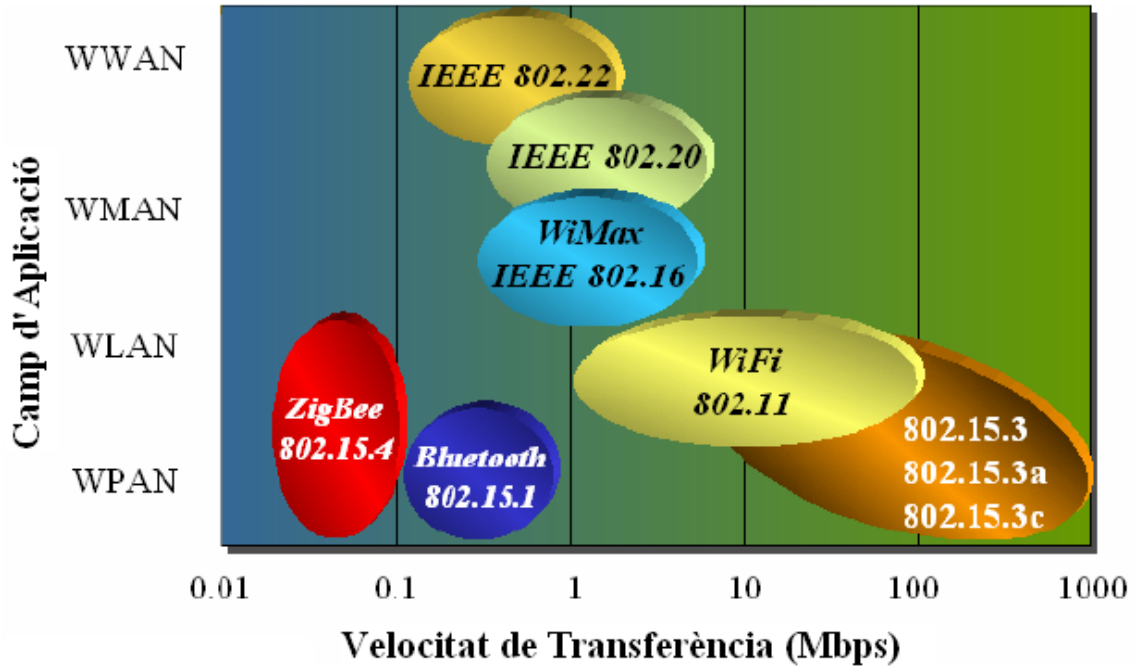


Figura 3.2.1 Usuaris de la banda ISM

Amb tants usuaris és lògic pensar que treballar en una banda tant plena com la de 2'4 GHz podria ser un problema. La millor manera d'enfrontar-se a aquesta situació és dissenyar el sistema tenint en ment que hi haurà interferències.

3.2.1 Coexistència en ZigBee

La pila (stack) ZigBee té quatre capes, dues descrites per l'especificació ZigBee (la capa d'aplicació APL i la de xarxa NWK) i dues descrites en l'estàndard IEEE 802.15.4-2003 (la capa de control d'accés al medi MAC, i la capa física PHY).

Les dues especificacions contenen part de la funcionalitat especialment dissenyada a promoure la coexistència i la mitigació d'interferències, i aquesta funcionalitat es distribueix a través de les quatre capes.

3.2.2 IEEE 802.15.4

La política del IEEE requereix que cada comitè d'estàndards que segueixi l'IEEE publiqui una declaració de coexistència d'acord amb el propi estàndard. Per tant, l'especificació IEEE 802.15.4-2003 proporciona suport a la coexistència en les dues capes, la PHY i la MAC, començant per la capa física (encarregada de l'emissora de ràdio) adoptant la tecnologia de seqüència directe d'espectre estès (direct sequence spread spectrum, DSSS).

3.2.2.1 DSSS

El terme “espectre estès” fa referència a la classe de tecnologies, les quals són dissenyades per promoure la coexistència i la duresa enfront de les interferències. Hi ha un acord entre la comunitat d’estàndards que proposa l’ús de l’espectre estès ja que és crucial alhora de compartir l’espectre ISM. Per il·lustrar aquest situació, a la figura de sota es mostra la col·lisió entre dos senyals de banda estreta (senyals que utilitzen una banda petita de freqüències al voltant de la seva freqüència central o canal).

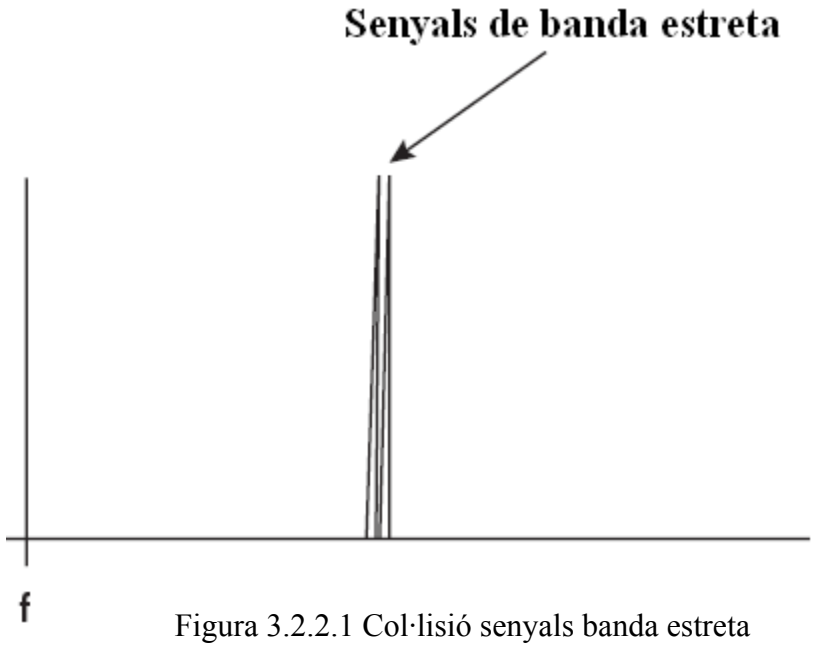


Figura 3.2.2.1 Col·lisió senyals banda estreta

Encara que la freqüència central d’aquests senyals no és exactament la mateixa, el solapament és substancial i és probable que causi pèrdues de dades. Aquesta ha estat la funció dels cossos reguladors com pot ser el U. S. Federal Communications Comission de prevenir exactament aquest tipus de col·lisions entre senyals de banda estreta mitjançant regulacions estrictes de quins emissors poden operar en quins canals d’una banda en particular, i en quines regions geogràfiques. Aquesta protecció no està disponible en les bandes ISM i per tant els usuaris de tecnologia de banda estreta corren el risc de trobar-se amb aquests tipus de col·lisions.

Contrastant amb la següent figura que mostra la col·lisió entre un senyal estès amb un de banda estreta. Hi ha varis mètodes per estendre senyals d’ús comú, però l’idea essencial darrera de tots ells és utilitzar un ample de banda que pot ser varis ordres de magnitud superior del que seria estrictament requerit per la informació que està sent enviada. Degut a que el senyal està estès sobre un ample de banda llarg, pot coexistir amb senyals de banda estreta, els quals generalment apareixen al receptor d’espectre estès com a una lleugera reducció en la relació senyal-soroll sobre l’espectre utilitzat. Les tecnologies d’espectre estès també proporcionen múltiple accés a un únic canal.

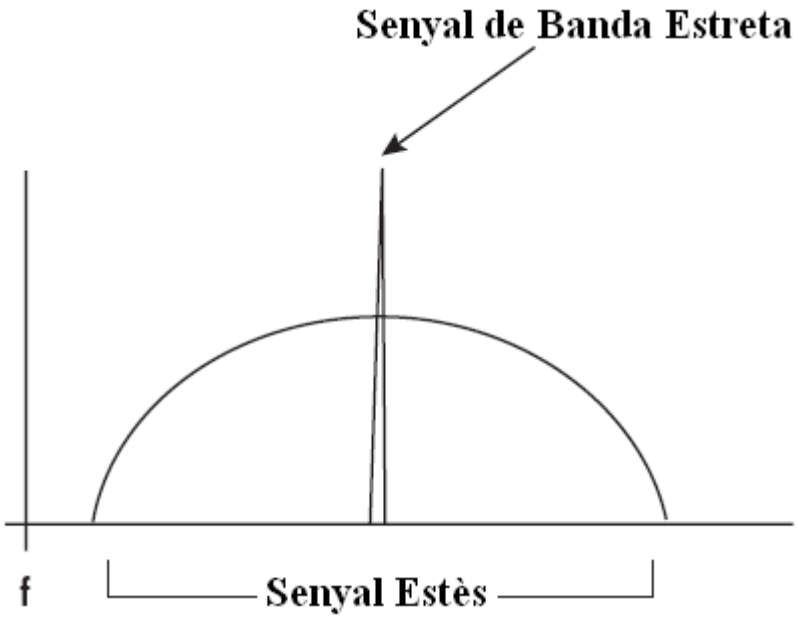


Figura 3.2.2.2 Col·lisió senyals banda estreta i senyal estès

3.2.2.2 Múltiples canals

A més a més del DSS, el 802.15.4 augmenta les oportunitats de coexistència utilitzant una tècnica generalment coneguda com accés multiple per divisió de freqüència (FDMA). Senzillament significa que l'estàndard divideix la banda ISM de 2'4 GHz en 16 canals sense solapament, els quals estan separats per 5 MHz com es mostra a la següent figura.

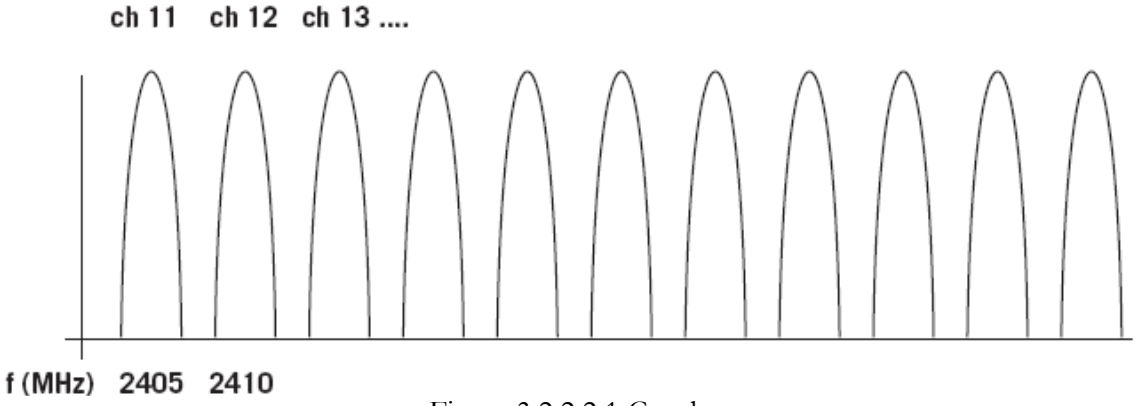


Figura 3.2.2.2.1 Canals

Com a mínim dos d'aquests canals, específicament el 15 i el 20, cauen entre els canals sense solapament 1, 6 i 11 del 802.11. La mínima resistència requerida als senyals de dispositius 802.15.4 operant en altres canals, anomenat "jamming resistance", per una ràdio que compleix l'estàndard es mostra en la següent taula

Mínim "Jamming Resistance"	
Rebuig a Canal Adjacent	Alternació de Canal
0 dB	30 dB

Com a resultat d'aquests requeriments de "jamming resistance", els dispositius que compleixin i operin en canals adjacents poden coexistir i els dispositius que operin en canals separats més ampliament basicament no se sentiran els uns als altres.

3.2.2.3 Taxa de dades

Moltes de les aplicacions per dispositius ZigBee requereixen una taxa de dades molt baixa. Molts dissenyadors de sistemes RF han explotat aquest factor construint transmissors amb taxa de dades tant baixa com 9'6 Kbps. Els dissenyadors de la capa física IEEE 802.15.4, han elegit una taxa de dades relativament alta de 250Kbps. El raonament és que una de les millors maneres de promoure la coexistència és reduir l'ocupació del canal. Clarament, una ràdio amb una taxa de dades alta ocuparà el canal molt menys i oferirà menys oportunitats de col·lisió amb altres usuaris que un amb una taxa de dades més baixa.

3.2.2.4 Construït per escanejar i informar

Per tal de poder explotar plenament els múltiples canals sota 802.15.4, la interfície amb la capa PHY proporciona l'habilitat de mostrejar un canal i informar si el canal està lliure per transmetre. També mesura l'energia, i així l'interferència, present en un canal. La resta de capacitat es troba en la capa MAC i altres superiors en les que els usuaris de ràdios 802.15.4 tenen l'habilitat de sel·leccionar el millor canal disponible per operar.

3.2.2.5 CSMA

Tot i el que hem descrit anteriorment, un dispositiu ZigBee pot estar transmetent i trobar-se interferències produint-se col·lisions, aquesta situació ha estat contemplada. Hi ha varies maneres d'afrontar aquesta problemàtica però la solució adoptada per l'IEEE per a l'estàndard 802.15.4 és una coneguda com múltiple accés per detecció de portadora (CSMA). Aquesta tècnica té l'avantatge que no requereix sincronització entre els dispositius, es basa en una estrategia tant simple com "escoltar abans de parlar". Si el canal està ocupat espera una estona abans de tornar a comprovar l'estat del canal, això es repetèix un cert nombre de cops, si el canal segueix ocupat es deixa.

3.2.2.6 Transmissió de reconeixement i reenviament

Sovint al transmetre missatges, el receptor no el rep bé. Totes les comunicacions en 802.15.4 s'han de reconèixer (acknoledge, ACK). Al rebrer un missatge hi ha un cert temps durant el qual el receptor ha de respondre amb un missatge de reconeixement. L'emissor espera l'arribada de l'ACK, si no el rep reenvia un altre cop el missatge, així fins que passat un cert nombre d'intents informa de falla.

3.2.3 Característiques addicionals de ZigBee

L'estàndard ZigBee es basa en l'estàndard IEEE 802.15.4 i afegeix funcionalitat de suport al treball en xarxa i a l'aplicació. De les moltes característiques adicionals unes quantes estan orientades a la coexistència i mitigament d'interferències.

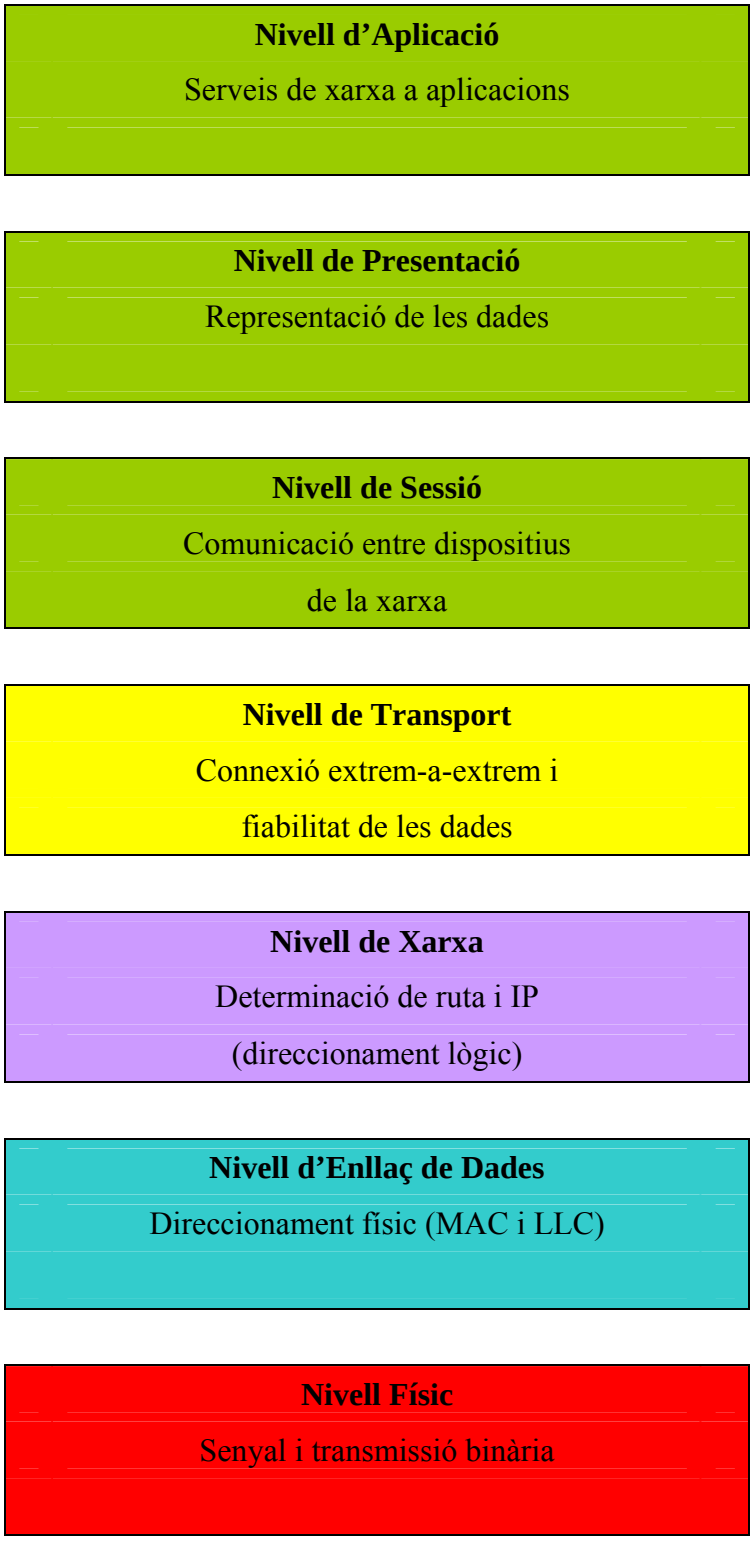
3.2.3.1 Procés de formació de xarxa

Quan es forma una xarxa ZigBee, el dispositiu que inicia la formació, el Coordinador ZigBee (ZC), ha d'escanejar els canals utilitzant les característiques proporcionades per 802.15.4, i automàticament seleccionar el millor canal amb menys interferències.

3.3 Primer contacte

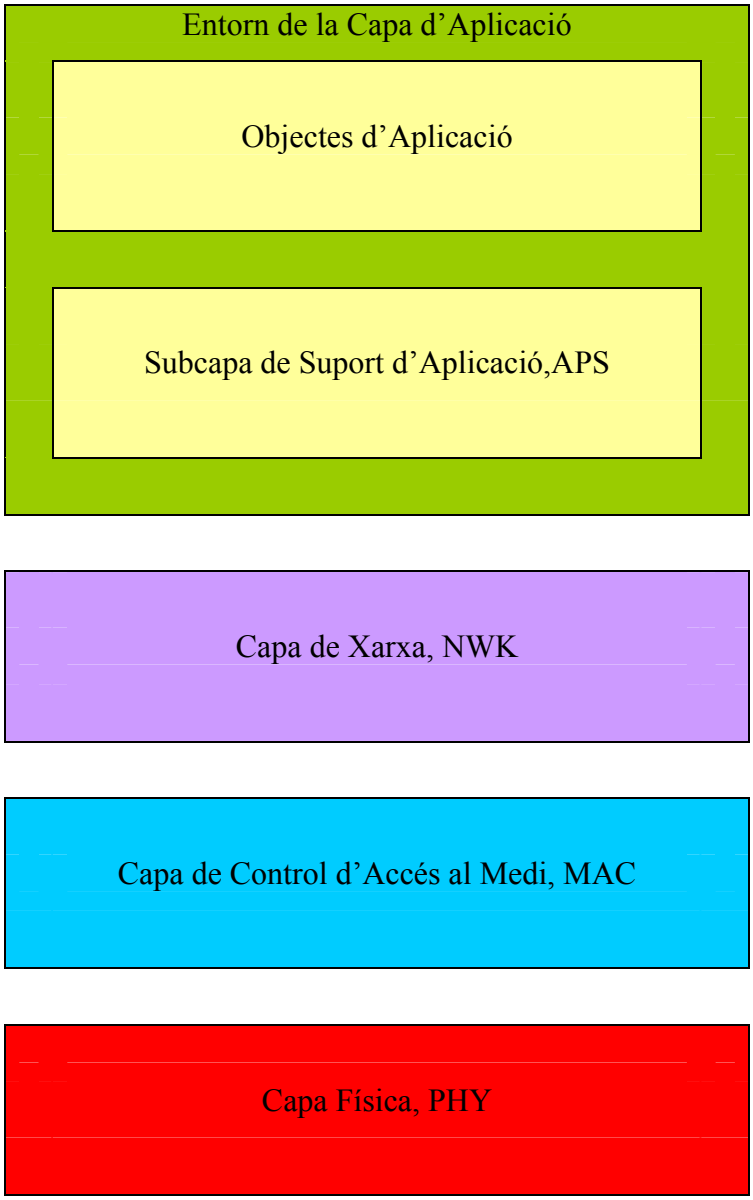
Es defineix ZigBee com una pila de protocols que permet la comunicació de forma senzilla entre múltiples dispositius. Especifica diverses capes, adequant-se al model OSI (Open System Interconnection). Les capes bàsiques, la física (PHY) i de control d'accés al medi (MAC) estan definides per l'estàndard IEEE 802.15.4 LR-WPAN (Low Rate – Wireless Personal Area Network). Aquest estàndard va ser dissenyat pensant en la simplicitat de la implementació i el baix consum, sense perdre potència ni possibilitats.

Aquesta és la pila OSI:



L'estàndard ZigBee amplia l'estàndard IEEE 802.15.4 aportant una capa de xarxa (NWK) que gestiona les feines d'encaminament i de manteniment dels nodes de la xarxa; i un entorn d'aplicació que proporciona una subcapa d'aplicació (APS) que estableix una interfície per la capa de la xarxa, i els objectes dels dispositius tant de ZigBee com del dissenyador.

Així els estàndards IEEE 802.15.4 i ZigBee es complementen proporcionant una pila completa de protocols que permeten la comunicació entre multitud de dispositius d'una manera eficient i senzilla.



3.4 Característiques significatives de ZigBee/IEEE 802.15.4

Entre moltes altres peculiaritats, podem destacar:

IEEE 802.15.4:

- Varies bandes de treball: 2.4 GHz (16 canals), 915 MHz (10 canals), 868 MHz (1 canal)
- Taxes de transferència: 250 Kb/s (2.4 GHz), 40 Kb/s (915 MHz), 20 Kb/s (868 MHz)
- Topologies: estrella i p2p (punt-a-punt)
- Adreçament MAC retallat (16 bits) i estès (64 bits)
- Mètode d'accés al canal: CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance)
- Suporta xarxes slotted (QoS) i non-slotted
- Baix consum energètic
- Gran densitat de nodes per xarxa
- Distància d'abast mitjana de la ràdio: 50 m (fins a 500 m, depenent de l'entorn)

ZigBee:

- Adreçament a nivell de xarxa (16 bits)
- Suport per encaminament de paquets
- Permet topologia de malla, gràcies a les possibilitats d'encaminament
- Dispositius FFD (Full Function Device: coordinador, router i dispositiu final) i RFD (Reduced Function Device: dispositiu final)

3.5 Requeriments hardware

ZigBee és un estàndard que requereix una implementació per poder funcionar. Aquesta pot fer-se per software en multitud d'arquitectures. Però, independentment d'on s'implementi, necessita uns recursos mínims. Ja que els dispositius poden efectuar diferents rols, els requeriments varien d'uns a altres.

· Un microcontrolador de 8 bits
· Pila completa, menys de 32 KB
· Pila senzilla, 6 KB aproximadament

Pel que fa a la memòria RAM, cada implementació necessita una quantitat diferent, en funció del grau d'optimització de la mateixa, però és d'interès ressaltar que els coordinadors i/o routers tindran més exigències ja que necessiten mantenir taules per els dispositius de la xarxa, enllaçament, etc...

3.6 Arquitectura de la pila (stack)

L'arquitectura de la pila ZigBee està composta d'un conjunt de blocs anomenats capes (layers). Cada capa realitza un conjunt específic de serveis per a la capa superior: una entitat de dades proporciona un servei de transmissió de dades i una entitat de gestió proporciona tots els altres serveis.

Cada entitat de servei exposa una interfície a la capa superior a traves d'un punt d'accés al servei (SAP), i cada SAP suporta un nombre de serveis primitius per a dur a terme la funcionalitat requerida.

L'arquitectura de la pila ZigBee, tal com ja hem comentat anteriorment, es basa en el model de set capes de l'estàndard d'Interconnexió de Sistemes Oberts (OSI), però només defineix aquelles capes rellevants per aconseguir funcionalitat en l'espai de mercat previst. L'estàndard IEEE 802.15.4-2003 defineix les dues capes més baixes: la física (PHY) i la sub-capa de control d'accés al medi (MAC). L'aliança construeix en aquests fonaments proveint la capa de xarxa (NWK) i el *framework* per la capa d'aplicació. El *framework* de la capa d'aplicació està compost de la sub-capa de suport a l'aplicació (APS), el ZigBee Device Objects (ZDO) i els objectes definits per l'aplicació del fabricant.

IEEE 802.15.4-2003 té dos capes PHY que operen en dos rangs de freqüència separats: 868/915 MHz i 2.4 GHz. La capa PHY de freqüència més baixa cobreix les dos bandes l'Europea de 868 MHz i la d'USA i Australia de 915 MHz. La capa PHY de freqüència més alta és utilitzada a tot el món.

La sub-capa MAC IEEE 803.15.4-2003 controla l'accés al canal de ràdio utilitzant un mecanisme CSMA-CA. Les seves responsabilitats també poden incloure transmetre beacon frames, sincronització i proveir un mecanisme de transmissió de confiança.

Aquest és l'esquema de la pila ZigBee:

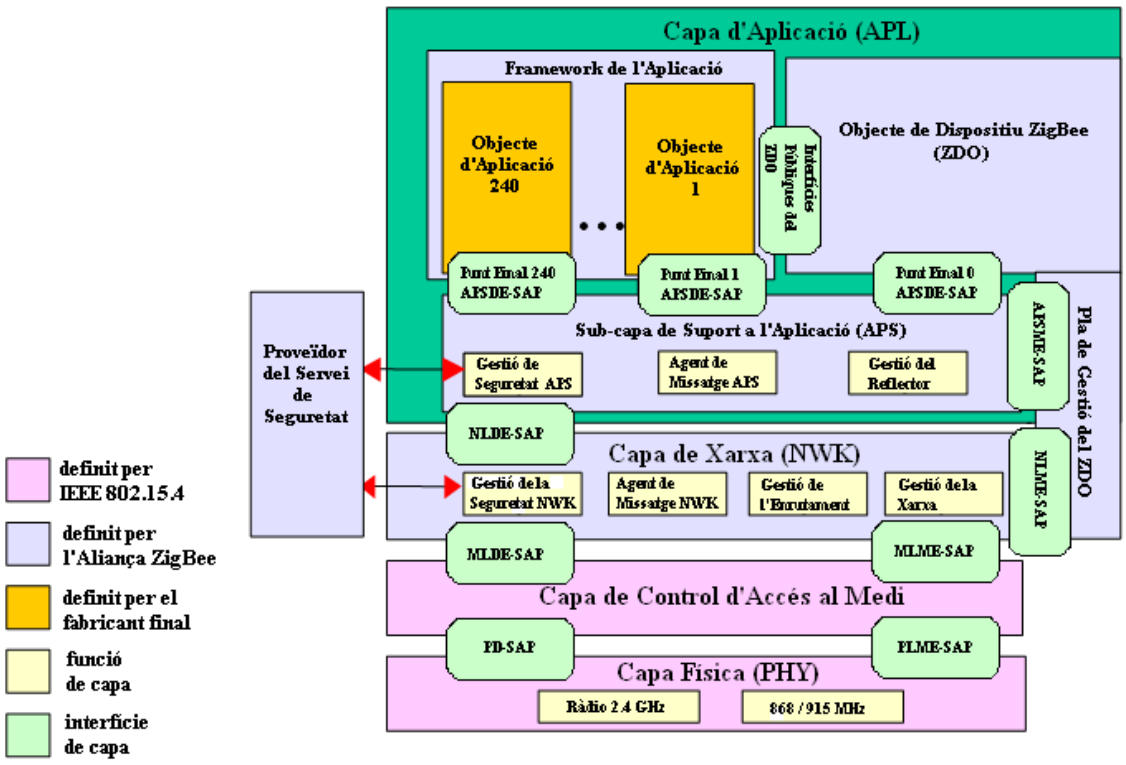


Figura 3.6.1 Pila ZigBee

Les responsabilitats de la capa NWK inclou mecanismes utilitzats per:

- Afegir-se i deixar una xarxa
- Aplicar seguretat al frames
- Encaminar frames cap al seus destins
- Descobrir i mantenir rutes entre dispositius
- Descobrir veïns "d'un sol salt"
- Emmagatzemar informació pertinent dels veïns

La capa NWK d'un coordinador ZigBee és responsable de començar una nova xarxa, quan sigui apropiat, i assignar adreces als dispositius que s'associen novament.

La capa d'aplicació ZigBee consisteix de l'APS, el Famework de l'Aplicació (AF), el ZDO i els objectes d'aplicació definits per el fabricant.

Les responsabilitats de la sub-capa APS inclouen:

- Manteniment de taules de vincles, definit com l'habilitat de relacionar dos dispositius junts en funció dels seus serveis i les seves necessitats.
- Transferir missatges entre dispositius vinculats.

Les responsabilitats del ZDO inclouen:

- Definir el rol del dispositiu dins de la xarxa (per exemple, coordinador o dispositiu final).
- Iniciar i/o respondre a una petició de vincle.

- Establir una relació segura entre dispositius de xarxa.

El ZDO també és responsable de descobrir dispositius en la xarxa i determinar quins serveis d'aplicació ofereixen.

3.7 Topologia de xarxa i dispositius

Profunditzem una mica més en les possibilitats a l'hora de crear xarxes IEEE 802.15.4/ZigBee. L'estàndard de la IEEE especifica dos tipus de dispositius: de funció reduïda (RFD, Reduced Function Device) i de funció completa (FFD, Full Function Device), dissenyats per a propòsits diferents.

El RFD està pensat per aplicacions molt senzilles, com interruptors d'iluminació i sensors infrarojos, que no necessiten enviar i rebre grans quantitats d'informació. Tant sols pot comunicar-se amb dispositius FFD. Tot això permet que pugui ser implementat utilitzant els mínims recursos possibles, així com un estalvi energètic visible. En canvi, els FFD poden actuar com a coordinadors o com a dispositius finals. Poden comunicar-se amb altres FFD i RFD. Degut a això necessiten més recursos, han d'implementar la pila completa i precisen un consum més exigent.

ZigBee aprofita aquesta diferenciació. A més a més del coordinador de la xarxa, és possible l'existència de routers, evidentment han de ser FFD, que augmenten les possibles topologies de xarxa.

Per poder tenir una xarxa són necessaris com a mínim dos elements. Un coordinador (FFD) que crearà la xarxa, li assignarà el NWKID (NetWork IDentifier), i posseirà els elements necessaris per a la incorporació i eliminació de nodes a la xarxa. A més a més és necessari, com a mínim, un node, que pot ser FFD o RFD, amb el que comunicar-se.

La capa de xarxa ZigBee (NWK) suporta topologies estrella, arbre i mesh. En una topologia estrella, la xarxa és controlada per un sol dispositiu, un coordinador ZigBee (FFD). El coordinador és responsable d'iniciar i mantenir els dispositius de la xarxa, i tots els altres dispositius, coneguts com a dispositius finals (RFD o FFD), es comuniquen directament amb el coordinador.

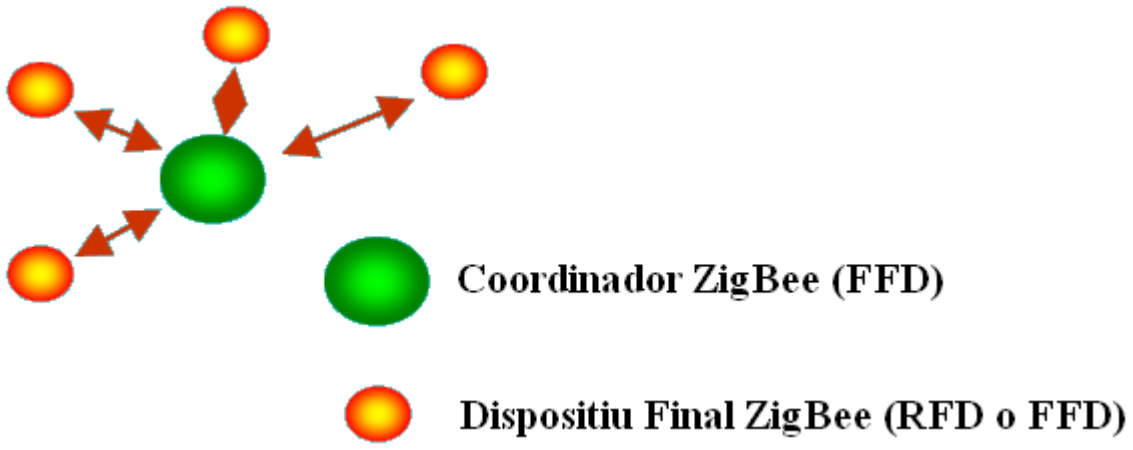


Figura 3.7.1 Xarxa estrella

En les topologies mesh i tree, el coordinador ZigBee és responsable d'iniciar la xarxa i d'elegir certs paràmetres de xarxa claus però la xarxa pot ser ampliada utilitzant routers ZigBee. En les xarxes tree, els routers mouen dades i missatges de control a través de la xarxa utilitzant una estratègia d'encaminament jeràrquic. Les xarxes tree poden utilitzar comunicacions orientades a beacon tal com es descriu en l'especificació IEEE 802.15.4-2003. Les xarxes mesh permetran comunicacions peer-to-peer completes. Els routers ZigBee en les xarxes mesh no emitiran beacons IEEE 802.15.4-2003 regularment. Aquesta especificació només descriu xarxes intra-PAN, que són aquelles xarxes en les que les comunicacions comencen i acaben dins de la mateixa xarxa.

Xarxa Mesh:

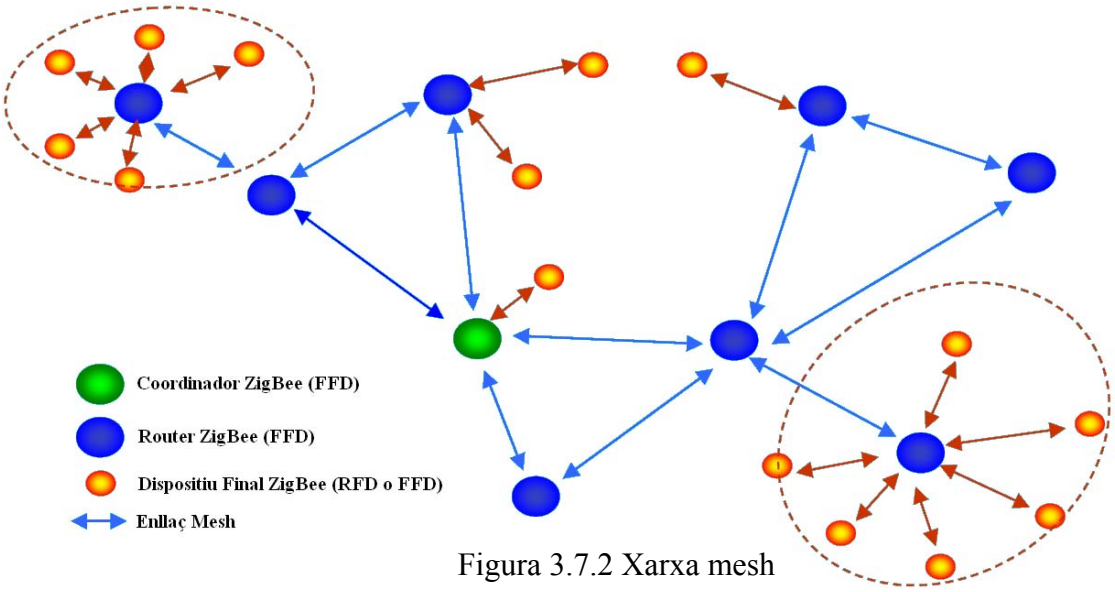


Figura 3.7.2 Xarxa mesh

Xarxa Clúster Tree:

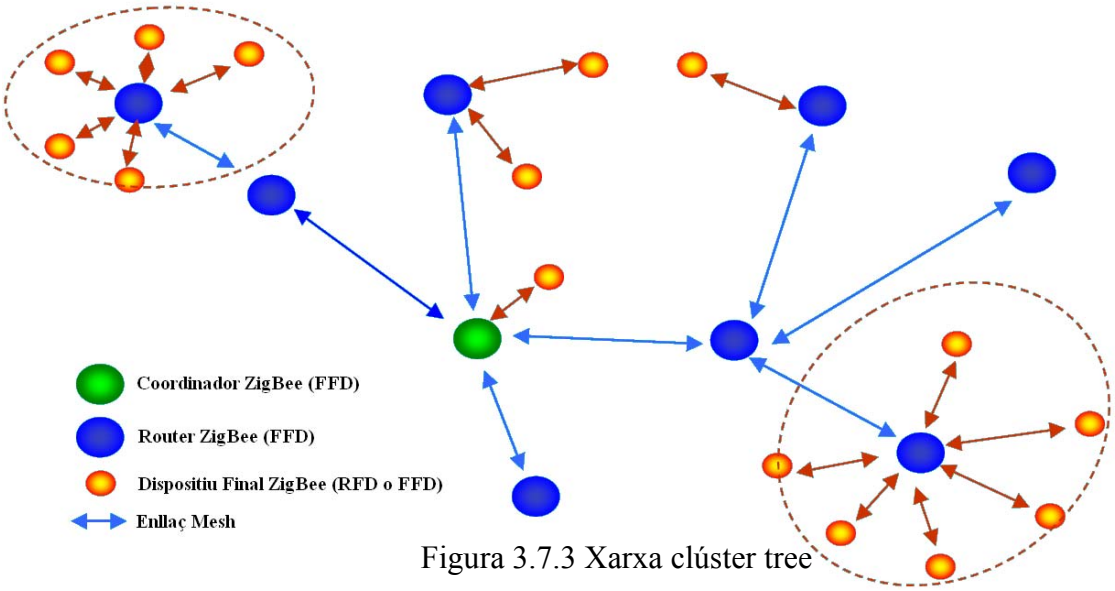


Figura 3.7.3 Xarxa clúster tree

3.8 Comunicacions

Les dades que realment es desitgen enviar comencen en les capes superiors de la pila, i cada capa afegeix informació pròpia, formant els PDU (Protocol Data Unit). Així doncs, quan s'envia un conjunt de dades, aquest conté informació de control de totes les capes de la pila. Quan arriba al seu destí, cada capa extreu les dades que la involucren i, si és possible o necessari, passa la resta a la capa superior. Així es produeix una comunicació virtual entre capes de diferents dispositius.

Comencem per el més bàsic: la comunicació a nivell físic. Els dispositius sense fils envien les dades utilitzant ones electromagnètiques. En aquest cas s'utilitza modulació per freqüència, concretament la ràdio utilitza codificació DSSS la qual és dirigida cap al modulador, la modulació en les bandes dels 868 i 915 MHz és BSPK, i en l'espectre dels 2'4 GHz s'utilitza QPSK ortogonal. Ens centrarem en la banda dels 2'4 GHz, per tant tenim 16 canals disponibles en els que transmetre (separats entre si 5 MHz).

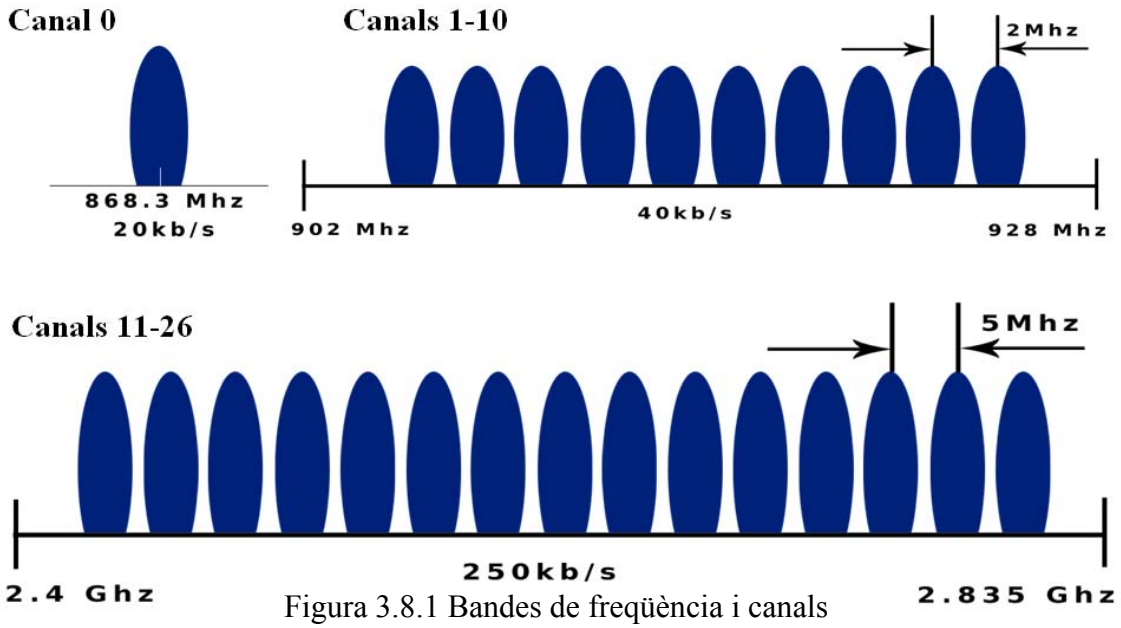


Figura 3.8.1 Bandes de freqüència i canals

L'accés al canal es fa utilitzant CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance) que és un mecanisme utilitzat per evitar que dos dispositius utilitzin el mateix canal a la vegada, produint col·lisió. Així doncs, quan un dispositiu transmet, la resta espera. Tots els dispositius que estiguin en el radi d'abast del transmissor podran escoltar el missatge. Però la major part de les vegades volem comunicar-nos amb un de sol.

Per dur a terme això, necessitem alguna manera d'identificar els dispositius dins de la xarxa. Això ens ho prové la capa superior: la capa MAC. Així doncs, cada dispositiu posseeix una direcció MAC que ha de ser única, de 64 bits. Es pot utilitzar aquesta mateixa en les comunicacions dins de la xarxa, o es pot intercanviar amb el coordinador de la PAN (Personal Area Network) per una de més curta de 16 bits. Aquesta direcció és la que identifica l'origen i el destí d'una trama dins de la xarxa.

Cada trama ha de tenir una mida màxima de 127 bytes, incloent les capçaleres MAC, que són com a màxim de 25 bytes (sense utilitzar seguretat).

Tenint això en compte, i considerant les dues topologies de xarxa possibles a aquest nivell, a saber estrella i peer-to-peer, només podem enviar dades als nodes que estiguin dins del nostre radi d'abast. És més, si el dispositiu és RFD, només podrà enviar dades al coordinador. Això és molt limitat. Per solucionar-ho, hem de confiar en la següent capa: la capa de xarxa (NWK, NetWorK).

Fins ara, tot el que hem vist pertany a les especificacions de l'estàndard IEEE 802.15.4. La capa de xarxa l'aporta ZigBee. La principal tasca d'aquesta capa és proveir d'un nivell major d'abstracció. Afegeix una nova direcció lògica als dispositius (16 bits). Això permet que es puguin enviar dades a altres nodes que no estan dins la cobertura de transmissió. Per dur-ho a terme és necessari comptar amb uns dispositius especials que encaminen les dades a través de la xarxa, anomenats routers (que han de ser FFD). Ara si que podem crear una xarxa amplia en la que cada node pot comunicar-se amb tots els altres nodes de la mateixa xarxa. A més a més podem fer xarxes utilitzant topologies diferents, com malla o cluster-tree.

Per tal de dur a terme la seva tasca, la capa de xarxa proporciona dos serveis, un de dades (NLDE, Network Layer Data Entity) i un altre de gestió (NLME, Network Layer Management Entity). El NLDE encapsula les dades de la capa superior afegint-hi les capçaleres necessàries, i les passa a la capa MAC, per ser enviades al destí. També s'encarrega de retransmetre aquelles NPDUs (Network Protocol Data Unit) que tenen com a destí un altre node de la xarxa (*routing*).

El NLME coordina les tasques de manteniment de la xarxa: crear una nova xarxa o associar-se/desassociar-se a un existent, adreçament de dispositius, descobriment de nodes veïns i rutes, control de recepció de dades, etc...

Amb les capes PHY-MAC-NWK podem crear una xarxa completa, permetent a tots els nodes, poder comunicar-se amb altre nodes, de la mateixa o de diferent xarxa, solucionant problemes com l'accés simultani al canal o adreçament lògic de nodes. Tot i així, per permetre més funcionalitats afegim una capa a la pila que interactua entre els objectes de l'aplicació que es desitgi utilitzar i la capa de xarxa. Parlem de la subcapa d'Aplicació (APS, Aplication Support sub-layer).

La capa APS és l'encarregada d'enviar les PDUs d'una aplicació entre dos o més dispositius (anomenada APSDE, APS Data Entity) i de descobrir i enllaçar els dispositius i mantenir una base de dades dels objectes controlats, coneguda com AIB, APS Information Base. Dos dispositius s'enllacen en la AIB en funció dels serveis que ofereixen i de les seves necessitats. Això és útil per l'adreçament indirecte. Així, és possible enviar un paquet a un dispositiu en funció de la direcció origen, ja que estan vinculats. Aquesta taula només pot estar present en el dispositiu coordinador o en un designat a tal efecte.

Quan la comunicació és directa, s'han d'especificar les direccions d'origen i destí del paquet. En canvi, en les comunicacions indirectes llur origen tingui una entrada en la taula d'enllaçament (AIB), l'emissor només necessita especificar l'origen i enviar

les dades al coordinador, que s'encarregarà de fer-los arribar al destí, que poden ser varis dispositius, en funció de la AIB.

3.9 Encaminament (routing)

La capa MAC ens ofereix un identificador únic de 64 bits per a l'adreçament, i la capa de xarxa un altre de 16 bits. Això implica que podem tenir una gran quantitat de nodes en una mateixa xarxa. Però, si només som capaços de comunicar-nos amb aquells que tenim en el radi d'abast, la xarxa està molt limitada. Per això, existeixen les tècniques d'encaminament. Es creen dispositius que reenvien aquells missatges que no van dirigits a si mateixos. Així doncs, qualsevol node de la xarxa pot comunicar-se amb qualsevol altre node.

Els routers són dispositius de propòsit específic. Han de posseir tota la funcionalitat (FFD). No poden entrar en mode "estalvi d'energia" com els RFD, ja que han de ser capaços de retransmetre els missatges el més aviat possible. Per la mateixa raó, han d'escoltar el trànsit constantment. Han de mantenir taules amb les rutes descobertes i funcionalitat per participar o iniciar el descobriment de noves o millors rutes. També han de ser capaços de detectar i corregir errors. Les taules contenen informació sobre el cost de cada ruta. El cost determina quina és la millor en un moment donat. La funció que elegeix el cost d'una ruta es determina alhora de crear la implementació de la pila. Es pot basar en latència del recorregut dels missatges, però és recomanable que es tingui en consideració la carga mitja de la bateria dels dispositius que participen en la ruta.

En les taules d'encaminament poden aparèixer direccions de qualsevol dispositiu, però només els routers poden participar en els mètodes d'encaminament. Si un missatge arriba a un dispositiu FFD que no és un router, comprova la direcció de destí, i tant sols els reenvia si pertany a algun dels seus fills (és a dir, pertany a algun dels RFD que estan associats a ell). Si és per a ell, el passa a la capa superior. En cas contrari es descarta.

3.10 Perfils

Ja que ZigBee està pensat per a la comunicació entre diversos dispositius, possiblement de fabricants diferents, és necessari un mecanisme per fer compatible els missatges, comandes, etc. que poden enviar-se els uns als altres. Per això existeixen els perfils de ZigBee.

Els perfils són la clau per a la comunicació entre dispositius ZigBee. Defineixen els mètodes de comunicació, el tipus de missatge a utilitzar, les comandes disponibles i les respostes, etc. que permeten a dispositius separats comunicar-se per crear una aplicació distribuïda. Gairebé tot el tipus d'operacions han de ser definides en un perfil. Per exemple, les tasques típiques d'unir-se a una xarxa o descobrir dispositius i serveis estan suportades per el "perfil de dispositius" ZigBee.

Cada perfil ha de tenir un identificador i, òbviament, aquest ha de ser únic. Per això, la ZigBee Alliance es reserva el dret d'assignar identificadors als diversos perfils. Si és necessari la creació d'un nou perfil, ha de fer-se la petició a la ZigBee Alliance.

Cada perfil conté les descripcions dels dispositius que inclou, els identificadors de cada clúster (i en el seu cas, els seus atributs) i els tipus de servei oferts. Les descripcions dels serveis estan definides per un valor de 16 bits (és a dir, hi ha 65536 possibles descripcions). Els identificadors de clúster són de 8 bits (existeixen 256 possibles clústers). Tanmateix, si el tipus de servei ofert és orientat a parelles clau-valor, cada clúster pot contenir atributs per valor de 16 bits (o 65536 atributs per clúster). Existeixen moltes possibilitats dins d'un mateix perfil, i és tasca del dissenyador del perfil adequar les necessitats per crear descriptors senzills i permetre un procés eficient de missatges.

Ja que cada dispositiu pot suportar més d'un perfil, i cada perfil pot posseir varis clústers i múltiples descripcions, és necessària una jerarquia d'adreçament per accedir als elements del dispositiu. En primer lloc, es fa referència al dispositiu sencer utilitzant les seves direccions IEEE i NWK. Per una altra banda, es defineixen els Endpoints, que són camps de 8 bits que apunten a cada una de les diferents aplicacions que estan suportades per un dispositiu. Per exemple, el 0x00 fa referència al endpoint del perfil del dispositiu, i el 0xff fa referència a tots els endpoint actius (endpoint broadcasting). Ja que els endpoints 0xf1 – 0xfe estan reservats, és possible tenir un total de 240 aplicacions en els endpoints 0x01 – 0xf0.

Una vegada establerts els endpoints per a les aplicacions, s'han de definir els descriptors. Com a mínim, el descriptor "simple" ha de ser present i disponible per les tasques de descobriment de servei. Existeixen altres tipus de descriptors que contenen informació sobre l'aplicació, del servei, etc... Poden contenir informació sobre l'endpoint al que fan referència, el perfil o la versió, però també sobre el tipus d'alimentació del dispositiu, el nivell de bateria, el fabricant, número de sèrie i fins i tot una icona o la direcció d'aquest per representar el node en PCs, PDAs, etc...

Així en resum, podem dir que el Perfil d'Aplicació ZigBee és un acord entre els dispositius ZigBee sobre els continguts del paquet. Aquests perfils són definits per l'Aliança ZigBee. Poden ser classificats en dues categories, Perfil de Dispositiu ZigBee i Perfil d'Aplicació.

El Perfil de Dispositiu ZigBee és utilitzat per el gestor, el qual és realitzat per el ZDO (ZigBee Device Object Entity). L'identificador de perfil és definit com 0x0000.

El Perfil d'Aplicació pot ser dividit en Públic i Privat. El Perfil d'Aplicació Públic és definit per l'Aliança ZigBee, i el Privat pot ser definit per els usuaris. L'Aliança ZigBee ha desenvolupat uns quants Perfils d'Aplicació Públics com poden ser:

- Perfil de control de l'enllumenat de casa
- Perfil d'automatització d'edifici

Si és necessari desenvolupar un nou perfil d'aplicació, els usuaris poden definir components de perfils d'aplicació específics.

Els Perfils d'Aplicació estan constituïts dels 5 següents components:

- Identificador de Perfil: 16-bits
- Identificador de Dispositiu: 16-bits
- Clúster: 8-bits
- Identificador d'Atribut: 16-bits
- Dada d'Atribut: Variable

A continuació fem una breu descripció d'alguns conceptes que intervenen en el que són els perfils d'aplicació ZigBee:

- Identificador de Perfil: És un número únic de 16-bits assignat a cada perfil com poden ser control de casa, automatització d'edifici, i supervisió remota.
- Punt Final: És una unitat funcionat integrada en el dispositiu. Per exemple, un dispositiu d'enlluernament que té quatre interruptors de làmpada tindrà quatre punts finals.
- Identificador de Dispositiu: Indica el tipus de dispositiu físic com pot ser un interruptor de llum, sensor de temperatura, control remot, etc. Cada punt final tindrà el seu propi identificador de dispositiu. Per exemple, si un dispositiu ZigBee té dos punts finals llavors cada punt final tindrà un identificador de dispositiu diferent.
- Clúster: Un conjunt d'identificadors d'atribut.
- Identificador d'atribut: És un número assignat a cada atribut. L'atribut és una comanda específica per cada control com per exemple un control On/Off, control de sensor d'un sol bit, control de sensor multi-bit.
- Dada d'Atribut: És una dada variable per cada identificador d'atribut. La dada d'atribut indica l'actual comportament del control com pot ser un ON o OFF

3.11 Tecnologies actuals

Ja que la ZigBee Alliance només defineix un estàndard, la pila de ZigBee ha de ser implementada per tercers, per tant, existeixen diferents implementacions en el mercat.

Apèndix 4. Comparativa MC1321x (Freescale) vs CC2430 i CC2431 (Texas Instruments)

En aquest apartat realitzarem una breu descripció d'algunes de les característiques de les plataformes de desenvolupament de ZigBee MC1321x de Freescale i CC2430 i CC2431 de Texas Instruments, que ens serviran alhora de seleccionar quina plataforma de desenvolupament adquirir. Per una informació més detallada sobre dites plataformes, consultar les referències [9] (MC1321x), [10] (CC2430) i [11] (CC2431).

4.1 Freescale: MC1321x

La família MC1321x és la plataforma de ZigBee de segona generació de Freescale la qual incorpora un transceptor de ràdio freqüència de baixa potència a 2'4 GHz i un microcontrolador de 8 bits. La solució MC1321x pot ser utilitzada per aplicacions sense fils des de una simple connexió punt a punt fins a una complerta xarxa mesh (és una topologia de xarxa en la que cada node està connectat a un o més dels altres nodes. D'aquesta manera és possible portar els missatges d'un node a un altre per diferents camins) de ZigBee.

El MC1321x conté un transceptor de RF el qual compleix amb l'estàndard IEEE 802.15.4 que opera en la banda de freqüència 2'4 GHz ISM (Industrial, Scientific and Medical). El transceptor inclou un amplificador de baix soroll, potència nominal de sortida de 1 mW, PA amb oscil·lador controlat per tensió (VCO) intern, interruptor transmetre/rebre integrat, regulador de tensió d'alimentació on-board, i codificador i descodificador de tot l'espectre de difusió.

El MC1321x també conté un microcontrolador basat en la família d'Unitats Microcontroladores (MCU) HCS08 [12], concretament el HCS08 Versió A, i pot proporcionar fins a 60 KB de memòria flash i 4 KB de RAM. La MCU onboard permet l' stack de comunicacions i també que l'aplicació resideixi en el mateix system-in-package (SIP).

Característiques:

- Interfície dedicada a perifèrics sèrie (SPI) connectat internament al mòdem 802.15.4
- 8 canals convertidors AD de 8-10 bits

- Dos interfícies de comunicació sèrie independents (SCI)
- Interfície inter-integrated circuit (IIC = I2C)
- Fins a 32 MCU GPIO amb pullups programable
- El mòdem RF afegeix set GPIO més
- El BeeStack de Freescale suporta xarxes star, mesh i tree, així com també l'estàndard avançat d'encriptació (AES [13]) de 128-bits de seguretat

4.2 Texas Instruments: CC2430 i CC2431

El CC2430 és una solució System-on-Chip (SoC) específicament per l'estàndard IEEE 802.15.4 i aplicacions ZigBee. Combina el bon rendiment del transceptor RF CC2420 [14] amb el MCU 8051 [15], fins a 128 KB de memòria flash, 8 KB RAM... i combinat amb l'stack de protocol ZigBee (Z-Stack [16]) de Texas Instruments proporciona una de les solucions ZigBee més competitives.

Característiques:

- Potent DMA (accés directe a memòria)
- Watchdog timer (és un temporitzador hardware que reinicia el sistema si en cert temps no hi ha resposta, per tal de sortir d'un estat penjat a l'operació normal)
- Suport hardware per a accés múltiple per detecció de portadora amb evasió de col·lisions (CSMA/CA), que permet que múltiples estacions utilitzin el mateix mitjà de transmissió.
- Digital RSSI (indicador de nivell de senyal rebut) / LQI (indicador de la qualitat de l'enllaç). Aquestes característiques són útils per poder valorar la qualitat de senyal que tenim
- Fins a 8 convertidors AD de 12 bits
- Coprocessador de seguretat AES
- Dos UARTs (transmissor/receptor universal asíncron) amb suport per varis protocols sèrie
- 21 pins GPIO (entrades/sortides de propòsit general) amb pullups programables

El CC2431 i el CC2430 els seus pins són compatibles, i les parts referents a MCU i RF són idèntics també excepte que el CC2431 compta amb un motor de localització (Location Engine). El motor de localització s'utilitza per estimar la posició dels nodes dins d'una xarxa sense fils. Existeixen nodes de referència amb coordenades conegudes. Altres nodes són els nodes cecs les seves coordenades han de ser estimades. Aquests nodes cecs sovint són mòbils i adjunts a recursos que necessiten ser rastrejats.

Apèndix 5. CC2430

En aquest apartat trobareu un resum de les principals característiques del CC2430, tant a nivell de hardware, Z-Stack i IAR Embedded Workbench. Per aprofundir més i obtenir informació addicional consultar les referències [10] (tot sobre el CC2430), [16] (tot sobre el Z-Stack) i [18] (tot sobre l'IAR)

5.1 Introducció

El CC2430 es tracta d'una solució en format SoC que compleix les especificacions IEEE 802.15.4 / ZigBee a 2'4 GHz.

5.1.1 Aplicacions

- Sistemes IEEE 802.15.4 a 2'4 GHz
- Sistemes ZigBee
- Automatització de la llar/edifici
- Supervisió i control industrial
- Xares de sensors sense fils de baixa potència
- Perifèrics per a PC
- Controls remots
- Electrònica de consum

5.1.2 Descripció del producte

Hi h tres versions diferents del CC2430, depenent de la quantitat de memòria flash que inclouen, CC2430F32/62/128, amb 32/64/128 KB de memòria flash respectivament. El CC2430 és una solució SoC confeccionada per aplicacions de l'IEEE 802.15.4 i ZigBee. Habilita la creació de nodes de ZigBee amb un cost total de material molt baix. El CC2430 combina el transceptor RF CC2420 amb el una model millorat de l'estàndard industrial 8051 MCU, 32/64/128 KB de memòria flash, 8 KB de RAM i altres característiques. Combinat amb l'stack del protocol ZigBee de Texas Instruments (Z-Stack), el CC2430 proporciona una de les solucions més competitives per la solució ZigBee.

5.1.3 Característiques clau

- RF/Layout
  - Transceptor RF (inclou el nucli del CC2420) conforme amb l'IEEE 802.15.4 2'4GHz
  - Receptor amb bona sensibilitat i robust a interferències
  - Molt pocs components externs

- Baixa potència
  - Poc consum de corrent (Rx: 27mA, Tx: 27 mA, amb el microcontrolador treballant a 32 MHz)
  - Només consumeix 0'5 μA en mode powerdown, on una interrupció externa o l'RTC (comptador en temps real) poden despertar el sistema
  - Els temps de transició entre modes de baix consum i mode actiu és molt ràpid i permet un consum mitjà molt baix en sistemes de baix cicle de treball
  - Ampli rang de voltatge d'alimentació (2'0 V – 3'6 V)

- Microcontrolador
  - Nucli microcontrolador d'alt rendiment i baix consum
  - 32, 64 o 128 KB de memòria flash programable en el circuit
  - 8 KB RAM, 4 KB amb retenció de dades en tots els modes de potència
  - DMA
  - Watchdog timer
  - Un IEEE 802.15.4 MAC timer, un timer general de 16-bits i dos tiemrs de 8-bits
  - Suport a depuració del hardware

- Perifèrics
  - Suport hardware CSMA/CA
  - Suport RSSI/LQI digital
  - Supervisió del nivell de bateria i sensor de temperatura
  - Fins a vuit entrades ADC de 12-bits amb resolució configurable
  - Coprocessador de seguretat AES
  - Dos USARTs amb suport per a varis protocols sèrie
  - 21 pins de I/O generals, dos amb capacitat d'absorbir/donar 20mA

- Eines de desenvolupament
  - Hi ha disponibles eines de desenvolupament potents i flexibles

5.2 Característiques emfatitzades

5.2.1 Microcontrolador compatible amb 8051 de baixa potència i alt rendiment

- Nucli del 8051 optimitzat, que obté un rendiment vuit vegades d'un model estàndard del 8051
- Punters de dades dual
- Suporta depuració interactiva en circuit utilitzant l'IAR Embedded Workbench

5.2.2 Fins a 128 KB de memòria no volàtil de programa i dos bancs de 4 KB de memòria de dades

- 32/64/128 KB de memòria flash no volàtil en el sistema
- En el pitjor dels casos, l'esperança de vida de la memòria flash és de 1000 cicles d'escriure/esborrar
- Bloqueig programable d'escriptura i lectura de porcions de la memòria flash per seguretat del codi
- 4096 bytes d'SRAM interna amb retenció de dades en tots els modes de potència
- 4096 bytes addicionals d'SRAM interna amb retenció de dades en modes de potència 0 i 1

5.2.3 Encriptació/desencriptació AES per Hardware

- Coprocessador hardware que suporta AES

5.2.4 Característiques de perifèrics

- DMA
- Vuit canals ADC amb resolució configurable
- Watchdog timer programable
- Rellotge de temps real amb oscil·lador de cristall a 32'768 KHz
- Quatre timers: un general de 16-bits, dos generals de 8-bits, un MAC
- Dos USARTs programables per operar com a SPI mestre/esclau o com a UART
- 21 pins d'I/O digitals de propòsit general configurables

5.2.5 Baixa potència

- Quatre modes de potència per reduir el consum
- El sistema pot despertar degut a una interrupció externa o un esdeveniment del comptador en temps real
- Disseny CMOS totalment estàtic de baixa potència
- El rellotge font del sistema pot ser un oscil·lador RC (resistència – condensador) a 16 MHz o un oscil·lador de cristall a 32 MHz. L'oscil·lador a 32 MHz s'utilitza quan la ràdio està activa
- Opcionalment es pot utilitzar una altra font de rellotge per operació de ultra baixa potència ja sigui utilitzant un oscil·lador RC de baixa potència o un oscil·lador de cristall a 32'768 KHz

5.2.6 Suport hardware IEEE 802.15.4 MAC

- Generador de preàmbul automàtic
- Inserció/detecció de paraula de sincronització
- Computació i comprovació del CRC-16 en el "payload" de MAC
- Avaluació de canal lliure
- Detecció d'energia / indicador digital de la intensitat dels senyals en el canal
- Indicador de la qualitat de l'enllaç
- Coprocessador CSMA/CA

5.2.7 Ràdio digital DSSS a 2'4 GHz integrada

- Transceptor RF conforme amb IEEE 802.15.4 2'4 GHz (basat en el nucli de ràdio CC2420)
- Receptor molt sensible i robust a interferències
- Taxa de dades 250 Kbps
- Els dissenys de referència són conformes amb les regulacions mundial sobre ràdio freqüències del ETSI EN 300 328 i EN 300 440 classe 2 (Europa), FCC CFR47 Part 15 (US) i ARIB STD-T66 (Japó). La transmissió a 2480 MHz sota FCC es fa mitjançant cicle de treball, o reduint la potència de sortida.

5.3 Configuració pins i ports I/O

Aquesta és la distribució dels pins del CC2430.

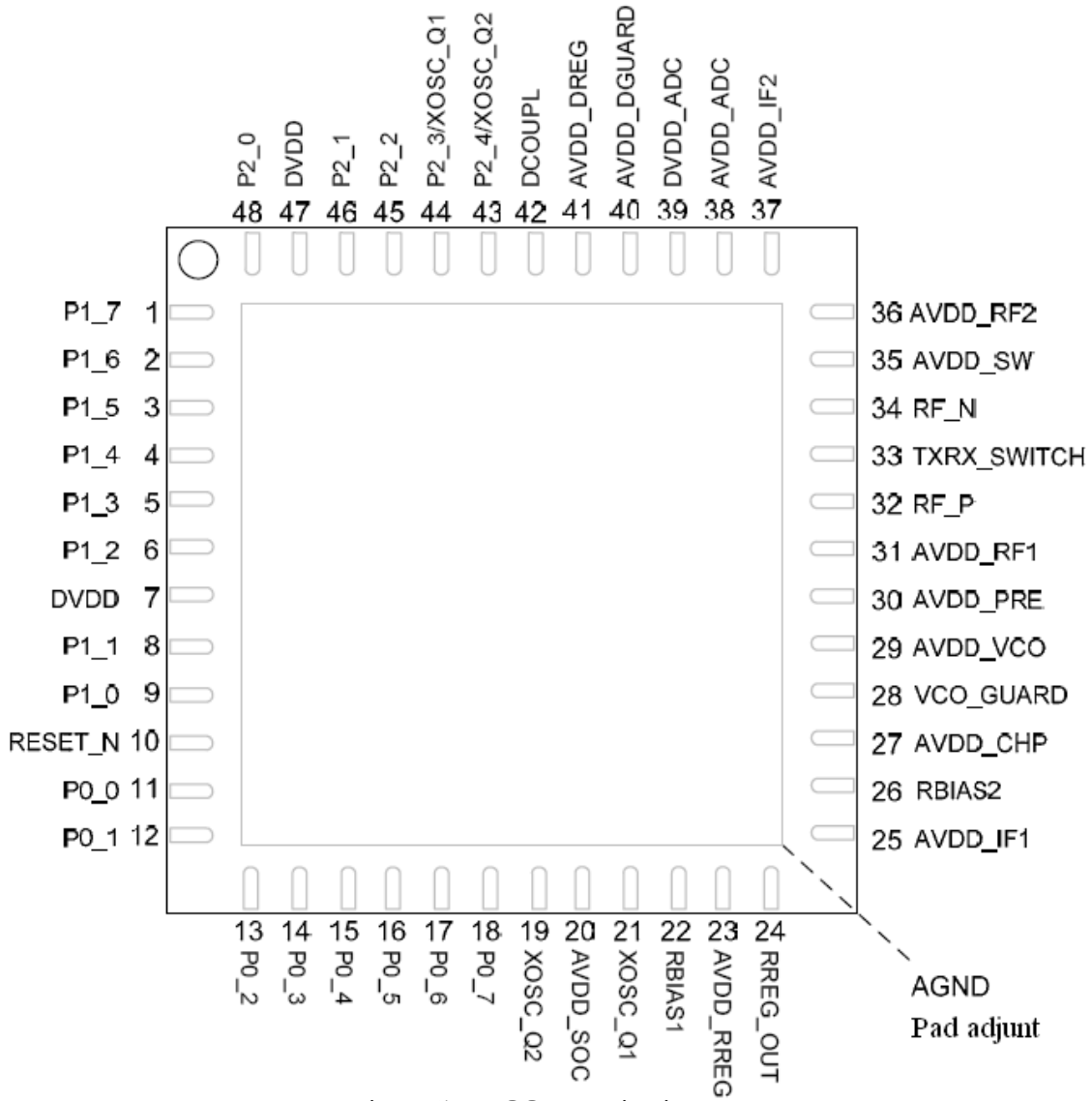


Figura 5.3.1 CC2430 pins i ports

El pad adjunt s'ha de connectar a un terres sòlid, ja que és la connexió de terra del chip

La següent taula mostra la descripció dels pins:

Pin	Nom del pin	Tipus de pin	Descripció
-	GND	Terra	El pad adjunt s'ha de connectar a un terres sòlid
1	P1_7	I/O digital	Port 1.7
2	P1_6	I/O digital	Port 1.6
3	P1_5	I/O digital	Port 1.5
4	P1_4	I/O digital	Port 1.4
5	P1_3	I/O digital	Port 1.3
6	P1_2	I/O digital	Port 1.2
7	DVDD	Power (digital)	2'0V-3'6V alimentació digital per les I/O digitals
8	P1_1	I/O digital	Port 1.1 – capacitat 20 mA
9	P1_0	I/O digital	Port 1.0 – capacitat 20 mA
10	RESET_N	Entrada digital	Reset, actiu a baixa
11	P0_0	I/O digital	Port 0.0
12	P0_1	I/O digital	Port 0.1
13	P0_2	I/O digital	Port 0.2
14	P0_3	I/O digital	Port 0.3
15	P0_4	I/O digital	Port 0.4
16	P0_5	I/O digital	Port 0.5
17	P0_6	I/O digital	Port 0.6
18	P0_7	I/O digital	Port 0.7
19	XOSC_Q2	I/O analògica	Pin 2 del oscil·lador de cristall a 32MHz
20	AVDD_SOC	Power (analògica)	2'0V-3'6V connector d'alimentació analògica
21	XOSC_Q1	I/O analògica	Pin 1 de l'oscil·lador de cristall a 32 MHz, o entrada rellotge extern
22	RBIAS1	I/O analògica	Resistència bias externa de precisió per corrent de referència
23	AVDD_RREG	Power (analògica)	2'0V-3'6V connector d'alimentació analògica
24	RREG_OUT	Sortida power	Sortida del regulador de tensió d'alimentació a 1'8 V. Només previst per alimentar la part analògica d'1'8V (alimentació pels pins 25, 27-31, 35-40)
25	AVDD_IF1	Power (analògica)	Alimentació d'1'8V per el filtre passa banda del receptor, mòdul de test analògic, bias global i la primera part del VGA
26	RBIAS2	Sortida power	Resistència externa de precisió, 43 KΩ ± 1 %
27	AVDD_CHIP	Power (analògica)	Alimentació d'1'8V per el detector de fase, la bomba de carga i la primera part del filtre loop
28	VCO_GUARD	Power (analògica)	Connexió de l'anell de guarda per la protecció del VCO (a AVDD)
29	AVDD_VCO	Power (analògica)	Alimentació d'1'8V per el VCO i l'última part del filtre loop PLL
30	AVDD_PRE	Power (analògica)	Alimentació d'1'8V per el preescalador, Div-2 i els buffers LO
31	AVDD_RF1	Power (analògica)	Alimentació d'1'8V perel LNA, front-end bias i l'amplificador de potència (PA)
32	RF_P	RF I/O	Senyal d'entrada RF positiu a LNA durant RX. Senyal de sortida RF positiu des de PA durant TX
33	TXRX_SWITCH	Power (analògica)	Voltatge d'alimentació regulat per el PA
34	RF_N	RF I/O	Senyal d'entrada RF negatiu a LNA durant RX. Senyal de sortida RF negatiu des de PA durant TX
35	AVDD_SW	Power (analògica)	Alimentació d'1'8V per el selector LNA/PA

36	AVDD_RF2	Power (analògica)	Alimentació d'1'8V per els mescladors de rebre i transmetre
37	AVDD_IF2	Power (analògica)	Alimentació d'1'8V per el filtre passa baixos de transmissió i l'última etapa del VGA
38	AVDD_ADC	Power (analògica)	Alimentació d'1'8V per les parts analògiques dels ADCs i el DACs
39	DVDD_ADC	Power (digital)	Alimentació d'1'8V per les parts digitals dels ADCs
40	AVDD_DGUARD	Power (digital)	Connexió d'alimentació per l'aïllament digital de soroll
41	AVDD_DREG	Power (digital)	2'0V-3'6V alimentació digital per el regulador de tensió del nucli digital
42	DCOUPL	Power (digital)	Alimentació digital d'1'8V per el decoplement. No utilitzar per alimentar circuits externs
43	P2_4/XOSC_Q2	I/O digital	Port 2.4/32'768 KHz XOSC
44	P2_3/XOSC_Q1	I/O digital	Port 2.3/32'768 KHz XOSC
45	P2_2	I/O digital	Port 2.2
46	P2_1	I/O digital	Port 2.1
47	DVDD	Power (digital)	2'0V-3'6V alimentació digital per les I/O digitals
48	P2_0	I/O digital	Port 2.0

5.4 Descripció del circuit

La següent figura mostra el diagrama de blocs del CC2430. Els mòduls poden ser dividits, de manera aproximada, en tres categories: relacionats amb la CPU, relacionats amb la potència, distribució del rellotge i test, i relacionats amb la ràdio.

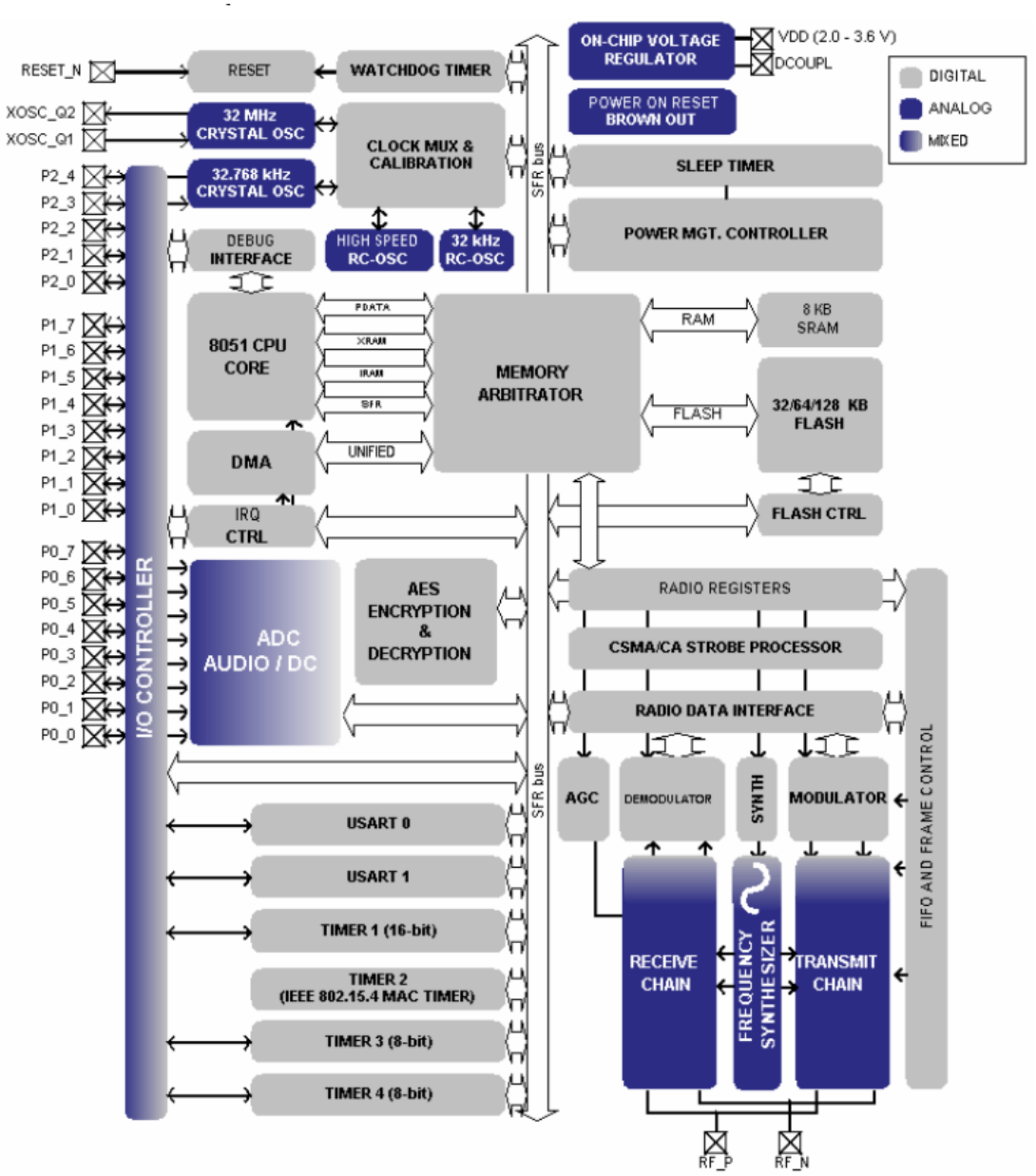


Figura 5.4.1 Diagrama de blocs CC2430

5.4.1 CPU i perifèrics

El nucli de la CPU 8051 és un nucli d'un sol cicle compatible amb el 8051. Té tres busos d'accés a memòria diferents (SFR, DATA i CODE/XDATA), una interfície de depuració i una unitat d'interrupcions de 18 entrades.

La commutació/arbitració de la memòria es troba al cor del sistema i connecta la CPU i el controlador DMA amb les memòries físiques i tots els perifèrics a través del bus SFR. L'àrbitre de la memòria té quatre punts d'accés a memòria, accés a quin es pot mapejar de les tres memòries físiques: 8 KB de SRAM, memòria flash o RF i registres SFR. L'àrbitre de memòria és responsable de realitzar l'arbitratge i seqüenciar els accessos simultanis a la mateixa memòria física.

El bus SFR és un bus comú que connecta tots els perifèrics hardware amb l'àrbitre de memòria. També dona accés al registres de ràdio en el banc de registres de ràdio encara que aquests estiguin mapejats en l'espai de memòria XDATA.

Els 8 KB de memòria SRAM mapejen a l'espai de memòria DATA i a parts dels espais de memòria XDATA. 4 KB dels 8 KB de SRAM són SRAM d'ultra baixa potència que retenen el seu contingut fins i tot quan es deixa d'alimentar la part digital (modes de potència 2 i 3). La resta de SRAM perd el seu contingut quan es deixa d'alimentar la part digital.

El bloc de memòria flash de 32/64/128 KB proporciona una memòria de programa no volàtil programable en circuit per el dispositius i mapeja en els espais de memòria CODE i XDATA. L'escriptura d'un bloc flash es fa a través d'un controlador flash que permet l'esborrat page-wise (2048 bytes) i programar 4 byte-wise.

També compte amb un controlador de DMA de cinc canals en el sistema i accedeix a la memòria utilitzant l'espai de memòria XDATA i així té accés a totes les memòries físiques. Cada canal és configurat (trigger, prioritat, mode de transferència, mode d'adreçament, punters de font i destí, i comptador de transferència) amb els descriptors de DMA en qualsevol lloc de memòria. Molts dels perifèrics hardware depenen del controlador DMA per una operació eficient (el nucli AES, controlador d'escriptura a flash, USARTs, Timers, interfície ADC) mitjançant transferències de dades entre una sola adreça SFR i flash/SRAM.

El controlador d'interrupcions serveix un total de 18 fonts d'interrupcions, dividides en sis grups d'interrupcions, cadascun és associat amb una dels quatre prioritats d'interrupció. Una petició d'interrupció es serveix fins i tot quan el dispositiu està en mode repòs (modes de potència 1-3) portant de nou el CC2430 a mode actiu (mode potència 0).

L'interfície de depuració implementa una pròpia interfície sèrie de dos cables que és utilitzada per la depuració en circuit. A través d'aquesta interfície de depuració és possible realitzar un esborrat complert de la memòria flash, controlar quins oscil·ladors estan habilitats, parar i començar l'execució del programa d'usuari, executar instruccions en el nucli 8051, incloure breakpoints en el codi, executar el codi instrucció per instrucció. Utilitzant aquestes tècniques és possible realitzar, de manera elegant, depuració en circuit i programa memòries flash externes.

El controlador d'I/O és responsable de tots els pins d'I/O de propòsit general. La CPU tant pot configurar que els mòduls perifèrics controlin certs pins com que els pins estiguin sota control software, i també, en els casos que ho permetin, configurar cada pin com a entrada o sortida i si en el pad hi ha connectada una resistència pull-up o pull-

down. Cada perifèric que connecta als pins I/O pot elegir entre dos localitzacions diferents de pin I/O per assegurar flexibilitat en varies aplicacions.

El timer de repòs és un timer d'ultra baixa potència que compte períodes de 32'768 KHz del oscil·lador de cristall o de 32 KHz del oscil·lador RF. El timer de repòs funciona contínuament en tots els mode d'operació excepte el mode de potència 3. Els usos típics són de comptador de temps real que funciona en tots els modes d'operació (excepte el 3), o per despertar el dispositiu, és a dir, treure'l dels modes de potència 1 o 2.

Conté watchdog timer que permet al CC2430 resetejar-se en cas de que es pengi. Quan és habilitat per software, el watchdog timer ha de ser reiniciat periòdicament, si no reiniciarà el dispositiu quan s'acabi el temps.

El timer 1 és un timer de 16-bits amb funcionalitat timer/comptador/PWM. Té una preescala programable, un valor de període de 16-bits i tres canals comptadors/capturadors individualment programables amb valor de comparació de 16-bits. Cadascú dels canals de comptador/capturador pot ser utilitzat com a sortides PWM o per capturar els temps dels flangs en senyals d'entrada.

MAC timer (timer 2) està especialment dissenyat per suportar un IEEE 802.15.4 MAC o altres protocols de time-slotted en software. El timer té un període de temps configurable i un desbordament de comptador de 8-bits que es pot utilitzar per portar el conte del nombre de períodes que s'han resultat. També hi ha un registre de captura de 16-bits utilitzat per enregistrar el temps exacte en el que el començament d'un frame de delimitació és rebut o transmès o el temps exacte de quina transmissió acaba, també un registre comparador de sortida de 16-bits que pot produir varies comandes d'strobe (iniciar RX, iniciar TX, etc) en temps específics als mòduls de ràdio.

Els timers 3 i 4 són timers de 8-bits amb funcionalitat timer/comptador/PWM. Tenen preescala programable, valor de període de 8-bits i un canal comptador programable amb valor de comparació de 8-bits. Cadascun dels canals de comptador pot ser utilitzat com a sortida PWM.

Tant l'USART 0 com l'1 poden ser configurats com a SPI mestre/esclau o UART. Proporcionen buffer doble en ambdós, RX i TX i control de flux hardware. Cadascun té el seu propi generador de precisió de baud-rate d'aquesta manera els timers queden lliures per altres uses.

El nucli d'encriptació/desencriptació AES permet a l'usuari encriptar i desencriptar dades utilitzant l'algorisme AES amb claus de 128-bits. El nucli té capacitat per suportar les operacions AES requerides per la seguretat IEEE 802.15.4 MAC, la capa de xarxa ZigBee i la capa d'aplicació.

L'ADC suporta de 7 a 12 bits de resolució en els amples de banda de 30 KHz a 4 KHz respectivament. Son possibles conversions DC i d'àudio fins a 8 canals d'entrada (Port 0). Les entrades es poden seleccionar com a "single ended" o diferencials. El voltatge de referència pot ser intern, AVDD, o un senyal extern "single ended" o diferencial. L'ADC també té un canal d'entrada de sensor de temperatura. L'ADC pot automatitzar el procés periòdic de mostreig o conversió sobre una seqüència de canals.

5.5 Circuit aplicat

Es necessiten pocs components externs per treballar amb el CC2430. Una aplicació típica (amb els valors típics) és la que es mostra en el següent circuit.

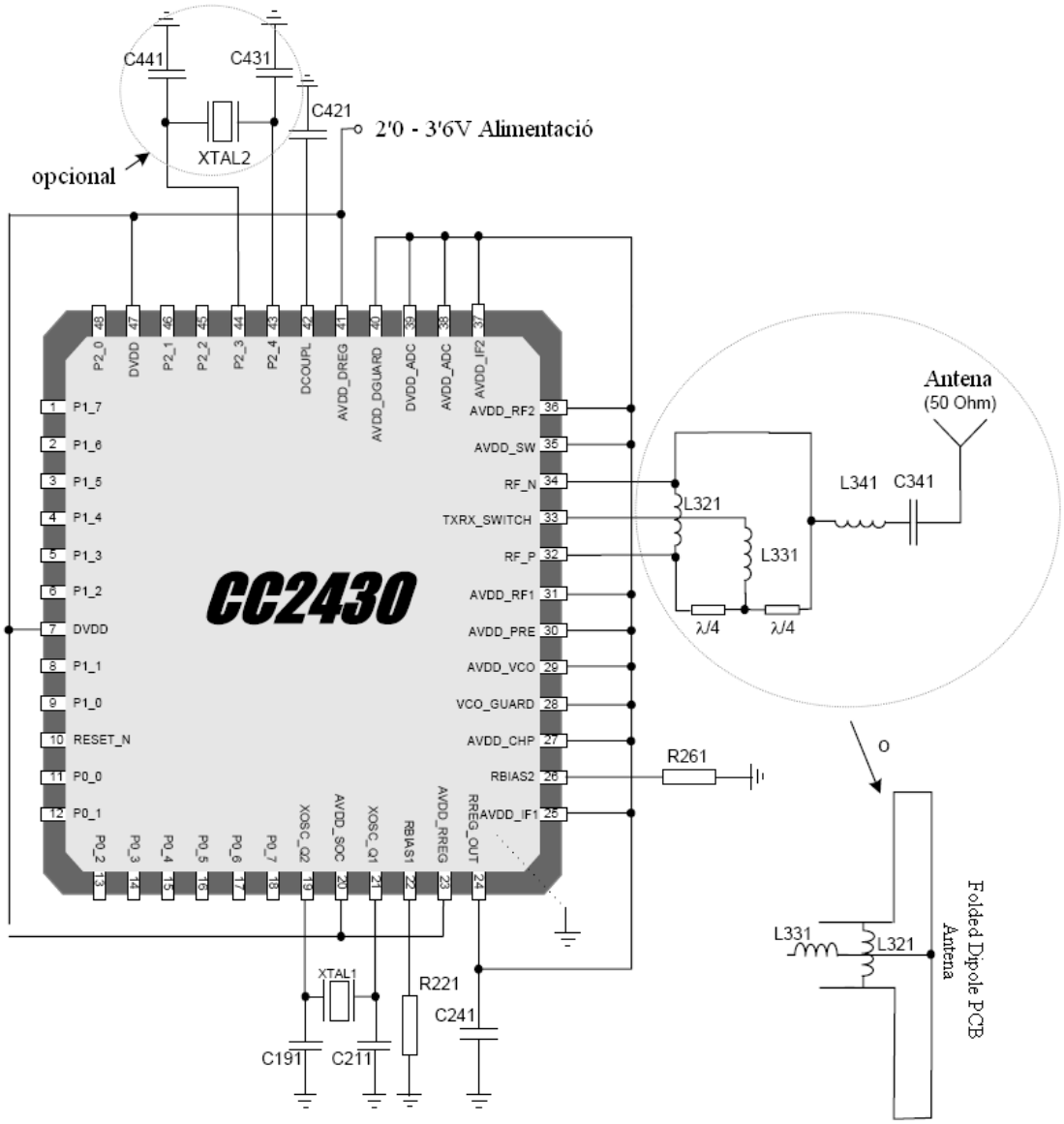


Figura 5.5.1 Circuit aplicat

La taula següent mostra els valors típcs dels components externs:

Component	Descripció	Sortida 50Ω Single Ended	Antena diferencial
C191	Capacitat de càrrega del cristall de 32 MHz	33pF, 5%, NP0, 0402	33pF, 5%, NP0, 0402
C211	Capacitat de càrrega del cristall de 32 MHz	27pF, 5%, NP0, 0402	27pF, 5%, NP0, 0402
C241	Capacitat de càrrega per els reguladors de tensió de l'alimentació analògica	220 nF, 10%, 0402	220nF, 10%, 0402
C421	Capacitat de càrrega per els reguladors de tensió de l'alimentació digital	1µF, 10%, 0402	1µF, 10%, 0402
C341	Bloc DC a antena i match	5'6pF, 5%, NP0,0402	No utilitzat
		1'8pF, Muarata COG 0402, GRM15 1.6nH, Murata 0402, LQG15HS1N6S02	
C431,C441	Capacitat de càrrega del cristall de 32'768 KHz (si es necessita cristall de freq. baixa)	15pF, 5%, NP0, 0402	15pF, 5%, NP0, 0402
L321	Discret balun i match	6'8nH, 5%, monolític/multicapa, 0402	12nH, 5%, monolític/multicapa, 0402
L331	Discret balun i match	22nH, 5%, monolític/multicapa, 0402	27nH, 5%, monolític/multicapa, 0402
L341	Discret balun i match	1'8nH, 5%, monolític/multicapa, 0402	No s'utilitza
R221	Resistència de precisió per el generador de corrent de referència de la part del SoC	56KΩ, 1%, 0402	56KΩ, 1%, 0402
R261	Resistència de precisió per el generador de corrent de referència per la part de RF	43KΩ, 1%, 0402	43KΩ, 1%, 0402
XTAL1	Cristall de 32 MHz	Cristall 32 MHz, ESR<60Ω	Cristall 32 MHz, ESR<60Ω
XTAL2	Cistall de guarda opcional a 32'768 KHz (si es necessita cristall de freq. Baixa)	Cristall 32'768 kHz, Epson MC 306	Cristall 32'768 kHz, Epson MC 306

5.6 Perifèrics

A continuació descriurem alguns dels perifèrics del CC2430 que hem utilitzat per desenvolupar el projecte. Per a una informació més detallada (com per exemple els registres utilitzats per cada perifèric) consultar el data sheet.

5.6.1 Gestió d'energia

Descriurem el controlador de gestió de potència. El controlador de gestió de potència controla l'ús dels modes de potència i el rellotge per aconseguir operar en baixa potència.

5.6.1.1 Gestió d'energia i rellotges

El CC2430 utilitza diferents modes d'operació, o modes de potència, per permetre el funcionament en baixa potència. Operar en ultra baixa potència s'obté apagant l'alimentació dels mòduls per evitar el consum de potència estàtica i també mitjançant clock gating i apagant els oscil·ladors per reduir el consum de potència dinàmica.

Els diferents modes de potència es designen de la següent manera mode potència 0, 1, 2 i 3 (PM0..3).

El PM0 és el mode actiu mentre que el PM3 és el que té menor consum. La manera en la que els diferents modes de potència influeixen en l'operació del sistema és mostra en la següent taula.

Mode potència	Oscil·lador alta freqüència	Oscil·lador baixa freqüència	Regulador tensió (digital)
Configuració	A Cap B 32MHz XOSC C 16MHz RCOSC	A Cap B 32753 kHz RCOSC C 32'768 KHz XOSC	
PM0	B,C	B o C	ON
PM1	A	B o C	ON
PM2	A	B o C	OFF
PM3	A	A	OFF

5.6.1.1.1 PM0

El PM0 és el mode d'operació de tota la funcionalitat on la CPU, perifèrics i el transceptor RF estan actius. El regulador de voltatge digital està en marxa. També fem referència a aquest mode com el mode actiu.

PM0 s'utilitza per a operació normal. També és interessant tenir present que habilitant el bit PCON.IDLE mentre estem en PM0 el nucli de la CPU para d'operar. La resta de perifèrics seguiran funcionant amb normalitat i el nucli de la CPU es despertarà quan es produeixi alguna interrupció.

5.6.1.1.2 PM1

En PM1, els oscil·ladors d'alta freqüència estan apagats (32 MHz XOSC i 16 MHz RCOSC). El regulador de voltatge i l'oscil·lador habilitat de 32 KHz estan actius. Quan s'entra en PM1, s'engega una seqüència d'apagada. Quan el dispositiu surt del PM1 cap a PM0, els oscil·ladors d'alta freqüència s'activen. El dispositiu correrà amb l'oscil·lador RC de 16 MHz fins que el de 32 MHz sigui seleccionat com a font.

5.6.1.1.3 PM2

PM2 té el segon consum de potència més baix. En PM2 el power-on reset, interrupcions externes, l'oscil·lador a 32'768 KHz i els perifèrics de timers de "dormir" estan actius. Els pins I/O retenen la configuració del mode I/O i el valor de sortida que tenien abans d'entrar a PM2. La resta de circuits interns s'apaguen. El regulador de voltatge també s'apaga. Quan s'entra a PM2, s'executa una seqüència d'apagada. Típicament s'entra a PM2 quan s'utilitza el timer de "dormir" com a esdeveniment per despertar, i també combinat amb interrupcions externes. Normalment es selecciona el PM2 en comparació a PM1 quan el temps de dormir excedeix els 3 ms, ja que si el temps de dormir és inferior a 3 ms no es reduiria el consum en comparació amb PM1.

5.6.1.1.4 PM3

PM3 és utilitzat per operar en el mode amb el consum de potència menor. En PM3 tots els circuits interns que s'alimenten a través del regulador de voltatge són pagats (bàsicament tots els mòduls digitals, l'única excepció són la detecció d'interrupció i el mesurador de nivell del power on reset). També s'apaguen el regulador de voltatge intern i tots els oscil·ladors.

El reset i les interrupcions del port d'I/O extern són les úniques funcions que operen en aquest mode. Els pins I/O retenen el mode I/O i el valor de sortida configurats abans d'entrar en PM3. Una condició de reset o un esdeveniment d'interrupció d'I/O extern habilitat despertarà el dispositiu i el posarà en PM0 (una interrupció externa començarà on s'havia entrat a PM3, mentre que un retorn de reset començarà l'execució del programa). El contingut de la RAM i els registres és pràcticament preservat en aquest mode. El PM3 utilitza la mateixa seqüència power down/up que PM2.

El PM3 s'utilitza per aconseguir consum ultra baix de potència quan s'està esperant a un esdeveniment extern.

5.6.1.2 Control de la gestió d'energia

El mode de potència requerit es selecciona mitjançant els bits de mode en el registre de control SLEEP. Una interrupció habilitada des dels pins del port o l'sleep timer o power on reset despertaran el dispositiu d'altres modes de potència i el portaran a PM0 reinicialitzant els bits del mode.

5.6.1.3 Oscil·ladors i rellotges

El CC2430 té un rellotge de sistema intern. La font per el rellotge de sistema tant pot ser un oscil·lador RC a 16 MHz o un oscil·lador de cristall a 32 MHz. El control del rellotge es fa a través del registre SFR CLKCON. El rellotge del sistema també alimenta tots els perifèrics del CC2430.

També hi ha una font de rellotge a 32 KHz que tant pot ser un oscil·lador RC com un oscil·lador de cristall, també controlat per el registre CLKCON.

L'elecció de l'oscil·lador permet obtenir un conveni entre l'alta precisió en el cas del oscil·lador de cristall i el baix consum de potència quan s'utilitza un oscil·lador RC. Tenir en compte que per el funcionament del transceptor RF es requereix l'oscil·lador de cristall de 32 MHz. Aquest esquema mostra una visió general del que és el rellotge del sistema mostrant les fonts de rellotge disponibles.

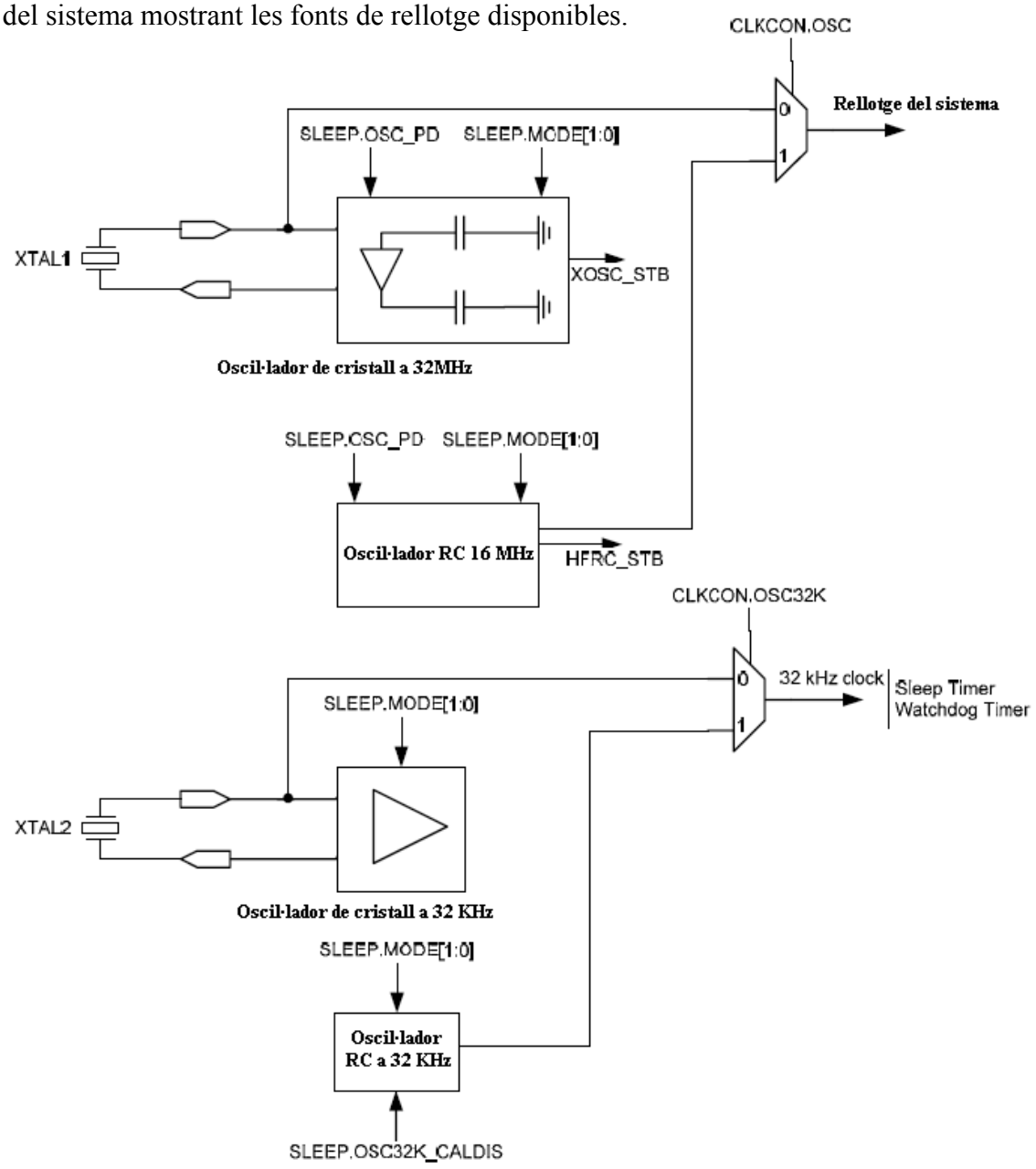


Figura 5.6.1.3.1 Rellotge del sistema

5.6.1.3.1 Oscil·ladors

Hi ha dos oscil·ladors d'alta freqüència en el dispositiu:

- Oscil·lador de cristall a 32 MHz
- Oscil·lador RC a 16 MHz

En la posta en marxa s'utilitza l'oscil·lador RC fins que l'oscil·lador de cristall està estabilitzat. Utilitzant l'oscil·lador RC es consumeix menys però no és suficientment precís per utilitzar el transceptor RF.

Hi ha dos oscil·ladors de baixa freqüència en el dispositiu:

- Oscil·lador de cristall a 32 KHz
- Oscil·lador RC a 32 KHz

L'oscil·lador de cristall està dissenyat per treballar a 32'768 KHz i proporciona un senyal de rellotge estable per sistemes que requereixen precisió temporal. L'oscil·lador RC s'utilitza per reduir cost i consum en comparació amb l'oscil·lador de cristall.

5.6.2 Ports I/O

El CC2430 té 21 pins I/O que poden ser configurats a I/O digitals de propòsit general o com a senyals d'I/O de perifèrics connectats als perifèrics ADC, timers o USART. L'ús dels ports I/O és totalment configurable des del software d'usuari a través d'un conjunt de registres de configuració.

Els ports I/O tenen les següents característiques clau:

- 21 pins d'I/O digitals
- I/O de propòsit general o perifèrics
- Les entrades tenen capacitat de pull-up o pull-down
- Capacitat d'interrupció externa

L'habilitat d'interrupció externa és disponible en tots els 21 pins. Els dispositius externs poden generar interrupcions si ho requereixen. Aquesta característica d'interrupció externa pot ser utilitzada per despertar el dispositiu d'un mode de repòs.

5.6.2.1 Pins I/O no utilitzats

Els pins I/O no utilitzats han de tenir un nivell definit i no s'han de deixar flotant. Una manera de fer això és deixar el pin desconnectat i configurar-lo com a pin d'entrada de propòsit general amb resistència pull-up. Aquest també és l'estat dels pins després d'un reset. Alternativament el pin pot ser configurat com sortida de propòsit general. En cap dels dos casos el pin s'ha de connectar directament a VDD o a GND per tal d'evitar un consum excessiu.

5.6.2.2 Voltatge d'alimentació de les I/O a un nivell baix

En les aplicacions en les que el voltatge d'alimentació de les I/O digitals (pin DVDD) és per sota dels 2'6 V, el bit del registre PICTL.PADSC s'ha d'activar a 1 per obtenir una bona característica DC de sortida.

5.6.2.3 I/O de propòsit general

Quan s'utilitza com a I/O de propòsit general, els pins s'organitzen en tres ports (P0, P1 i P2) de 8-bits. P0 i P1 són ports complerts de 8 bits, mentre que el port P2 només té utilitzables 5 bits (total 21 pins d'I/O). Cada pin de cada port pot ser configurat individualment com a I/O de propòsit general o perifèric.

La intensitat de sortida és de 4 mA en totes les sortides, excepte P1_0 i P1_1 que són de 20 mA.

5.6.2.4 I/O perifèrics

Els pins I/O digitals es poden configurar com a I/O perifèrics. Aquestes són les diferents unitats perifèriques que poden fer d'interfície amb un sistema extern a través dels pins d'I/O digitals: ADC, USART0 SPI, USART0 UART, USART1 SPI, USART1 UART, TIMER 1, TIMER 2, TIMER 3, TIMER 4, 32'768 KHz XOSC, DEPURACIÓ..

S'ha de tenir en compte que com a regle general només dos perifèrics poden ser utilitzats al mateix temps per port d'I/O, per tant s'han de definir prioritats.

5.6.3 16-bits timer, Timer 1

El timer 1 és un timer de 16-bits independent que suporta les típiques funcions de timer/comptador com poden ser captura d'entrada, comparador de sortida i PWM. El timer utilitza un pin I/O per canal. El timer s'utilitza per un ampli rang d'aplicacions de control i mesura i la disponibilitat del mode comptador ascendent/decreixent amb tres canals permet per exemple la implementació d'aplicacions de control de motors.

Les principals característiques del Timer 1 són:

- Tres canals de captura/comparació
- Captura d'entrada de pujada, baixada o qualsevol flang
- Comparador de sortida set, clear o toggle
- Operació de comptador lliure, modulo o up/down
- Preescala de rellotge per dividir per 1,8, 32 o 128
- Per cada captura/comparació i fi de compte es genera una petició d'interrupció
- Funció trigger DMA

5.6.3.1 Comptador timer de 16-bits

El timer consisteix en un comptador de 16-bits que incrementa o decrementa en cada flang actiu de rellotge. El període de flang actiu del rellotge es defineix escrivint en un registre que configura la divisió global del rellotge del sistema donant una freqüència de rellotge variable des de 0'25 MHz a 32 MHz (utilitzant el XOSC de 32 MHz). A part aquest pot ser dividit en el timer 1 mitjançant el valor de preescala. Aquesta preescala pot ser de 1, 8, 32 o 128.

El comptador tant opera corrent lliure, com comptador modulo o comptador up/down per utilitzar en PWM alineat central. El comptador produeix una petició d'interrupció quan s'arriba al valor de terminació de compte (overflow).

5.6.3.2 Mode carrera lliure

En el mode d' operació de carrera lliure el comptador comença a 0x0000 i incrementa a cada flang de rellotge. Quan el comptador arriba a 0xFFFF (overflow) s'activen uns flags, es genera una petició d'interrupció i el comptador es carrega amb 0x0000 i segueix incrementant.

5.6.3.3 Mode modulo

Procedèix com el mode carrera lliure però l'usuari estableix el valor de final de compte.

5.6.3.4 Mode up/down

El mode timer up/down compte repetidament, comença a 0x0000 va incrementant fins el valor especificat, aleshores decrementa fins 0x0000. Aquest mode s'utilitza quan es requereixen polsos de sortida simètrics amb un període diferent a 0xFFFF, així permet l'implementació d'aplicacions de sortida PWM d'alineació centrada. S'activen els flags pertinents i es produeix petició d'interrupció cada cop que s'arriba a 0x0000.

5.6.3.5 Mode captura d'entrada

Quan un canal es configura com a canal de captua d'entrada, el pin I/O amb aquest canal es configura com a entrada. Un cop engegat el timer, un flang de pujada, baixada o qualsevol flang en el pin d'entrada provocarà un trigger de captura del contingut del comptador de 16-bits en el registre de captura associat. Així conseguim capturar el temps en que es produeix un event. Quan es produèix la captura s'activen els flags partinents i la petició d'interrupció.

5.6.3.6 Mode comparador de sortida

En mode comparador de sortida el pin I/O associat amb un canal es configura com a sortida. Un cop el timer ha començat, el contingut del comptador es compara amb el contingut del registre del canal del comparador. Si són iguals el pin de sortida s'activa, reseteja o toggled d'acord amb la configuració. Quan es produèix una comparació s'activaen els flags i la petició d'interrupció associats.

5.6.3.7 Generació de senyal PWM

A continuació explicarem com generar un senyal PWM alineat a flang, i un d'alineació centrada.

- **Alineat a flang:** Els senyals de sortida PWM es poden generar utilitzant el timer en mode modulo i els canals 1 i 2 en mode comparador de sortida 6 o 7. El període del senyal PWM es determina configurant T1CC0 i el duty cycle a T1CCn, on n és el canal PWM 1 o 2.
  També es pot utilitzar el timer en mode cursa-lliure. En aquest cas CLKCON.TICKSPD i el valor del divisor de preescala determinen el període del PWM. La polaritat del PWM es determina seleccionant el mode de comparador (6 o 7).
  Els senyals PWM de sortida també es poden generar utilitzant mode de comparació de sortida 4 i 5, o utilitzant mode modulo. És preferible utilitzar el mode comparador de sortida 4 o 5 per PWM simple.
  Les següents figures mostren la generació d'aquests senyals utilitzant aquestes tènciques. La primera mostra els modes comparació de sortida i el mode timer carrera lliure.

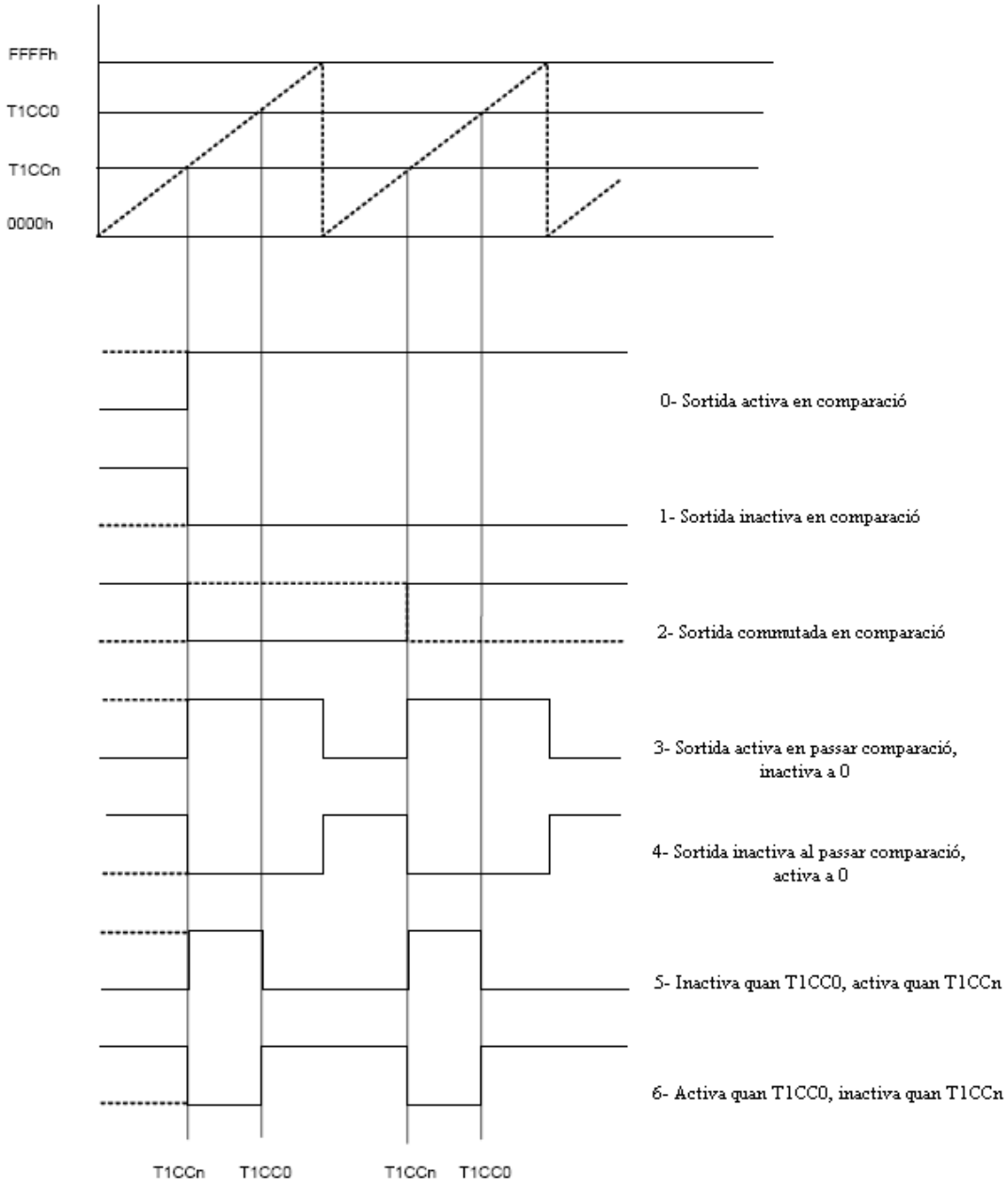


Figura 5.6.3.7.1 PWM modes comparació de sortida i mode timer carrera lliure

Aquesta figura mostra els modes comparació de sortida i el mode modulo timer.

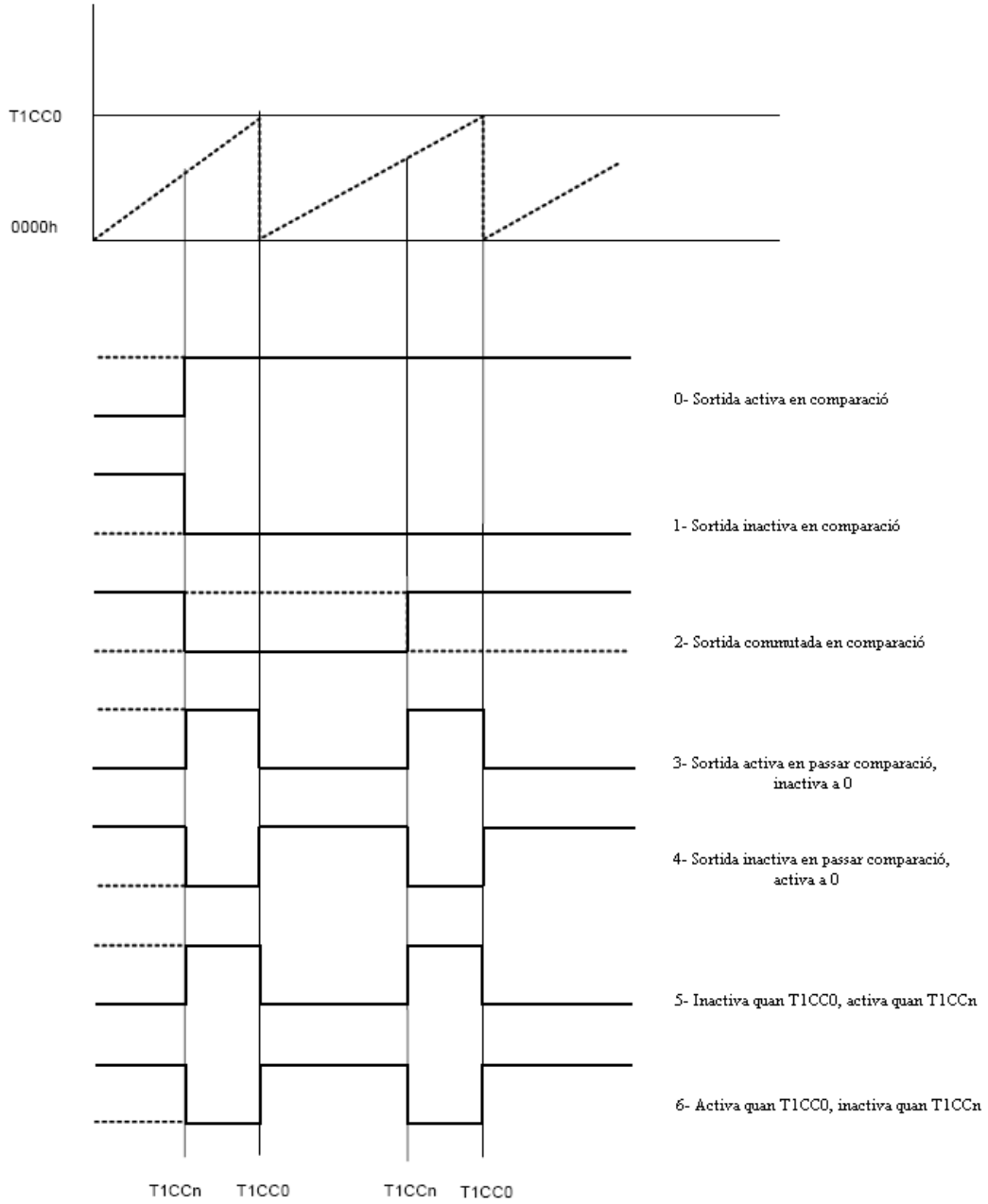


Figura 5.6.3.7.2 PWM modes comparació de sortida i mode modulo timer

- **Alineació centrada:** Les sortides PWM poden ser generades quan es selecciona el mode up/down timer. El mode comparador de canal de sortida 4 o 5 es selecciona depenent en la polaritat del senyal PWM requerida. El període del senyal PWM es determina amb T1CC0 i el duty cycle per el canal de sortida es determina amb T1CCn, on n és el canal PWM 1 o 2.
  El mode PWM d'alineació centrada és requerit per aplicacions de certs tipus de motors i normalment produeix menys soroll que el mode PWM d'alineació a flang ja que les transicions del pin I/O no s'alineen amb el mateix flang de rellotge.
  El següent diagrama temporal mostra els modes de sortida, mode timer up/down.

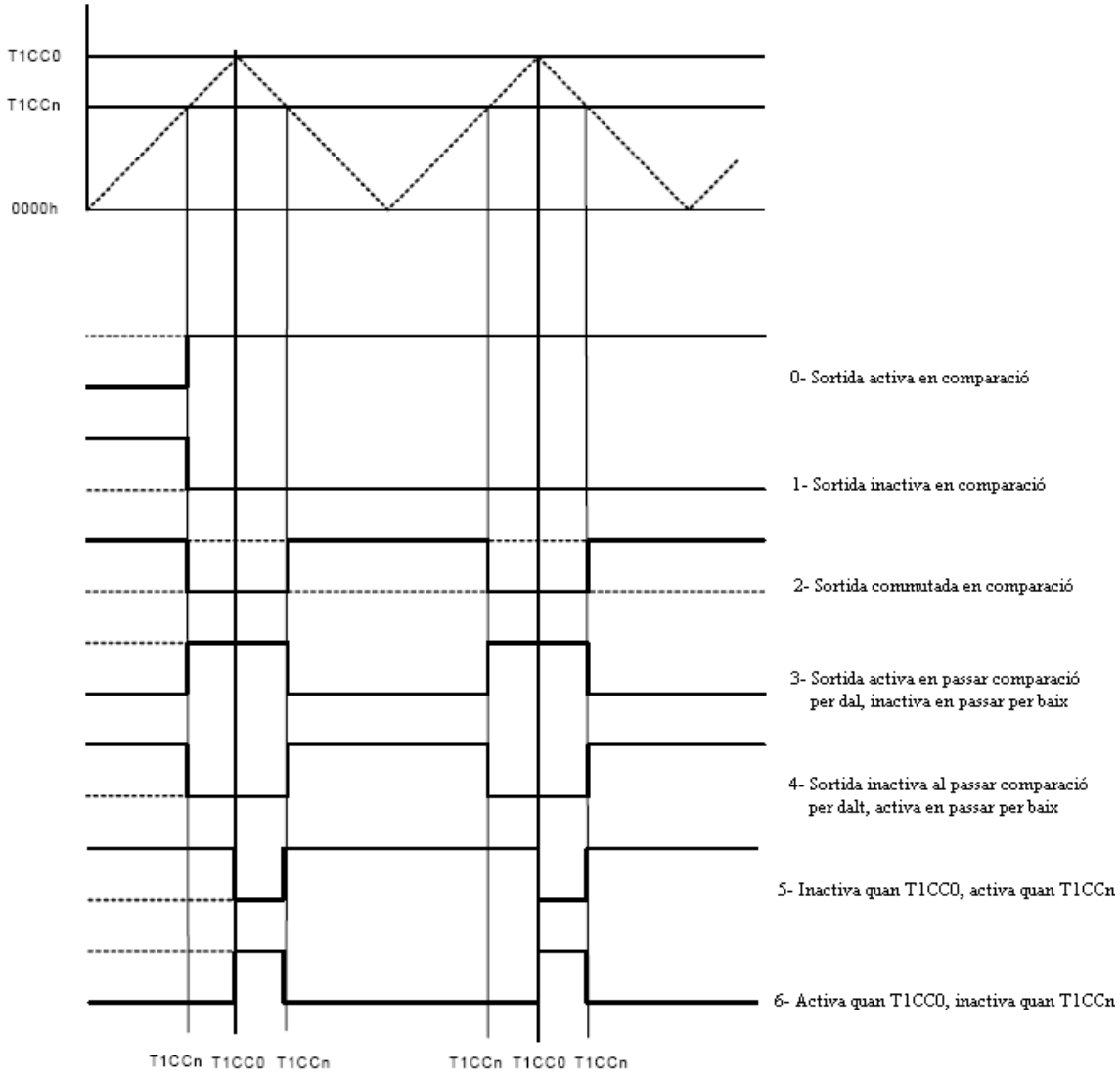


Figura 5.6.3.7.3 PWM modes de sortida, mode timer up/down

5.6.3.8 Interrupcions del Timer 1

Hi ha un vector d'interrupció assignat al timer. Una petició d'interrupció es genera quan es produeix un d'aquests tres esdeveniments:

- El comptador arriba al valor de fi de compte (overflow o torna a zero)
- Esdeveniment de captura d'entrada
- Esdeveniment de comparació de sortida

5.6.4 MAC Timer (Timer 2)

El MAC Timer s'utilitza principalment per proporcionar temporització a l'algorisme CSMA-CA 802.15.4 i cronòmetre general en la capa MAC 802.15.4.

Els mode de funcionament és similar al descrit per Timer 1, si es necessita més detall consultar el data sheet del CC2430.

5.6.5 8-bits Timers, Timer 3 i Timer 4

El timer 3 i el 4 són dos timers de 8-bits que suporten les típiques funcions de timer/comptador com pot ser comparació de sortida i PWM. Els timers tenen dos canals de comparació independents cadascun utilitzant un I/O per canal.

Les característiques dels timers 3/4 són:

- Dos canals de comparació
- Comparació de sortida activa, inactiva i commutada
- Preescala de clock dividida per 1, 2, 4, 8, 16, 32, 64, 128
- Generació de petició d'interrupció en cada esdeveniment de comparació i fi de compte
- Funció trigger DMA

Els modes d'operació dels timers 3 i 4 són: carrera lliure, modulo, up/down, comparació de sortida. Per tant són els mateixos que per el timer 1, i el seu funcionament és igual però en els timers 3 i 4 el timer en si es de 8-bits. L'únic mode que trobem en els timers 3 i 4 que no és present en el timer 1 és el model down que consisteix en que quan el timer s'executa es carrega un valor al comptador que serà decrementat fins arribar a 0, en aquest moment es produirà el flag i la petició d'interrupció associats. Per més detall sobre el funcionament dels timers 3 i 4 consultar el data sheet del CC2430.

5.6.6 Sleep timer

L'sleep timer s'utilitza per configurar el període entre el que el sistema entra i surt dels modes de repòs de baix consum.

L'sleep timer també s'utilitza per mantenir la temporització en el timer 2 (MAC timer) quan entrem en mode repòs de baixa consum.

Les principals característiques de l'sleep timer són:

- Comptador ascendent timer de 24-bits operant a rellotge de 32 KHz
- Comparador de 24-bits
- Operació en mode de baix consum en PM2
- Interrupció i trigger DMA

5.6.6.1 General

L'sleep timer és un timer de 24-bits que corre sobre el rellotge de 32 KHz (tant el RC com el XOSC). El timer comença a corre immediatament després d'un reset i segueix corrent ininterrompudament.

5.6.6.2 Comparació timer

Una comparació timer es produeix quan el valor del timer és igual al valor de 24-bits de comparació. Quan es produeix una comparació de timer s'activa el flag d'interrupció STIF. L'sleep timer funciona en tots els modes de potència exceptuant el PM3. En PM1 i PM2 l'esdeveniment de comparació de l'sleep timer s'utilitza per despertar el dispositiu i retornar-lo a operació activa en PM0.

5.6.7 ADC

5.6.7.1 Introducció

L'ADC suporta conversions ADC de fins a 12-bits. L'ADC un multiplexor analògic amb fins a vuit canals configurables individualment, generador de voltatge de referència i resultats de conversió escrits a memòria a través del DMA sense intervenció de la CPU. Hi ha disponibles varis modes d'operació.

Les principals característiques del ADC són:

- Nombre de decimals seleccionable que també configura la resolució (de 7 a 12 bits).
- 8 canals d'entrada individuals, single ended o diferencials
- Voltatge de referència seleccionable entre intern, extern single ended, extern diferencial o AVDD_SOC
- Generació de petició d'interrupció
- Trigger DMA al final de conversions
- Entrada de sensor de temperatura
- Capacitat de mesura de nivell de bateria

Aquest és el diagrama de blocs de l'ADC:

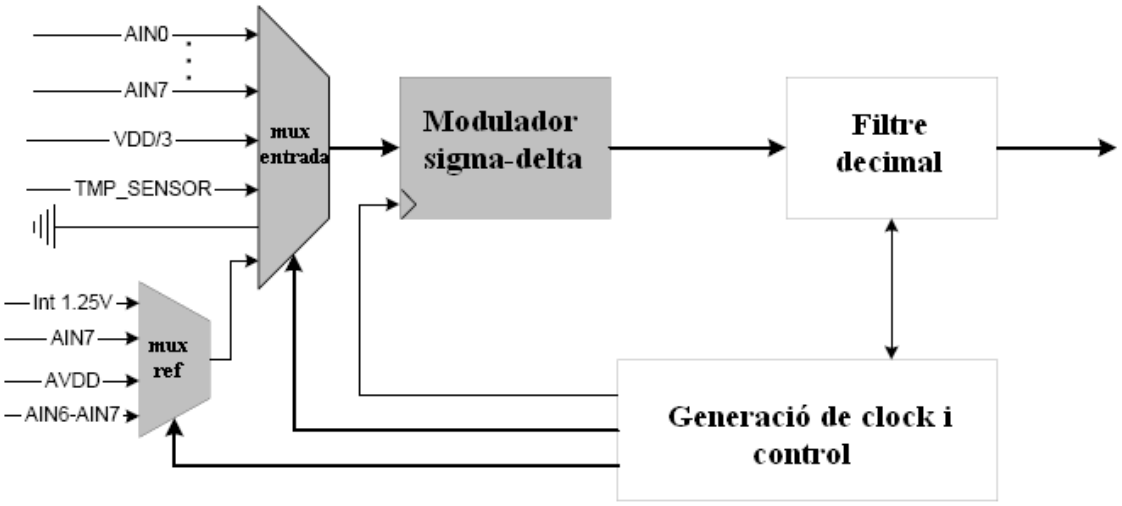


Figura 5.6.7.1.1 Diagrama de blocs de l'ADC

5.6.7.2 Nucli de l'ADC

L'ADC inclou un ADC amb capacitat de convertir una entrada analògica en una representació digital de fins a 12 bits de resolució. L'ADC utilitza un voltatge de referència positiu seleccionable.

5.6.7.3 Entrades ADC

Els senyals en els pins del port P0 poden ser utilitzats com a entrades ADC. A partir d'ara aquests pins del port seran referenciats com a pins AIN0-AIN7. Els pins d'entrada AIN0-AIN7 estan connectats al'ADC. L'ADC pot ser configurat per realitzar automàticament una seqüència de conversions i opcionalment realitzar una conversió externa des de qualsevol canal quan la està completada.

És possible configurar les entrades com a single ended o diferencial. No es poden aplicar tensions negatives ni tampoc tensions superiors a VDD. Addicionalment la sortida d'un sensor de temperatura on-chip pot ser seleccionada com a entrada a l'ADC per mesures de temperatura. També és possible seleccionar un voltatge corresponent a AVDD_SOC/3 com a entrada ADC, això permet implementar un monitor de bateria.

5.6.7.4 Canals

Els canals es troben representats en ADCCON2.SCH. Les entrades single ended de AN0 a AN7 són del canal 0 al 7. Els canals del 8 a l'11 representen les entrades diferencials consistents en AN0-AN1, AN2-AN3, AN4-AN5 i AN6-AN7. Els canals del 12 al 15 representen respectivament GND, referència de voltatge intern, sensor de temperatura i AVDD_SOC/3.

5.6.7.5 Modes d'operació de l'ADC

L'ADC té tres registres, per configurar i per informar de l'estat: ADCCON1, ADCCON2, ADCCON3.

El bit ADCCON1.EOC és un bit d'estatus que està actiu quan s'acaba un conversió i inactiu quan es llegeix l'ADCH.

El bit ADCCON1.ST s'utilitza per començar una seqüència de conversions. La seqüència començarà quan aquest bit és actiu, el ADCCON1.STSEL és 11 i no hi ha cap conversió en curs. Quan la seqüència acaba aquest bit s'inactiva automàticament.

El bit ADCCON1.STSEL selecciona quin event començarà una nova seqüència de conversions. El registre ADCCON2 controla com es realitza la seqüència de conversions. El ADCCON2.SREF s'utilitza per seleccionar el voltatge de referència. Els bits ADCCON2.SDIV seleccionen el nombre de decimals. L'últim canal d'una seqüència s'elegeix amb els bits ADCCON2.SCH.

5.6.8 Coprocessador AES

L'encriptació de dades en el CC2430 es realitza utilitzant un coprocessador dedicat que suporta l'"Advanced Encryption Standard, AES". El coprocessador permet encriptar/desencriptar amb un mínim ús de la CPU.

5.6.8.1 Operació AES

Per encriptar un missatge, s'ha de seguir el següent procés:

- Carregar la clau
- Carregar el vector d'inicialització (IV)
- Descarregar i carregar dades per encriptar/desencriptar

El coprocessador AES treballa amb bolcs de 128 bits, si l'últim bloc en conté menys s'ha d'emplenar amb zeros abans d'escriure'l al coprocessador.

5.6.8.2 Compartir el coprocessador AES entre capes

El coprocessador AES és un recurs compartit per totes les capes. El coprocessador AES només pot ser utilitzat per una instància al mateix temps.

5.6.8.3 Interrupcions AES

L'interrupció AES, ENC, es produeix quan l'encriptació o desencriptació d'un bloc està completa.

5.6.9 USART

USART 0 i USART 1 són interfícies de comunicació sèrie que poden operar separadament tant en mode UART asíncron com mode SPI síncron. Les dues USARTs tenen idèntica funcionalitat, i estan assignades a pins I/O separats.

5.6.9.1 Mode UART

Per interfícies sèrie asíncrones el mode UART és proporcionat. En el mode UART l'interfície utilitza una interfície de dos o quatre fils, consistent en els pins RXD, TXD i opcionalment RTS i CTS. El mode d'operació de l'UART inclou les següents característiques:

- 8 o 9 bits de dades
- Parell, imparell o sense paritat
- Nivell configurable del bit d'start i stop
- Configurable si la primera transferència és LSB o MSB
- Recepció i transmissió d'interrupcions independent
- Recepció i transmissió de DMA triggers independent
- Estat d'error de paritat i framing

El mode UART proporciona transferències asíncrones totalment duplex, i la sincronització dels bits en el receptor no interferèixen amb la funció de transmissió. La transfarència d'un byte d'UART consistèix en un bit d'start, vuit bits de dades, opcionalment un novè bit de dades o paritat, i un o dos bits d'stop. Tenir en compte que les dades transferides es refereix a un byte, tot i que actualment les dades poden consistir en vuit o nou bits.

5.6.9.2 Mode SPI

El mode SPI s'utilitza per operar en comunicacions síncrones. En mode SPI, l'USART es comunica amb un sistema extern a través d'una interfície de 3 o 4 fils. L'interfície consistèix en els pins MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (clock sèrie) i SS_N (Slave Select pin).

El mode SPI inclou les següents característiques:

- Interfície SPI de 3 fils (màster) i 4 fils
- Modes màster i esclau
- Polaritat i fase del SCK configurables
- Primera transferència comfigurable LSB o MSB

5.7 Ràdio

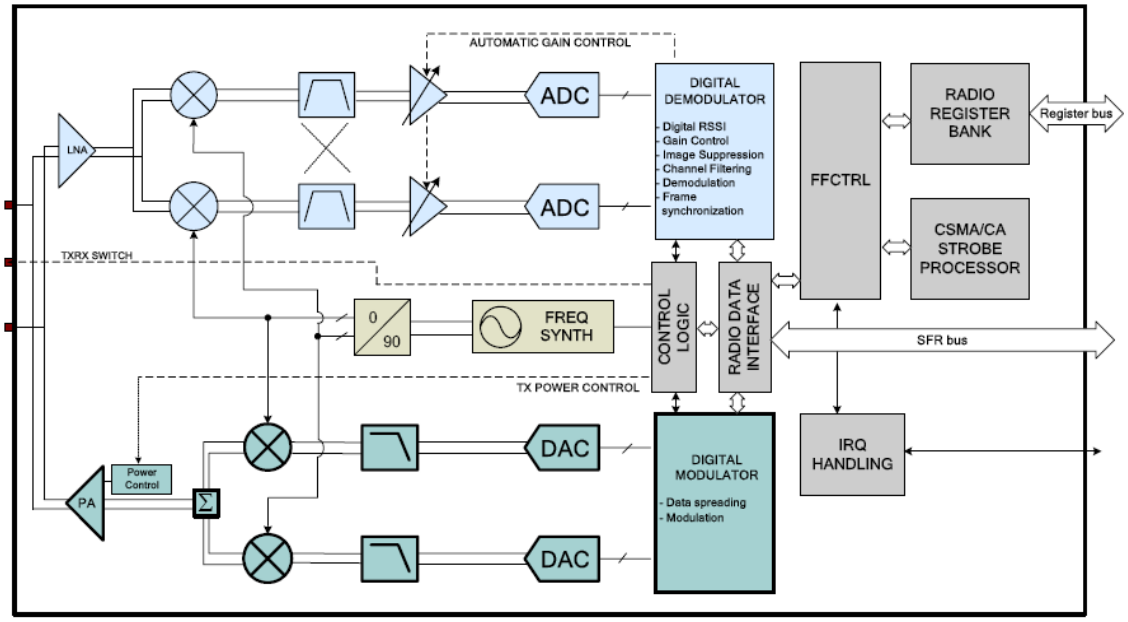


Figura 5.7.1 Diagrama de blocs de la ràdio

La figura superior mostra un diagrama de blocs simplificat d'una ràdio conforme amb l'estàndard IEEE 802.15.4 com la que hi ha dins del CC2430. El nucli de ràdio es basa en el transceptor RF CC2420.

El CC2430 té com a característica un receptor d'IF baixa. El senyal RF rebut és amplificat per el LNA i convertit en quadratura (I i Q) a l'IF. A IF (2 MHz), el senyal I/Q complexa es filtra i s'amplifica, i després es digitalitza per els ADCs del receptor RF. De manera digital es realitza control de guany automàtic, filtrat del canal final, despreading ("des estendre"), correlació de símbol i sincronització de byte.

Una interrupció indica que s'ha detectat un inici de frame de delimitació. El CC2430 bufferitza les dades rebudes en una FIFO de recepció de 128 bytes.

El CRC es verifica en hardware. L'RSSI i la correlació dels valors s'afegeixen al frame. La verificació de canal lliure (CCA), és disponible a través d'una interrupció en mode recepció.

El transmissor del CC2430 es basa en conversions directes. Les dades es bufferitzen en una FIFO (separada de la FIFO de recepció) de transmissió de 128 bytes. El preàmbul i l'inici del delimitador de frame es generen en hardware. Cada símbol (4 bits) és estès utilitzant la seqüència d'extenció de l'IEEE 802.15.4 a 32 chips i sortida als convertidors DACs (digital a analògic).

Un filtre passa baixos analògic passa el senyal als mescladors de conversió de quadratura (I i Q). El senyal RF és amplificat en l'amplificador de potència (PA) i enviat a l'antena.

La circuiteria de l'interruptor intern de transmissió/recepció fa d'interfície amb l'antena. La connexió RF és diferencial. Es pot utilitzar un balun per les antenes single

ended. El biasing del PA i del LNA es fa connectant el TXRX_SWITCH al RF_P i RF_N a través d'un path DC extern.

El sintetitzador de freqüència inclou un complert LC VCO on-chip i un desfasador de 90 graus per generar els senyals I i Q de l'oscil·lador local per els mescladors de conversió en mode recepció i els mescladors de conversió en mode transmissió. El VCO opera en el rang de freqüències 4800 – 4966 MHz, i la freqüència és dividida per dos quan es barreja amb els senyals I i Q.

La bandabase digital inclou suport per tractament de frame, reconeixement d'adreça, bufferitzat de dades, processador CSMA-CA i seguretat MAC.

Un regulador de voltatge on-chip s'encarrega d'entregar una tensió d'alimentació regulada d'1'8 V.

5.7.1 Registres de la ràdio

Els registres de RF s'utilitzen per controlar i saber l'estat de la ràdio. N'hi ha molts amb funcionalitats com control del mòdem, control de transmissió, control de recepció, control de potència de transmissió, adreça IEEE...

5.8 Eines i software

En aquest apartat farem referència a les característiques del Z-Stack (OSAL, API), HAL API, IAR IDE

5.8.1 Z-Stack

El conjunt de paràmetres d'un stack que han de ser configurats a valors específics s'anomena perfil d'stack. Aquests paràmetres que formen part del perfil d'sack estan definits per l'Aliança ZigBee. Tots els dispositius d'una xarxa han de conformar el mateix perfil d'stack.

L'Aliança ZigBee ha definit un perfil d'stack per l'especificació ZigBee-2006 amb l'objectiu de promoure interoperabiltat. Tots els dispositius conformes a aquest perfil d'stack podran treballar en una xarxa amb dispositius d'altres fabricants que també el compleixin.

En l'apèndix 2 "Especificació ZigBee" hem fet una descripció de ZigBee i l'stack. Ara ens centrarem en algunes de les característiques del Z-Stack que és l'implementació de l'Stack ZigBee per Texas Instruments.

5.8.1.1 Z-Stack OSAL

L'OSAL és una API que permet que els components software en el Z-Stack siguin escrits independentment de les especificacions del sistema operatiu, kernel o l'ambient de tasques (incloent loops de control o sistemes connectats a interrupcions). L'OSAL s'implementa en el dispositiu.

Així que el que fa l'OSAL és proporcionar un conjunt de crides a funcions que actuen com a interfície amb els components software del Z-Stack. Proporciona les següents funcionalitats de manera que és independent de l'ambient de processament.

- Enregistrament de tasques, inicialització, posta en marxa
- Intercanvi de missatge entre tasques
- Sincronització de tasques
- Gestió d'interrupcions
- Timers
- Assignació de memòria

5.8.1.2 Z-Stack API

Consisteix en una aplicació que fa d'interfície de programació per tots els components proporcionats dins del Z-Stack. Aquests són els àmbits que cobreix l'API:

- ZDO: La capa ZDO proporciona funcionalitat per gestionar un dispositiu ZigBee. El ZDO API proporciona la interfície per aplicacions d'endpoints per gestionar funcionalitat de Coordinadors ZigBee, Routers o Dispositius Finals que inclou crear, descobrir i afegir-se a una xarxa ZigBee, vincular aplicacions d'endpoints i gestionar seguretat.
- AF: L'interfície AF suporta una interfície a Endpoints (incloent el ZDO) cap a l'stack subjacent. L'AF del Z-Stack proporciona les estructures de dades i les funcions d'ajuda que es requereixen per construir una descripció de dispositiu i és el multiplexor d'endpoint per els missatges d'entrada.
- APS: L'APS API proporciona un conjunt general de serveis de suport que tant són utilitzats per el ZDO com per els objectes d'aplicació definits per el fabricant.
- NWK: La capa NWK ZigBee proporciona gestió i serveis de dades als components de la capa (aplicació) superior.

5.8.1.2.1 ZDO

La capa ZDO proporciona les crides a funcions necessàries per l'implementació de totes les comandes i respostes definides en el Perfil de Dispositiu ZigBee (ZDP), i també altres funcions que habiliten els dispositius a operar com a dispositiu ZigBee.

El ZDP descriu com les característiques d'un dispositiu ZigBee general són implementades dins del ZDO. Defineix descripcions de dispositiu i clústers que utilitzen parelles comanda-resposta. A partir de la definició de missatge en l'estructura de comanda, el ZDP proporciona la següent funcionalitat al ZDO i a les aplicacions:

- Inicialització de dispositiu de xarxa
- Descobriment de dispositius i serveis
- Vinculació d'End Device, i vincular i desvincular serveis
- Servei de gestió de xarxa

El descobriment de dispositiu és el procés per el que un dispositiu ZigBee descobreix un altre dispositiu ZigBee. Descobriment de servei proporciona l'habilitat a un dispositiu de descobrir els serveis oferts per els altres dispositius ZigBee de la PAN. Utilitza varis descriptors per especificar les capacitats del dispositiu.

Vinclament d'end device, vinculació i desvinculació de serveis ofereix a ZigBee les capacitats de vincular. Típicament, la vinculació s'utilitza durant la instal·lació d'una xarxa quan l'usuari necessita vincular dispositius de control amb dispositius controlats. Particularment, la vinculació d'end devices suporta un mètode de vincle simplificat on s'identifiquen parelles de dispositius controlador/controlat.

Els serveis de gestió de xarxa proporcionen l'habilitat de recuperar informació de gestió dels dispositius, incloent resultats de descobriment de xarxa, continguts de taula d'encaminament, qualitat d'enllaç als nodes veïns, i continguts de la taula de vinculació. També proporciona l'habilitat de controlar l'associació de xarxa mitjançant la desassociació de dispositius de la PAN. Els serveis de gestió de la xarxa són dissenyats majoritàriament perquè es gestioni la xarxa per part de l'usuari o per eines encarregades.

5.8.1.2.2 AF

La capa d'AF és l'aplicació d'interfície "per aire" a la capa APS. Conté les funcions i usos d'aplicació per enviar dades per l'aire a través de les capes APS i NWK. Aquesta capa també és el multiplexor d'endpoint per els missatges de dades d'entrada.

L'AF proporciona les següents funcionalitats a les aplicacions:

- Gestió d'endpoint
- Enviar i rebre dades

La gestió d'endpoint fa referència a que cada dispositiu és un node en ZigBee. Cada node té una adreça llarga i curta, l'adreça curta del node s'utilitza per altres nodes per enviar-hi dades. Cada node té 241 endpoints (el 0 està reservat, de l'1 al 240

s'assignen per a l'aplicació). Cada endpoint és adreçable separadament; quan un dispositiu envia dades aquest ha d'especificar l'adreça curta del node destí i l'endpoint que rebrà les dades. Una aplicació ha de registrar com a mínim un endpoint per poder enviar i rebre dades en una xarxa ZigBee.

5.8.1.2.3 APS

L'APS proporciona les següents funcionalitats de gestió accessibles a les caper superiors:

- Gestió de la taula vincles
- Gestió de la taula de grups
- Lookup d'adreces ràpid

A més de les funcions de gestió, l'APS també proporciona serveis a dades que no són accessibles a l'aplicació. Les aplicacions haurien d'enviar les dades a través de l'interfície de dades AF. Per utilitzar les funcions de taula de vincles s'ha d'incloure "BindingTable.h" a l'aplicació.

La taula de vincles de la subcapa APS és una taula definida en la RAM estàtica (però no destinada). La taula de grup de l'APS és una llista d'enllaç definida amb destí a la RAM , a mesura que s'afegeixen grups a la taula la quantitat de d'OSAL utilitzat s'incrementa. Per utilitzar les funcions de la taula de grup s'ha d'incloure "aps_groups.h" a l'aplicació. L'APS proporciona un parell de funcions per realitzar conversions (lookup) d'adreça ràpidament. Aquestes conversions permeten convertir una adreça curta a una IEEE i a l'inversa si la lookup ha estat feta i emmagatzemada en el gestor d'adreces (capa NWK) o si és la teva pròpia adreça.

5.8.1.2.4 NWK

El NWK proporciona les següents funcionalitats accessibles al les capes superiors:

- Gestió de xarxa
- Gestió d'adreça
- Variables de xarxa i funcions d'utilitat

A més a més de les funcions de gestió, el NWK també proporciona serveis de dades que no són accessibles a l'aplicació. Les aplicacions haurien d'enviar les dades a través de l'interfície de dades AF.

El gestor d'adreces proporciona funcions per descobrir routers veïns, crear xarxes, configurar dispositiu com a router, afegir dispositiu a xarxa... El gestor d'adreça proporciona una gestió d'adreça a baix nivell.

5.8.2 HAL API

Aquesta API proporciona la interfície per accedir als timers, GPIO i als ADC. Aquesta API és independent de la plataforma i proporciona un gran conjunt de característiques per a cada servei. No totes les característiques estan disponibles per totes les plataformes.

Aquesta API inclou crides de funcions per:

- Inicialitzar: S'utilitzen per inicialitzar un servei i/o configurar paràmetres opcionals per les dades específiques de la plataforma. Les funcions d'inicialització sovint són cridades en la primera etapa, quan s'alimenta el dispositiu.
- Accés a serveis: Aquestes crides de funcions poden accedir directament als registres del hardware per llegir/escriure certs valors del hardware (per exemple l'ADC o controlar els components del hardware (per exemple els LEDs).
- Callback: Aquestes funcions han de ser implementades per l'aplicació i són utilitzades per passar esdeveniments generats per el hardware (interrupcions, comptadors, timers...) o per mecanisme de polling (UART poll, Timer poll...) cap a la capa superior.

Els controladors del HAL proporcionen serveis de Timer, GPIO, LEDs, interruptors, UART i ADC per la capa MAC i superiors.

5.8.3 IAR Embedded Workbench (EW8051)

L'IAR és una suit d'eines de desenvolupament de software. Aquesta eina suporta gestió de projecte, compilar, assemblar, linking, descarregar i depurar per a varis processadors basats en el 8051, incloent la família CC2430 de Texas Instruments, i dona suport per el Z-Stack. La versió de l'IAR que és compatible amb el Z-Stack és la 7.20H.

Apèndix 6. CC2430DK

En aquest apartat farem un resum dels principals aspectes del kit CC2430DK. Per aprofundir més i obtenir informació addicional consultar la referència [17]

6.1 Introducció

El CC2430DK és una eina de Texas Instruments per desenvolupar aplicacions basades en l'estàndard de xarxes 802.15.4. El kit inclou tot el hardware i el software requerit per avaluar, demostrar, prototipar i desenvolupar varies aplicacions basades en l'estàndard de xarxes 802.15.4.

El CC2430DK és un kit molt flexible que pot ser utilitzat per desenvolupar qualsevol cosa, des de un simple interruptor de llum a nodes avançats amb molts perifèrics. El software inclou una llibreria d'interície a hardware.

6.2 Definicions

SmartRF04EB: Placa d'avaluació. Placa principal amb LCD, interfície USB, LEDs, potenciòmetre... Plataforma per els Mòduls d'avaluació (EM).

Figura 6.2.1 SmartRF04EB

SmartRF04DK: Un terme col·lectiu utilitzat per tots els kits de desenvolupament per la plataforma SmartRF04, per exemple: SmartRF04EB + EM .

CC2430EM: CC2430 Mòdul d'Avaluació, un mòdul petit de plug-in per SmartRF04DK, pot ser utilitzat com a disseny de referència per el layout RF.

Figura 6.2.2 CC2430EM

USB MCU: La MCU C8051F320 de Silicon Labs utilitzat per proporcionar una interfície USB al SmartRF04DK.

Firmware de fàbrica: El firmware que es subministra programat dins del USB MCU. Aquest firmware suporta l'operació SmartRF Studio així com també stand-alone PER tester.

PER: Packet Error Rate. Compte el número de paquets perduts i/o erronis i mostra la relació: (paquets perduts/erronis)/número de paquets enviats.

SoC: Sistema en un xip. Un terme col·lectiu utilitzat per referir-se als circuits integrats de Texas Instruments amb transmissor RF i MCU en el xip. Com és el cas del CC2430.

SoC_DEM: SoC Debug plug-in Module, un petit mòdul plug-in que es pot utilitzar conjuntament amb SmartRF04EB quan programem plaques externes.

Figura 6.2.3 SoC_DEM

ICE: In Circuit Emulator.

6.3 Contingut del kit

6.3.1 Hardware

El kit de desenvolupament conté el següent:

- 2 x SmartRF04EB
- 2 x mòduls d'avaluació (CC2430EM)
- 2 x Antenes de 2'4 GHz
- 2 x Cables USB
- 1 x cable pla de "10-cables" per utilitzar el SmartRF04EB com un emulador per sistemes externs
- 2 x SOC_DEM System on Chip debug plug-in board

Figura 6.3.1.1 Hardware CC2430DK

El kit de desenvolupament SmartRF04DK inclou un nombre de funcions i aplicacions que permeten un test ràpid de l'interície RF i els perifèrics del xip.

- Avaluar els productes SmartRF04. Tant sols traient-lo de la caixa, el kit pot ser utilitzat per testejar el rang.
- Utilitzar el SmartRF Studio per realitzar mesures RF. La ràdio pot ser fàcilment configurada per mesurar sensibilitat, potència de sortida i altres paràmetres RF.
- Desenvolupament de prototips. El SmartRF04DK inclou una interfície USB que pot ser utilitzada com a interfície d'emulació per el CC2430. Tots els ports d' I/O del CC2430 estan disponibles en els pins de connexió de la placa per permetre que aplicacions externes accedeixin fàcilment al CC2430. Aquests connectors també són compatibles amb sondes d'analitzadors lògics per una depuració fàcil.

6.4 SmartRF04EB

Aquest capítol inclou informació aplicada a SmartRF04EB.

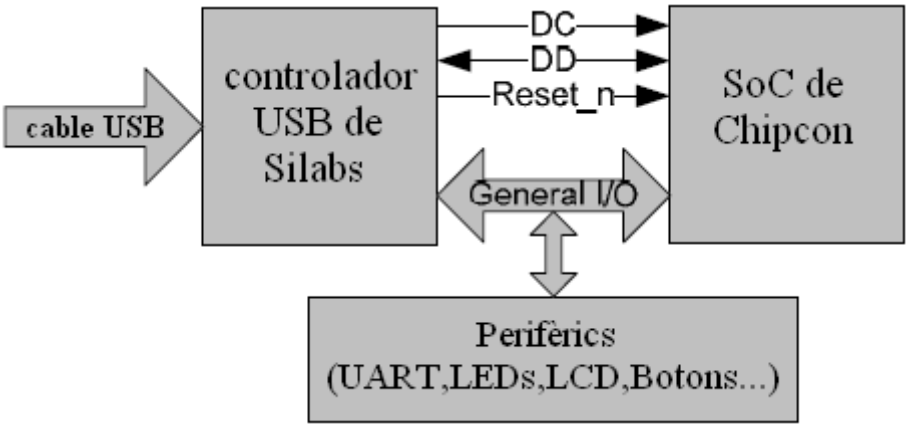


Figura 6.4.1 Principals components de l'SmartRF04EB

La figura de dalt mostra els principals components de l'SmartRF04EB. Un cable USB es connecta directament del PC al controlador USB. Quan es connecta un mòdul SoC d'avaluació a l'SmartRF04EB, l'únic propòsit del controlador USB és traduir els senyals USB a l'interfície SoC de depuració de 2 cables. Aquesta interfície inclou Debug Clock (DC) i Debug Data (DD). El controlador USB també controla la línia Reset_n connectada al SoC.

La majoria dels perifèrics de l'SmartRF04EB estan connectats a ambdós el controlador USB i al SoC. Això significa que els ports I/O dels dos controladors poden accedir al mateix pin. Quan el controlador USB detecta que un SoC està connectat, llavors configurarà totes les I/O en mode de three-state (alta impedància). La connexió al controlador USB d'alguna manera pot influenciar a aquestes línies I/O.

6.4.1 Descripció del hardware de l'SmartRF04EB

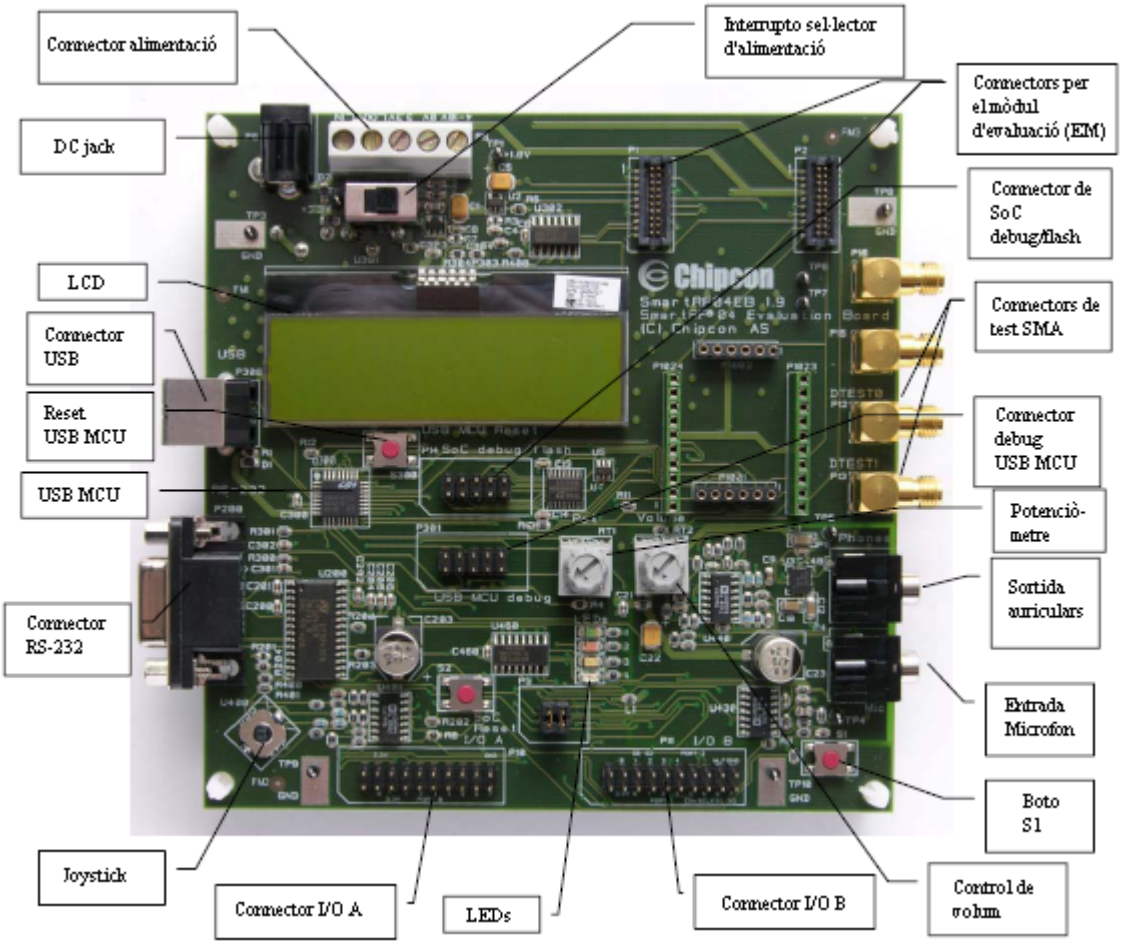


Figura 6.4.1.1 Principals parts de l'SmartRF04EB

La figura de dalt les principals parts de l'SmartRF04EB. La SmartRF04EB serveix com a plataforma principal en el kit de desenvolupament.

6.4.2 Alimentació SmartRF04EB

Si varies fonts d'alimentació estan connectades a la SmartRF04EB, la placa serà alimentada per la font que proporcioni més voltatge. Es pot alimentar de varies maneres: mitjançant DC jack, font d'alimentació de laboratori, alimentat per USB o utilitzant piles.

6.4.3 Interfície USB

La interfície USB s'utilitza per fer d'interfície amb el PC per utilitzar SmartRF Studio i per programar i depurar.

6.4.4 Interfície RS-232

El RS-232 pot ser utilitzat per aplicacions personalitzades per comunicar amb altres dispositius. L'interfície RS-232 utilitza un dispositiu convertidor de voltatge de manera que el port RS-232 és compatible amb els nivells bipolar RS-232.

6.4.5 Interfície d'usuari

L'SmartRF04EB inclou un joysick i un botó de pulsació com a dispositius d'entrada d'usuari, i quatre LEDs i un display LCD de 2x16 caràcters com a dispositius de sortida d'usuari.

6.4.6 Connectors I/O

Els connectors I/O porten a fora tots els senyals dels connectors EM. Aquests connectors són compatibles amb sondes d'analitzadors lògics. Els connectors permeten un accés fàcil als senyals I/O i connectar plaques de prototipatge.

Pin	Funció
1	N/C
2	N/C
3	P0_0/MIC_IN
4	VDD
5	VDD
6	N/C
7	P0_1/BUTTON_PUSH
8	N/C
9	P0_2/UART_RD
10	N/C
11	P0_3/UART_TD
12	N/C
13	P0_4/RTS
14	N/C
15	P0_5/JOY_PUSH
16	N/C
17	P0_6/JOY
18	N/C

Pin	Funció
1	N/C
2	N/C
3	VDD
4	P2_0/LED4
5	P1_0/LED1
6	P2_1/DD
7	P1_1/PWM_OUTPUT
8	P2_2/DC
9	P1_2/LED2
10	P2_3/SDA
11	P1_3/LED3
12	P2_4/SCL
13	P1_4/CSn
14	N/C
15	P1_5/SCLK
16	RESET_N
17	P1_6/MOSI
18	Debug Data Direction (DD_DIR)

19	P0_7/POT
20	GND

Descripció dels pins del connector A d'I/O (P10)

19	P1_7/MISO
20	GND

Descripció dels pins del connector B d'I/O (P11)

6.4.7 Connectors EM

Els connectors EM són utilitzats per connectar l'EM a l'SmartRF04EB. Els connectors P1 i P2 són utilitzats com a principal interfície.

L'EM pot ser utilitzat com a disseny de referència de RF amb el SoC de Chipcon, decoplament, i tota la circuiteria RF necessària. Es recomana copiar aquest disseny de referència quan es dissenyen aplicacions amb dispositius RF de chipcon per tal d'aconseguir el millor comportament en RF.

6.4.8 Flux de dades

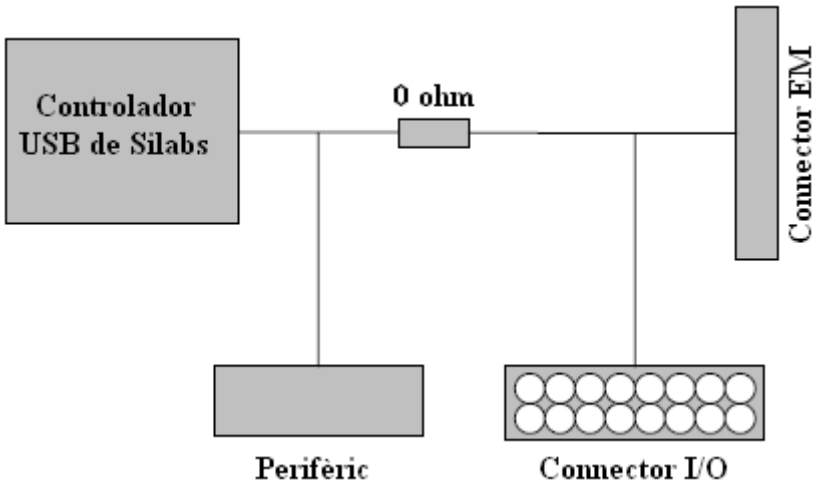


Figura 6.4.8.1 Flux de dades

Les línies de senyal del connector EM corren via resistències de 0-ohm cap el USB MCU i els diferents perifèrics que hi ha en el SmartRF04EB. Això permet connectar un mòdul EM a altres aplicacions. El USB MCU pot ser desconnectat dels pins de senyal traient les resistències de 0-ohm. Els connectors I/O es troben fora de les resistències 0-ohm, així que segueixen connectats als connectors EM fins i tot si es treuen les resistències de 0-ohm.

6.4.9 Connexió de l'EM LCD

Els pins de I/O P2_3 i P2_4 del SoC estan connectats a un oscil·lador-X a 32 KHz en les plaques CC2430EM. Per tant internament el displa LCD, en el EM, intercambia el P1_2 per P2_3 i el P2_0 per P2_4 com mostra la següent figura. Per tant no hi ha accés al LED2 (vermell) al LED4 (blau) del SoC.

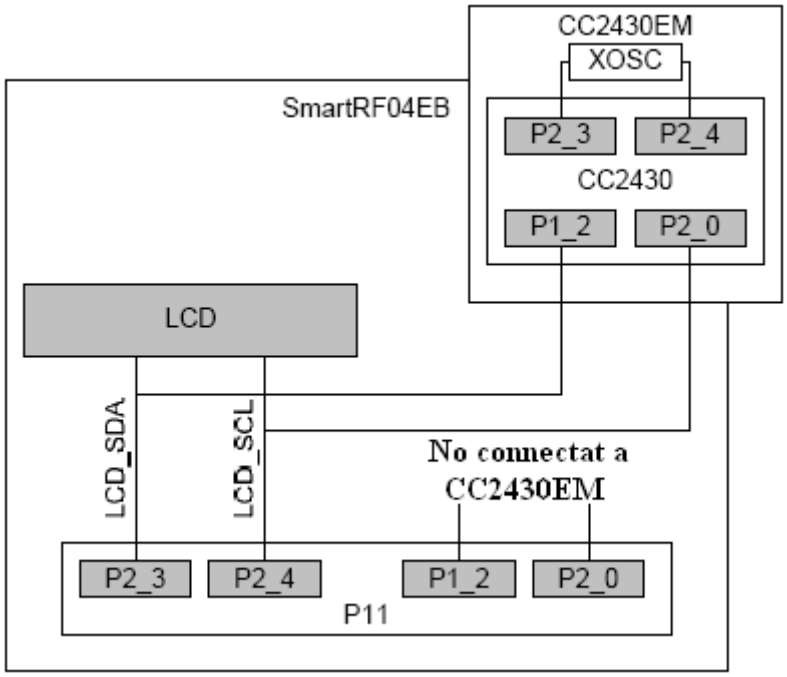


Figura 6.4.9.1 Connexió de l'EM LCD

6.5 Utilitzar l'SmartRF04EB per prototipar

L'SmartRF04EB inclou una interfície per depurar i programar. L'interfície de depuració està controlada per dos pins de comunicació. En l'SartRF04EB l'interfície és controlada per l'USB MCU. Això permet les dos interfícies, programació i emulació utilitzant el port USB.

Les fileres de pins P10 i P11 es poden utilitzar per connectar un altre PCB o una placa de prototipatge a l'SmartRF04EB.

6.5.1 Utilitzar l'SmartRF04EB com a In-Circuit Emulator (ICE)

L'SmartRF04EB pot ser utilitzat com un ICE per els dispositius connectats, ja siguin en el socket EM o un sistema extern amb aplicacions a mida. La següent figura mostra els principals components de l'SmartRF04EB quan s'utilitza com a ICE. S'ha de tenir en consideració que quan es s'utilitza l'SmartRF04EB per depurar un sistema extern, s'ha de treure l'EM, i viceversa. És molt recomanat utilitzar el SOC_DEM en el connector EM quan l'SmartRF04EB s'utilitza per depurar un sistema extern. La següent figura mostra l'SmartRF04EB utilitzat com a ICE.

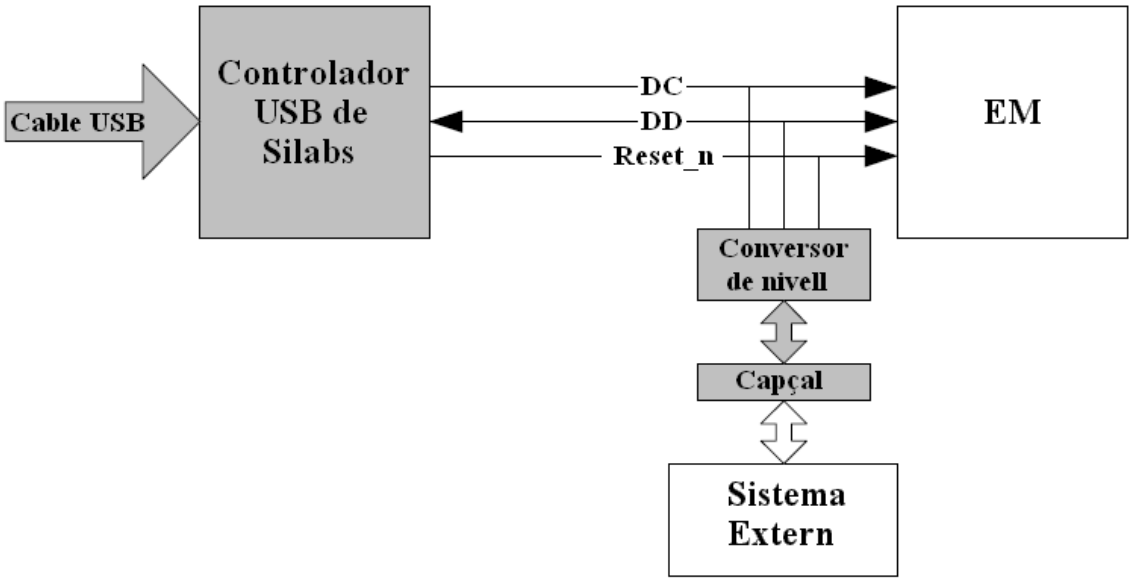


Figura 6.5.1.1 SmartRF04EB utilitzat com a ICE

6.5.2 Interfície de depuració

Quan es dissenyen aplicacions amb els SoC de Chipcon es recomana incloure un capçal de pin o punts de test per permetre in-circuit emulation o programar utilitzant l'SmartRF04EB o altres eines de programació.

En la taula de més abaix s'expliquen els pins utilitzats en l'SmartRF04EB. El connector inclou 4 senyals de control SPI, que no s'utilitzen però estan inclosos per donar més flexibilitat.

L'SmartRF04EB inclou un convertidor de voltatge per donar suport a la programació i depuració de sistemes externs amb voltatge d'operació diferent de l'SmartRF04EB. El connector de depuració (P14, "SoC debug/flash") inclou dos connexions VDD, en els pins 2 i 9. La funció és diferent per aquestes connexions.

El pin 2 de VDD dona voltatge d'alimentació al convertidor de voltatge. Es recomana connectar aquest pin a VDD, a la placa externa per assegurar-nos que s'està utilitzant el voltatge d'alimentació correcte per el convertidor de voltatge. Aquest pin ha d'estar sempre connectat.

El pin 9 de VDD dona alimentació VDD (3.3 V) de l'SmartRF04EB. Si el dispositiu d'ap licació s'alimenta a través de l'SmartRF04EB durant la programació i la depuració aquest pin s'ha de connectar. Si el voltatge del dispositiu no és de 3.3 V, no s'hauria de connectar aquest pin.

Els pins utilitzats per l'SmartRF04EB s'expliquen en la següent taula i figura. Tots els senyals en negreta són necessaris. La figura mostra el layout d'un connector amb els mínims senyals requerits.

Pin	Funció	Notes
1	**GND**	
2	**VDD**	**Utilitzat per configurar el voltatge correcte per el convertidor de nivell de voltatge**
3	**Debug Clock (DC)**	
4	**Debug Data (DD)**	
5	CSn	
6	SCLK	
7	**Reset_n**	
8	MOSI	
9	3.3V VDD, alt. NC	Entrega VDD de l'SmartRF04EB
10	MISO	

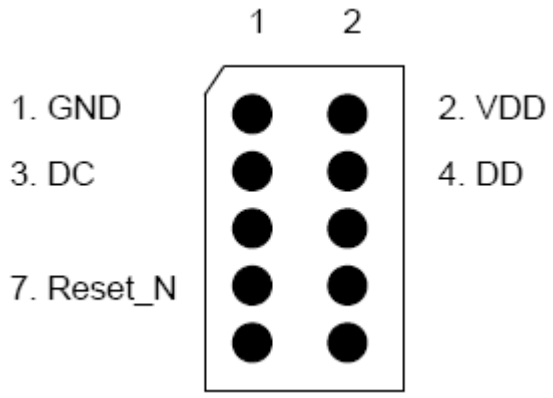


Figura 6.5.2.1 Connector

6.5.3 Placa plug-in de depuració de SoC (SOC_DEM)

El CC2430 DK està equipat amb dues plaques plug-in de depuració de SoC (SOC_DEM) com es mostra en la figura següent.

Figura 6.5.3.1 Plug-in SoC_DEM

El SOC_DEM proporciona una interfície SoC depuració/programació física addicional. Aquestes plaques s'inclouen perquè se sap que la interfície original de depuració de l'SmartRF04EB P14 en situacions concretes no és de confiança.

Quan programem i/o depurem una placa utilitzant l'SmartRF04EB com a ICE es recomana d'utilitzar sempre la placa SOC_DEM en comptes del connector SoC debug/flash de l'SmartRF04EB.

Apèndix 7. Planificació del treball i valoració econòmica

7.1 Planificació del treball

Aquesta és la planificació realitzada al principi del projecte:

- Estudi i elecció dels estàndards / protocols existents al mercat: 2 setmanes
- Estudi i elecció de les plataformes existents en el mercat per l'estàndard / protocol elegit: 2 setmanes
- Estudi a fons de l'estàndard elegit: 3 setmanes
- Estudi del funcionament de la plataforma (hardware, stack, eines de programació...): 4 setmanes
- Escriure el codi, revisar codi, muntar aplicació... : 5 setmanes
- Redactar la memòria, resultats, conclusions, preparar presentació...: 11 setmanes

El següent esquema mostra la planificació per setmanes de manera gràfica:

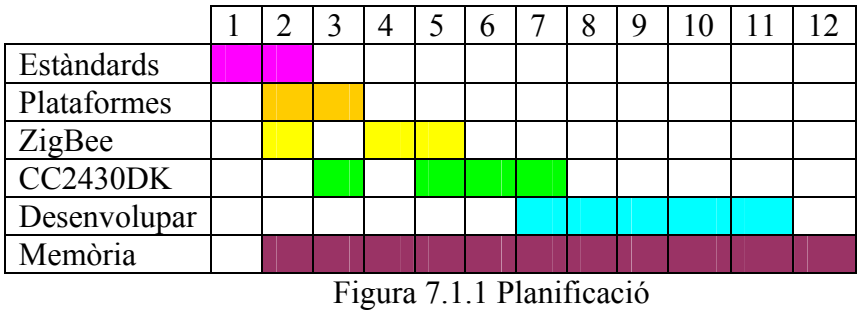


Figura 7.1.1 Planificació

Com s'observa, hi ha tasques que s'entra creuen i que es duen a terme en paral·lel. Mentre realitzem l'estudi dels estàndards per comunicacions sense fils existents al mercat és inevitable indagar quines solucions físiques existeixen, ja que també és un dels factors a tenir en compte alhora d'elegir amb quin estàndard ens quedarem, de la mateixa manera, quan es decideix quin estàndard adoptarem ja s'està fent un estudi més a fons d'aquell estàndard. El mateix passa durant l'elecció de la plataforma. Mentre s'estudien les plataformes existents, és a dir, mentre mirem les especificacions dels hardwares que implementen l'estàndard adoptat, estem valorant la plataforma que utilitzarem i col·lateralment també estem recollint informació sobre el mateix estàndard. Quan s'està estudiant a fons ZigBee és inevitable donar un cop d'ull a com és duen a terme tot aquest conjunt de protocols sobre la plataforma que hem

adquirit, i mentre estudiem la plataforma es van fent proves sobre el hardware per comprovar que s'assimila la informació extreta dels manuals i veure com es realitza en la pràctica. Mentre s'està fent tot això, paral·lelament es redacta tota la documentació, però aquesta documentació no es pot acabar fins que es donen per finalitzades totes les altres tasques, ja que dins de la memòria hi han d'aparèixer els resultats i conclusions estretes de tot el treball, i com no, preparar la presentació com a resum del projecte.

Aquesta planificació s'ha seguit al peu de la lletra, l'únic problema que hi ha hagut és que el període d'entrega del CC2430DK va ser de 3 setmanes, per tant, des de que seleccionem quina plataforma utilitzarem fins que la varem tenir a les mans, van passar 3 setmanes. Mentre estant es va seguir amb la planificació establerta (llegir la literatura sobre l'stack ZigBee, manual de funcionament de la plataforma de Texas Instruments, etc...)

7.2 Valoració econòmica

7.2.1 Hores invertides

El temps de desenvolupament del projecte (des de que es va començar, fins al redactat de la memòria) ha estat de 12 setmanes (aproximadament), durant els quals s'han realitzat les tasques descrites en l'apartat anterior. La valoració de les hores invertides fa de mal calcular ja que la dedicació no ha estat exclusiva a aquesta tasca. Comptant una dedicació de 8 hores diàries de dilluns a divendres a un rendiment del 40% (percentatge d'hores invertides en aquest projecte sobre les hores totals) durant dotze setmanes, en total s'han invertit 192 hores. Tenint en compte que es partia d'una base de coneixements sobre el tema de zero, tampoc és una inversió excessiva tenint en compte tot el que s'ha realitzat.

7.2.2 Preu material

Per desenvolupar el projecte hem adquirit el kit CC2430DK de Texas Instruments que té un preu de 720 €. El preu unitari per xip CC2430 és de 11 €, i els components addicionals necessaris per a fer funcionar el CC2430 (veure apèndix 5 "CC2430", concretament l'apartat "5.5 Circuit aplicat") són mínims.

7.2.3 Conclusions

Tant el temps invertit com el cost material no són cap exageració. A més a més, la plataforma de desenvolupament presenta gran facilitat en la manipulació del codi, tant per fer petites modificacions com per generar projectes totalment nous, és molt versàtil i permet crear implementacions ZigBee en un període de temps curt. Tot això fa que el projecte sigui totalment viable i es pugui tenir un preu de mercat competitiu.

Apèndix 8. Premisses, programació i algorismes comentats

En aquest apèndix es recullen tots els aspectes que fan referència al programa, tals com premisses que hem tingut en compte alhora de programar (com pot ser la seguretat), proves que hem fet, ets...

8.1 Introducció

L'eina de desenvolupament de programa utilitzada és l'IAR API [18], i el llenguatge de programació és C. També s'utilitzen altres softwares addicionals com és el Z-Stack OSAL API [16], Z-Stack API [16] i el HAL API [10]. Es recomana consultar la documentació d'aquestes eines, sobretot la del software addicional, ja que en el codi utilitzem les funcions descrites per aquestes APIs , i per tant, aquí no seran definides totes.

El Z-Stack OSAL API dona suport per:

- Registre de tasques, inicialització, inici
- Intercanvi de missatges entre tasques
- Sincronització de tasques
- Gestió d'interrupcions
- Timers
- Gestió de memòria

El Z-Stack API dona suport per:

- Gestionar la funcionalitat dels dispositius ZigBee (coordinadors, routers, o dispositius finals)
- Creació, descobriment i afegir-se a xarxes
- Enllaçar punts finals
- Gestió de la seguretat
- Estructures de dades i funcions útils per a construir descripció de dispositiu (és el multiplexor de punt final per quan arriben missatges)
- Un conjunt general de serveis de suport per als objectes d'aplicació
- Gestió i servei de dades

El HAL API dona suport per accedir a:

- Timers
- GPIO
- USART
- ADC

8.2 Premisses

Ara que ja tenim definides les eines que utilitzarem per desenvolupar el codi definirem certes premisses a complir per tal de garantir un bon funcionament del codi, seguretat, fiabilitat de les dades...

8.2.1 Confirmació d'enviament de missatge

Cada cop que enviem un missatge hem de confirmar si el missatge s'ha pogut enviar, si no ha pogut ser ho reintentarem 5 vegades, si passats els 5 intents no s'ha aconseguit enviar el missatge s'avisarà a l'usuari.

8.2.2 Confirmació de recepció de missatge

Quan enviem un missatge esperem rebre una confirmació de recepció per part del dispositiu destí. Si no la rebem reenviarem el missatge fins a 5 vegades, en el cas que no s'aconsegueixi la recepció avisarem a l'usuari.

8.2.3 ADC

Quan treballem amb el ADC tindrem en compte que abans de poder llegir els registres que contenen el resultat de la conversió hem d'esperar que el registre d'estat ens confirmi que la conversió ha estat realitzada.

8.3 Programació i algorismes comentats

En aquest apartat mostrarem el codi que hem escrit i el comentarem. Evidentment, compleix amb les premisses exposades en l'apartat anterior.

El codi que escrit genera quatre modes d'operació diferents. Els modes de treball es seleccionen a través del joystick que incorpora com a perifèric l'SmartRF04EB. Aquests són els modes de treball:

- Amunt: Cada cop que premem el joystick en direcció ascendent incrementem el cicle de treball del PWM en 100 unitats, comprovem que no es desbordi el registre, i l'enviem.
- Avall: Cada cop que premem el joystick en direcció descendent restem 100 unitats al cicle de treball del PWM, comprovem que no es desbordi el registre, i l'enviem.

- Dreta: Cada cop que premem el joystick en direcció dreta realitzem una petició de lectura del sensor de pressió. Com que no tenim sensors, el que s'ha implementat és una lectura d'un port analògic on podem tenir-hi qualsevol valor.
- Esquerra: Quan premem el joystick en direcció esquerra executem una interrupció per timer cada 0'3 segons que provoca una lectura d'un port analògic on tenim connectat un potenciòmetre. Es captura el valor del potenciòmetre i s'envia com a cicle de treball d'un senyal PWM.

A continuació es mostra el codi:

8.3.1 XSensors.h

No hi ha gran cosa a explicar d'aquesta part del codi. Bàsicament són definicions, com l'identificador de dispositiu, el número de punt final, el nombre de clústers i els clústers, el valor de timer per generar l'esdeveniment cada 0'3 segons, etc...

```
#ifndef XSensors_H
#define XSensors_H

/*********************************************************************
   Filename:       XSensors.h
   Revised:        $Data: 2008-06-13  $
   Revision:       $ $

   Description:

      Aquest fitxer conté les definicions de l'aplicació.


*********************************************************************/

#ifdef __cplusplus
extern "C"
{
#endif

/*********************************************************************
 * INCLUDES
 */
#include "ZComDef.h"

/*********************************************************************
 * CONSTANTS
 */

#define XSensors_ENDPOINT           10

#define XSensors_PROFID             0x0F04
#define XSensors_DEVICEID           0x0001
#define XSensors_DEVICE_VERSION     0
```

```
#define XSensors_FLAGS             0

#define XSensors_MAX_CLUSTERS      10
#define XSensors_request_temperatura_CLUSTERID    1
#define XSensors_receive_temperatura_CLUSTERID    2
#define XSensors_request_pressio_CLUSTERID        3
#define XSensors_receive_pressio_CLUSTERID        4
#define XSensors_request_caudal_CLUSTERID         5
#define XSensors_receive_caudal_CLUSTERID         6
#define XSensors_request_revolucions_CLUSTERID    7
#define XSensors_receive_revolucions_CLUSTERID    8
#define XSensors_request_tot_CLUSTERID            9
#define XSensors_send_PWM_CLUSTERID               10


// Timeout d'enviament del cicle de treball del PWM
#define XSensors_SEND_PWM_TIMEOUT    300      // Això en el timer implica 0'3
segons

// Application Events (OSAL) - These are bit weighted definitions.
#define XSensors_SEND_MSG_EVT       0x0001
#define XSensors_SEND_PWM_EVT       0x0002

/*********************************************************************
 * MACROS
 */

/*********************************************************************
 * FUNCTIONS
 */

/*
 * Task Initialization for the Generic Application
 */
extern void XSensors_Init( byte task_id );

/*
 * Task Event Processor for the Generic Application
 */
extern UINT16 XSensors_ProcessEvent( byte task_id, UINT16 events );

/*********************************************************************
*********************************************************************/

#ifdef __cplusplus
}
#endif

#endif /* XSensors_H */
```

8.3.2 XSensors.c

Aquest codi conté tot el que és l'aplicació, les funcions, variables, etc...

```
/*********************************************************************
  Filename:       XSensors.c
  Revised:        $Date Data: 2008-06-13 $
  Revision:       $ $

  Description:
                                  - Aplicació.



        Key control:
          SW1: incrementem cicle de treball PWM
          SW2:  fem petició de pressió
          SW3: reduïm cicle de treball PWM
          SW4:  lectura potenciòmetre i enviament cicle de treball PWM cada 0'3 segons

  Notes:


*********************************************************************/

/*********************************************************************
 * INCLUDES
 */
#include "OSAL.h"
#include "AF.h"
#include "ZDApp.h"

#include "XSensors.h"
#include "DebugTrace.h"

#if !defined( WIN32 )
  #include "OnBoard.h"
#endif

/* HAL */
#include "hal_lcd.h"
#include "hal_led.h"
#include "hal_key.h"
#include "hal_uart.h"
#include "hal_adc.h"

/*********************************************************************
 * MACROS
 */
```

```
/*********************************************************************
 * CONSTANTS
 */

/*********************************************************************
 * TYPEDEFS
 */

/*********************************************************************
 * VARIABLES GLOBALS
 */

// Aquesta llista s'ha d'omplir amb els identificadors de clústers de l'aplicació
const cId_t XSensors_ClusterList[XSensors_MAX_CLUSTERS] =
{
  XSensors_request_temperatura_CLUSTERID,
  XSensors_receive_temperatura_CLUSTERID,
  XSensors_request_pressio_CLUSTERID,
  XSensors_receive_pressio_CLUSTERID,
  XSensors_request_caudal_CLUSTERID,
  XSensors_receive_caudal_CLUSTERID,
  XSensors_request_revolucions_CLUSTERID,
  XSensors_receive_revolucions_CLUSTERID,
  XSensors_request_tot_CLUSTERID,
  XSensors_send_PWM_CLUSTERID

};

const SimpleDescriptionFormat_t XSensors_SimpleDesc =
{
  XSensors_ENDPOINT,            //  int Endpoint;
  XSensors_PROFID,              //  uint16 AppProfId[2];
  XSensors_DEVICEID,            //  uint16 AppDeviceId[2];
  XSensors_DEVICE_VERSION,      //  int   AppDevVer:4;
  XSensors_FLAGS,               //  int   AppFlags:4;
  XSensors_MAX_CLUSTERS,        //  byte  AppNumInClusters;
  (cId_t *)XSensors_ClusterList,  //  byte *pAppInClusterList;
  XSensors_MAX_CLUSTERS,        //  byte  AppNumInClusters;
  (cId_t *)XSensors_ClusterList   //  byte *pAppInClusterList;
};

//  Aquesta és la descripció del punt final/interfície. Es defineix aiquí, però s'omple
// en XSensors_Init().
endPointDesc_t XSensors_epDesc;

/*********************************************************************
 * EXTERNAL VARIABLES
 */

/*********************************************************************
```

```
 * EXTERNAL FUNCTIONS
 */

/*********************************************************************
 * VARIABLES LOCALS
 */
byte XSensors_TaskID;   // Task ID per processat d'esdeveniments/tasques internes
                        // This variable will be received when
                        // XSensors_Init() is called.
devStates_t XSensors_NwkState;


byte XSensors_TransID;  // This is the unique message ID (counter)

afAddrType_t XSensors_DstAddr;


//Per processar dades
uint8 header; //Quan rebem missatges guardem aquí el byte d'informació addicional
uint16 dades_process; //Quan rebem missatges guardem aquí els bytes alt i baix
uint8 dades_request[1]; //Byte que enviem quan fem peticions
uint8 dades_send[3]; //On posem les dades que s'han d'enviar
uint16 PWM_duty_cycle = 0; //El cicle de treball

int conversio;
uint16 x = 0x0000; // Crides a enviar paquet PWM
uint16 y = 0x0000; // Paquets PWM que no es poden enviar
uint16 w = 0x0000; // Paquets PWM que es processen

//Per el control d'enviament
uint8 control_enviament = 0x00;
uint8 control_recepcio = 0x00;

/*********************************************************************
 * FUNCIONS LOCALS
 */
void XSensors_HandleKeys( byte shift, byte keys );
void XSensors_MessageMSGCB( afIncomingMSGPacket_t *pckt );

void XSensors_SendTemperatura( afIncomingMSGPacket_t *pckt );
void XSensors_SendPressio( afIncomingMSGPacket_t *pckt );
void XSensors_SendCaudal( afIncomingMSGPacket_t *pckt );
void XSensors_SendRevolucions( afIncomingMSGPacket_t *pckt );
void XSensors_SendTot( afIncomingMSGPacket_t *pckt );
void XSensors_SendPWM( uint16 PWM_DC );


void XSensors_RequestTemperatura( void );
void XSensors_RequestPressio( void );
void XSensors_RequestCaudal( void );
```

```
void XSensors_RequestRevolucions( void );
void XSensors_Reenviament(void);

void XSensors_ProcessTemperatura( afIncomingMSGPacket_t *pckt );
void XSensors_ProcessPressio( afIncomingMSGPacket_t *pckt );
void XSensors_ProcessCaudal( afIncomingMSGPacket_t *pckt );
void XSensors_ProcessRevolucions( afIncomingMSGPacket_t *pckt );
void XSensors_ProcessPWM( afIncomingMSGPacket_t *pckt );


void XSensors_Menu( byte menucounter, byte submenucounter, byte menuSW2 );

/*********************************************************************
 * NETWORK LAYER CALLBACKS
 */

/*********************************************************************
 * PUBLIC FUNCTIONS
 */

/*********************************************************************
 * @fn      XSensors_Init
 *
 * @brief   Initialization function for the Generic App Task.
 *          This is called during initialization and should contain
 *          any application specific initialization (ie. hardware
 *          initialization/setup, table initialization, power up
 *          notificaiton ... ).
 *
 * @param   task_id - the ID assigned by OSAL.  This ID should be
 *                    used to send messages and set timers.
 *
 * @return  none
 */
void XSensors_Init( byte task_id )
{
  XSensors_TaskID = task_id;
  XSensors_NwkState = DEV_INIT;
  XSensors_TransID = 0;

  // Device hardware initialization can be added here or in main() (Zmain.c).
  // If the hardware is application specific - add it here.
  // If the hardware is other parts of the device add it in main().

  //Configurem per emetre BroadCast
  XSensors_DstAddr.addrMode = (afAddrMode_t)AddrBroadcast;
  XSensors_DstAddr.endPoint = XSensors_ENDPOINT;
  XSensors_DstAddr.addr.shortAddr = 0xFFFF;

  // Fill out the endpoint description.
```

```
  XSensors_epDesc.endPoint = XSensors_ENDPOINT;
  XSensors_epDesc.task_id = &XSensors_TaskID;
  XSensors_epDesc.simpleDesc
          = (SimpleDescriptionFormat_t *)&XSensors_SimpleDesc;
  XSensors_epDesc.latencyReq = noLatencyReqs;

  // Register the endpoint description with the AF
  afRegister( &XSensors_epDesc );

  // Register for all key events - This app will handle all key events
  RegisterForKeys( XSensors_TaskID );

  // Update the display
#if defined ( LCD_SUPPORTED )
    HalLcdWriteString( "XSensors", HAL_LCD_LINE_1 );
#endif

}

/*********************************************************************
 * @fn      XSensors_ProcessEvent
 *
 * @brief   Generic Application Task event processor.  This function
 *          is called to process all events for the task.  Events
 *          include timers, messages and any other user defined events.
 *
 * @param   task_id  - The OSAL assigned task ID.
 * @param   events - events to process.  This is a bit map and can
 *                   contain more than one event.
 *
 * @return  none
 */
UINT16 XSensors_ProcessEvent( byte task_id, UINT16 events )
{
  afIncomingMSGPacket_t *MSGpkt;
  afDataConfirm_t *afDataConfirm;
  ZDO_NewDstAddr_t *ZDO_NewDstAddr;
  byte dstEP;
  zAddrType_t *dstAddr;

  // Data Confirmation message fields
  byte sentEP;
  ZStatus_t sentStatus;
  byte sentTransID;       // This should match the value sent

  if ( events & SYS_EVENT_MSG )
  {
    MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive( XSensors_TaskID );
    while ( MSGpkt )
    {
```

```
    switch ( MSGpkt->hdr.event )
    {
     case KEY_CHANGE: //Si utilitzem joystick
      XSensors_HandleKeys(  ((keyChange_t  *)MSGpkt)->state,  ((keyChange_t
*)MSGpkt)->keys );
      break;

     case AF_DATA_CONFIRM_CMD:
      // Aquest missatge es rep com a confirmació d'enviament de paquet.
      // L'estatus es de ZStatus_t type [defined in ZComDef.h]
      // Els camps del missatge estan definits en AF.h
      afDataConfirm = (afDataConfirm_t *)MSGpkt;
      sentEP = afDataConfirm->endpoint;
      sentStatus = afDataConfirm->hdr.status;
      sentTransID = afDataConfirm->transID;
      (void)sentEP;
      (void)sentTransID;

      //  Acció a realitzar quan rebem confirmació
      if ( sentStatus != ZSuccess )
      {
       // Les dades no s'han entregat
       XSensors_Reenviament();
       if (control_recepcio = 0x06)
       {
        HalLcdWriteScreen( "ERROR", "NO REBEM RESPOSTA" );
        while(true)
       {
       }//while
       }
      }
      break;

     case AF_INCOMING_MSG_CMD:
      XSensors_MessageMSGCB( MSGpkt );
      break;

     case ZDO_NEW_DSTADDR:
      ZDO_NewDstAddr = (ZDO_NewDstAddr_t *)MSGpkt;

      dstEP = ZDO_NewDstAddr->dstAddrDstEP;
      dstAddr = &ZDO_NewDstAddr->dstAddr;
      XSensors_DstAddr.addrMode = (afAddrMode_t)dstAddr->addrMode;
      XSensors_DstAddr.endPoint = dstEP;

      XSensors_DstAddr.addr.shortAddr = dstAddr->addr.shortAddr;
      break;

     case ZDO_STATE_CHANGE:
      XSensors_NwkState = (devStates_t)(MSGpkt->hdr.status);
```

```
        if ( (XSensors_NwkState == DEV_ZB_COORD)
           || (XSensors_NwkState == DEV_ROUTER)
           || (XSensors_NwkState == DEV_END_DEVICE) )
        {
          // Mostrem per pantalla quan es canvia d'estat
          HalLcdWriteScreen( "ZDO_STATE_CHANGE", "" );
        }
        break;

      default:
        break;
    }

    // Release the memory
    osal_msg_deallocate( (uint8 *)MSGpkt );

    // Next
    MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive( XSensors_TaskID );
  }

  // return unprocessed events
  return (events ^ SYS_EVENT_MSG);
}


// Enviem el duty cycle del PWM períodicament
//  (setup en XSensors_Init()).

  if ( events & XSensors_SEND_PWM_EVT )
{
  // Enviem PWM

    PWM_duty_cycle = 2 * HalAdcRead(HAL_ADC_CHANNEL_5,
HAL_ADC_RESOLUTION_14); // Llegim canal 5 que és on tenim el potenciòmetre
    if (PWM_duty_cycle > 0xFF69 ) { PWM_duty_cycle = 0xFF69;} // Comparem amb
65385 i posem 65434 per no desbordar
    if (PWM_duty_cycle < 0x0096 ) { PWM_duty_cycle = 0x0065;} // Comparem amb
150 i posem 101 per no desbordar
    else {};

    //HalLcdWriteString( "Conversio AD5", HAL_LCD_LINE_1);
    //HalLcdWriteValue(PWM_duty_cycle, 10, HAL_LCD_LINE_2 );

  XSensors_SendPWM( PWM_duty_cycle );

  // Tornem a activar el timer de 0'3 segons

  osal_start_timer( XSensors_SEND_PWM_EVT,
                XSensors_SEND_PWM_TIMEOUT );
```

```
    // return unprocessed events
    return (events ^ XSensors_SEND_PWM_EVT);
  }

  // Discard unknown events
  return 0;
}

/*********************************************************************
 * Event Generation Functions
 */
/*********************************************************************
 * @fn      XSensors_HandleKeys
 *
 * @brief   Handles all key events for this device.
 *
 * @param   shift - true if in shift/alt.
 * @param   keys - bit field for key events. Valid entries:
 *                 HAL_KEY_SW_4
 *                 HAL_KEY_SW_3
 *                 HAL_KEY_SW_2
 *                 HAL_KEY_SW_1
 *
 * @return  none
 */
void XSensors_HandleKeys( byte shift, byte keys )
{
  // Shift is used to make each button/switch dual purpose.
  if ( shift )
  {
    if ( keys & HAL_KEY_SW_1 )
    {
    }
    if ( keys & HAL_KEY_SW_2 )
    {
    }
    if ( keys & HAL_KEY_SW_3 )
    {
    }
    if ( keys & HAL_KEY_SW_4 )
    {
    }
  }
  else
  {
    if ( keys & HAL_KEY_SW_1 ) //JOYSTICK AMUNT
    {
     // XSensors_RequestTemperatura();
```

```
    // PWM

    PWM_duty_cycle = PWM_duty_cycle + 0x0064; // Sumem 100 unitats
    if (PWM_duty_cycle > 0xFF69 ) { PWM_duty_cycle = 0xFF69;} // Comparem amb
65385 i posem 65434 per no desbordar-nos
    else {};
    XSensors_SendPWM(PWM_duty_cycle);

   }

   if ( keys & HAL_KEY_SW_2 ) // JOYSTICK DRETA
   {

XSensors_RequestPressio();

   }

   if ( keys & HAL_KEY_SW_3 ) //JOYSTICK AVALL
   {

    PWM_duty_cycle = PWM_duty_cycle - 0x0064; // Restem 100 unitats
    if (PWM_duty_cycle < 0x0096 ) { PWM_duty_cycle = 0x0065;} // Comparem amb
150 i posem 101
    else {};
    XSensors_SendPWM(PWM_duty_cycle);
   }

   if ( keys & HAL_KEY_SW_4 ) //JOYSTICK ESQUERRA
   {
    osal_start_timer( XSensors_SEND_PWM_EVT,
                XSensors_SEND_PWM_TIMEOUT );
   }
  }
}

/*********************************************************************
 * LOCAL FUNCTIONS
 */

/*********************************************************************
 * @fn      XSensors_MessageMSGCB
 *
 * @brief   És la crida al processador de missatges de dades.  Funció que processa
 *          qualsevol dada d'entrada depenent de l'identificador de clúster
 *
 *
 * @param   none
 *
 * @return  none
 */
```

```
void XSensors_MessageMSGCB( afIncomingMSGPacket_t *pkt )
{
 switch ( pkt->clusterId )
 {
    case XSensors_request_temperatura_CLUSTERID:
    //HalLcdWriteScreen("","");
    HalLcdWriteScreen("Peticio de ","temperatura");
    XSensors_SendTemperatura(pkt);
    break;

    case XSensors_receive_temperatura_CLUSTERID:
    //HalLcdWriteScreen("","");
    HalLcdWriteScreen("Peticio de ","temperatura");
    XSensors_ProcessTemperatura(pkt);
    break;

    case XSensors_request_pressio_CLUSTERID:
    //HalLcdWriteScreen("","");
    HalLcdWriteScreen("Peticio de ","pressio");
    XSensors_SendPressio(pkt);
    break;

    case XSensors_receive_pressio_CLUSTERID:
    //HalLcdWriteScreen("","");
    HalLcdWriteScreen("Peticio de ","pressio");
    XSensors_ProcessPressio(pkt);
    break;

    case XSensors_request_caudal_CLUSTERID:
    //HalLcdWriteScreen("","");
    HalLcdWriteScreen("Peticio de ","caudal");
    XSensors_SendCaudal(pkt);
    break;

    case XSensors_receive_caudal_CLUSTERID:
    //HalLcdWriteScreen("","");
    HalLcdWriteScreen("Peticio de ","caudal");
    XSensors_ProcessCaudal(pkt);
    break;

    case XSensors_request_revolucions_CLUSTERID:
    //HalLcdWriteScreen("","");
    HalLcdWriteScreen("Peticio de ","revolucions");
    XSensors_SendRevolucions(pkt);
    break;

    case XSensors_receive_revolucions_CLUSTERID:
    //HalLcdWriteScreen("","");
    HalLcdWriteScreen("Peticio de ","revolucions");
    XSensors_ProcessRevolucions(pkt);
```

```
      break;

      case XSensors_request_tot_CLUSTERID:
      //HalLcdWriteScreen("","");
      HalLcdWriteScreen("Peticio de ","tot");
      XSensors_SendTot(pkt);
      break;

      case XSensors_send_PWM_CLUSTERID:
      HalLcdWriteScreen("Recepció ","PWM");
      XSensors_ProcessPWM(pkt);
      break;
   }
}


/*********************************************************************
 * @fn      XSensors_RequestTemperatura
 *
 * @brief   Demanem que ens envin valors de temperatura.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_RequestTemperatura( void )
{
  dades_request[0] = 0x00;
for (control_enviament = 0x00; control_enviament < 0x06; control_enviament++)
{
  if ( AF_DataRequest( &XSensors_DstAddr, &XSensors_epDesc,
                       XSensors_request_temperatura_CLUSTERID,
                       1,
                       dades_request,
                       &XSensors_TransID,
                       AF_DISCV_ROUTE,      AF_DEFAULT_RADIUS      )      ==
afStatus_SUCCESS )
  {
    // Successfully requested to be sent.
    control_enviament = 0x06;
    HalLcdWriteScreen( "Enviem peticio", "temperatura" );
  }
  else
  {
    // Error occurred in request to send.
    HalLcdWriteScreen("Error intentant", "enviar");
  }
}//for
}
```

```
/*********************************************************************
 * @fn      XSensors_RequestPressio
 *
 * @brief   Demanem que ens envin valors de pressio.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_RequestPressio( void )
{
  dades_request[0] = 0x00;
for (control_enviament = 0x00; control_enviament < 0x06; control_enviament++)
{
  if ( AF_DataRequest( &XSensors_DstAddr, &XSensors_epDesc,
                       XSensors_request_pressio_CLUSTERID,
                       1,
                       dades_request,
                       &XSensors_TransID,
                       AF_DISCV_ROUTE,      AF_DEFAULT_RADIUS      )      ==
afStatus_SUCCESS )
  {
    // Successfully requested to be sent.
   contro_enviament = 0x06;
    HalLcdWriteScreen( "Enviem peticio", "pressio" );
  }
  else
  {
    // Error occurred in request to send.
    HalLcdWriteScreen("Error intentant", "enviar");
  }
}//for
}

/*********************************************************************
 * @fn      XSensors_RequestCaudal
 *
 * @brief   Demanem que ens envin valors de caudal.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_RequestCaudal( void )
{

  dades_request[0] = 0x00;
for (control_enviament = 0x00; control_enviament < 0x06; control_enviament++)
{
  if ( AF_DataRequest( &XSensors_DstAddr, &XSensors_epDesc,
```

```
                XSensors_request_caudal_CLUSTERID,
                1,
                dades_request,
                &XSensors_TransID,
                AF_DISCV_ROUTE,         AF_DEFAULT_RADIUS         )         ==
afStatus_SUCCESS )
  {
   // Successfully requested to be sent.
   control_enviament = 0x06;
   HalLcdWriteScreen( "Enviem peticio", "caudal" );
  }
  else
  {
   // Error occurred in request to send.
   HalLcdWriteScreen("Error intentant", "enviar");
  }
}//for
}

/*********************************************************************
 * @fn      XSensors_RequestRevolucions
 *
 * @brief   Demanem que ens envin valors de revolucions.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_RequestRevolucions( void )
{

  dades_request[0] = 0x00;
for (control_enviament = 0x00; control_enviament < 0x06; control_enviament++)
{
  if ( AF_DataRequest( &XSensors_DstAddr, &XSensors_epDesc,
                XSensors_request_revolucions_CLUSTERID,
                1,
                dades_request,
                &XSensors_TransID,
                AF_DISCV_ROUTE,         AF_DEFAULT_RADIUS         )         ==
afStatus_SUCCESS )
  {
   // Successfully requested to be sent.
   control_enviament = 0x06;
   HalLcdWriteScreen( "Enviem peticio", "revolucions" );
  }
  else
  {
   // Error occurred in request to send.
   HalLcdWriteScreen("Error intentant", "enviar");
```

```
  }
}//for
}
/*********************************************************************
 * @fn      XSensors_SendTemperatura
 *
 * @brief   Enviem valors de temperatura.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_SendTemperatura( afIncomingMSGPacket_t *pkt )
{

  conversio                =                HalAdcRead(HAL_ADC_CHANNEL_0,
HAL_ADC_RESOLUTION_14);
  HalLcdWriteString( "Conversio AD0", HAL_LCD_LINE_1);
  HalLcdWriteValue(conversio, 10, HAL_LCD_LINE_2 );
  dades_send[0] = 0x00;
  dades_send[1] = LO_UINT16(conversio);
  dades_send[2] = HI_UINT16(conversio);
for (control_enviament = 0x00; control_enviament < 0x06; control_enviament++)
{
  if ( AF_DataRequest( &XSensors_DstAddr, &XSensors_epDesc,
                       XSensors_receive_temperatura_CLUSTERID,
                       3,
                       dades_send,
                       &XSensors_TransID,
                       AF_DISCV_ROUTE,      AF_DEFAULT_RADIUS      )      ==
afStatus_SUCCESS )
  {
    // Successfully requested to be sent.
   control_enviament = 0x06;
  }
  else
  {
    // Error occurred in request to send.
    HalLcdWriteScreen("Error intentant", "enviar");
  }
}//for
}

/*********************************************************************
 * @fn      XSensors_SendPressio
 *
 * @brief   Enviem valors de pressio.
 *
 * @param   none
 *
```

```
 * @return  none
 */
void XSensors_SendPressio( afIncomingMSGPacket_t *pkt )
{
 conversio                =                HalAdcRead(HAL_ADC_CHANNEL_1,
HAL_ADC_RESOLUTION_14);
 HalLcdWriteString( "Conversio AD1", HAL_LCD_LINE_1);
 HalLcdWriteValue(conversio, 10, HAL_LCD_LINE_2 );
 dades_send[0] = 0x00;
 dades_send[1] = LO_UINT16(conversio);
 dades_send[2] = HI_UINT16(conversio);
for (control_enviament = 0x00; control_enviament < 0x06; control_enviament++)
{
 if ( AF_DataRequest( &XSensors_DstAddr, &XSensors_epDesc,
                XSensors_receive_pressio_CLUSTERID,
                3,
                dades_send,
                &XSensors_TransID,
                AF_DISCV_ROUTE,        AF_DEFAULT_RADIUS        )        ==
afStatus_SUCCESS )
 {
  // Successfully requested to be sent.
       control_enviament = 0x06;
}
 else
 {
  // Error occurred in request to send.
  HalLcdWriteScreen("Error intentant", "enviar");
 }
} //for
}
/*********************************************************************
 * @fn      XSensors_SendCaudal
 *
 * @brief   Enviem valors de caudal.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_SendCaudal( afIncomingMSGPacket_t *pkt )
{

 conversio                =                HalAdcRead(HAL_ADC_CHANNEL_2,
HAL_ADC_RESOLUTION_14);
 HalLcdWriteString( "Conversio AD2", HAL_LCD_LINE_1);
 HalLcdWriteValue(conversio, 10, HAL_LCD_LINE_2 );
 dades_send[0] = 0x00;
 dades_send[1] = LO_UINT16(conversio);
 dades_send[2] = HI_UINT16(conversio);
```

```
for (control_enviament = 0x00; control_enviament < 0x06; control_enviament++)
{
 if ( AF_DataRequest( &XSensors_DstAddr, &XSensors_epDesc,
                      XSensors_receive_caudal_CLUSTERID,
                      3,
                      dades_send,
                      &XSensors_TransID,
                      AF_DISCV_ROUTE,          AF_DEFAULT_RADIUS          )          ==
afStatus_SUCCESS )
  {
   // Successfully requested to be sent.
   control_enviament = 0x06;
  }
  else
  {
   // Error occurred in request to send.
   HalLcdWriteScreen("Error intentant", "enviar");
  }
}//for
}

/*********************************************************************
 * @fn      XSensors_Reenviament
 *
 * @brief   Reenviem si no rebem ack.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_Reenviament( void )
{


if (control_recepcio < 0x06)
{
  if ( AF_DataRequest( &XSensors_DstAddr, &XSensors_epDesc,
                      XSensors_CLUSTERID,
                      3,
                      dades_send,
                      &XSensors_TransID,
                      AF_DISCV_ROUTE,          AF_DEFAULT_RADIUS          )          ==
afStatus_SUCCESS )
  {
   // Successfully requested to be sent.
   control_enviament = 0x06;
  }
  else
  {
   // Error occurred in request to send.
```

```
   HalLcdWriteScreen("Error intentant", "enviar");
  }
}//if
control_recepcio++;
}

/*********************************************************************
 * @fn      XSensors_SendRevolucions
 *
 * @brief   Enviem valors de revolucions.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_SendRevolucions( afIncomingMSGPacket_t *pkt )
{

  conversio                 =                 HalAdcRead(HAL_ADC_CHANNEL_3,
HAL_ADC_RESOLUTION_14);
  HalLcdWriteString( "Conversio AD3", HAL_LCD_LINE_1);
  HalLcdWriteValue(conversio, 10, HAL_LCD_LINE_2 );
  dades_send[0] = 0x00;
  dades_send[1] = LO_UINT16(conversio);
  dades_send[2] = HI_UINT16(conversio);
for (control_enviament = 0x00; control_enviament < 0x06; control_enviament++)
{
  if ( AF_DataRequest( &XSensors_DstAddr, &XSensors_epDesc,
                       XSensors_receive_revolucions_CLUSTERID,
                       3,
                       dades_send,
                       &XSensors_TransID,
                       AF_DISCV_ROUTE,      AF_DEFAULT_RADIUS      )      ==
afStatus_SUCCESS )
  {
    // Successfully requested to be sent.
    control_enviament = 0x06;
  }
  else
  {
    // Error occurred in request to send.
    HalLcdWriteScreen("Error intentant", "enviar");
  }
}//for
}
/*********************************************************************
 * @fn      XSensors_SendTot
 *
 * @brief   Enviem valors de tots els sensors.
 *
```

```
 * @param   none
 *
 * @return  none
 */
void XSensors_SendTot( afIncomingMSGPacket_t *pkt )
{

  XSensors_SendTemperatura(pkt);
  XSensors_SendPressio(pkt);
  XSensors_SendCaudal(pkt);
  XSensors_SendRevolucions(pkt);
  HalLcdWriteScreen( "Enviem tot", "");
}

/*********************************************************************
 * @fn      XSensors_ProcessTemperatura
 *
 * @brief   Processem valors de temperatura.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_ProcessTemperatura( afIncomingMSGPacket_t *pkt)
{
  header = pkt->cmd.Data[0];
  dades_process = BUILD_UINT16(pkt->cmd.Data[1], pkt->cmd.Data[2] );
  HalLcdWriteString( "Rebut temp:", HAL_LCD_LINE_1);
  HalLcdWriteValue(dades_process, 10, HAL_LCD_LINE_2 );
}

/*********************************************************************
 * @fn      XSensors_ProcessPressio
 *
 * @brief   Processem valors de pressio.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_ProcessPressio( afIncomingMSGPacket_t *pkt )
{
  header = pkt->cmd.Data[0];
  dades_process = BUILD_UINT16(pkt->cmd.Data[1], pkt->cmd.Data[2] );
  HalLcdWriteString( "Rebut pressio:", HAL_LCD_LINE_1);
  HalLcdWriteValue(dades_process, 10, HAL_LCD_LINE_2 );
}

/*********************************************************************
 * @fn      XSensors_ProcessCaudal
```

```
 *
 * @brief   Processem valors de caudal.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_ProcessCaudal( afIncomingMSGPacket_t *pkt )
{
  header = pkt->cmd.Data[0];
  dades_process = BUILD_UINT16(pkt->cmd.Data[1], pkt->cmd.Data[2] );
  HalLcdWriteString( "Rebut caudal:", HAL_LCD_LINE_1);
  HalLcdWriteValue(dades_process, 10, HAL_LCD_LINE_2 );
}

/*********************************************************************
 * @fn      XSensors_ProcessRevolucions
 *
 * @brief   Processem valors de revolucions.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_ProcessRevolucions( afIncomingMSGPacket_t *pkt )
{
  header = pkt->cmd.Data[0];
  dades_process = BUILD_UINT16(pkt->cmd.Data[1], pkt->cmd.Data[2] );
  HalLcdWriteString( "Rebut rev:", HAL_LCD_LINE_1);
  HalLcdWriteValue(dades_process, 10, HAL_LCD_LINE_2 );
}



/*********************************************************************
 * @fn      XSensors_Menu
 *
 * @brief   Gestionem els missatges del menu.
 *
 * @param   none
 *
 * @return  none
 */

void XSensors_Menu( byte menucounter, byte submenucounter, byte menuSW2 )
{
  //HalLcdWriteValue(menuSW2, 10, HAL_LCD_LINE_1 );
  //HalLcdWriteValue(menucounter, 10, HAL_LCD_LINE_2 );
 switch ( menuSW2 )
  {
```

```
case 0x00:

 if(menucounter == 0x00) {HalLcdWriteScreen( "Peticio", "Temperatura?");}
 if(menucounter == 0x01) {HalLcdWriteScreen( "Peticio", "Pressio?");}
 if(menucounter == 0x02) {HalLcdWriteScreen( "Peticio", "Caudal?");}
 if(menucounter == 0x03) {HalLcdWriteScreen( "Peticio", "Revolucions?");}
 if(menucounter == 0x04) {HalLcdWriteScreen( "Peticio", "Tot?");}
 else {;}

break;

case 0x01:

 if(menucounter == 0x00)
 {
  HalLcdWriteScreen( "Peticio", "Temperatura");
  XSensors_RequestTemperatura();
 }
 if(menucounter == 0x01)
 {
  HalLcdWriteScreen( "Peticio", "Pressio");
  XSensors_RequestPressio();
 }
 if(menucounter == 0x02)
 {
  HalLcdWriteScreen( "Peticio", "Caudal");
  XSensors_RequestCaudal();
 }
 if(menucounter == 0x03)
 {
  HalLcdWriteScreen( "Peticio", "Revolucions");
  XSensors_RequestRevolucions();
 }
 if(menucounter == 0x04)
 {
  //XSensors_RequestTot();
  if(submenucounter == 0x00) {HalLcdWriteScreen( "Peticio", "Tot");}
  if(submenucounter == 0x01)
  {
   HalLcdWriteScreen( "Show", "Temperatura");
   XSensors_RequestTemperatura();
  }
  if(submenucounter == 0x02)
  {
   HalLcdWriteScreen( "Show", "Pressio");
   XSensors_RequestPressio();
  }
  if(submenucounter == 0x03)
```

```
        {
         HalLcdWriteScreen( "Show", "Caudal");
         XSensors_RequestCaudal();
        }
        if(submenucounter == 0x04)
        {
         HalLcdWriteScreen( "Show", "Revolucions");
         XSensors_RequestRevolucions();
        }
        else {;}
       }
       else {;}

     break;

     case 0x02:
       menu_counter = 0x00;
       submenu_counter = 0x00;
       menu_SW2 = 0x00;
       HalLcdWriteScreen( "Peticio", "Temperatura?");
     break;

  } // switch ( menuSW2 )


}

/*********************************************************************
 * @fn      XSensors_SendPWM
 *
 * @brief   Enviem Duty Cycle.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_SendPWM( uint16 PWM_DC )
{

  /* Comptem tots els intents d'enviar paquets fins a 500 per poder fer % de paquets OK
i no
  if ( x < 500 )
  {
  x++;
  HalLcdWriteValue(x, 10, HAL_LCD_LINE_1 );
  */

//  HalLcdWriteString( "Duty Cycle", HAL_LCD_LINE_1);
//  HalLcdWriteValue(PWM_DC, 10, HAL_LCD_LINE_2 );
  dades_send[0] = 0x00;
```

```
  dades_send[1] = LO_UINT16(PWM_DC);
  dades_send[2] = HI_UINT16(PWM_DC);

for (control_enviament = 0x00; control_enviament < 0x06; control_enviament++)
{
  if ( AF_DataRequest( &XSensors_DstAddr, &XSensors_epDesc,
                       XSensors_send_PWM_CLUSTERID,
                       3,
                       dades_send,
                       &XSensors_TransID,
                       AF_DISCV_ROUTE,          AF_DEFAULT_RADIUS          )          ==
afStatus_SUCCESS )
  {
    // Successfully requested to be sent.
  control_enviemant = 0x006;
}
  else
  {
    // Error occurred in request to send.
    //HalLcdWriteScreen("Error intentant", "enviar");

    /* Comptem paquets que no es poden enviar
    y++;
    HalLcdWriteValue(y, 10, HAL_LCD_LINE_2 );
    */
  }

  /* El que utilitzem per contar fins a 500 per fer % paquets enviats i no
  } //if ( x < 500 )
  else {;}
  */
} //for
}

/*********************************************************************
 * @fn      XSensors_ProcessPWM
 *
 * @brief   Processem valors de duty cycle.
 *
 * @param   none
 *
 * @return  none
 */
void XSensors_ProcessPWM( afIncomingMSGPacket_t *pkt )
{
  header = pkt->cmd.Data[0];
  dades_process = BUILD_UINT16(pkt->cmd.Data[1], pkt->cmd.Data[2] );
 // HalLcdWriteString( "Duty Cycle:", HAL_LCD_LINE_1);
  //HalLcdWriteValue(dades_process, 10, HAL_LCD_LINE_2 );
```

```
   /* Comptem paquets que processem
   w++;
   HalLcdWriteValue(w, 10, HAL_LCD_LINE_2 );
   */

    // P W M
    P1SEL = 0X03; // P1_0 P1_1 P1_2 en funció de perifèic
    PERCFG = 0X40; // Timer 1 I/O location alternative 2 (port 1)
    P2SEL = 0X08; // Prioritat sempre de timer 1
        T1CTL = 0X0E; // Preescala a 128 (per tant 250KHz) i modulo mode
    T1CCTL1 = 0x24; // Timer 1 canal 1 en output compare mode 100 i no
interrupcions
    T1CCTL2 = 0x24; // Timer 1 canal 2 en output compare mode 100 i no
interrupcions
    T1CC0H = 0xFF; // Fixem període
    T1CC0L = 0xFF; // Fixem període

    T1CC1H = HI_UINT16(dades_process); // Duty cycle
    T1CC1L = LO_UINT16(dades_process); // Duty cycle


}

/*****************************************************************
*****************************************************************/
```

Apèndix 9. Test de velocitat de transferència

S'ha realitzat una prova de test per tal de valorar la qualitat de la comunicació. Aquesta prova consisteix en comparar la distància entre dispositius i l'èxit en l'enviament/recepció de missatges.

9.1 Introducció

Hem agafat l'aplicació que hem desenvolupat en l'apèndix 8 "Premisses, programació i algorismes comentats" en el mode de treball d'enviament de cicle de treball com a resultat d'un esdeveniment provocat per un timer, i hem variat el període de l'esdeveniment i la distància entre dispositius.

9.2 Resultats

La prova consisteix en enviar 100 missatges de cicle de treball (és a dir, lectura del port ADC, enviament de missatge, recepció de missatge, processar missatge i modificar el cicle de treball del PWM) i fer un enregistrament dels paquets que no s'han pogut enviar i els paquets que hem rebut. Les distàncies seleccionades han estat de 0, 5, 10, 15, 20, 25, 30, 40 i 45 metres. La distància de 45 metres s'ha fet sense visibilitat entre dispositius, és a dir, un dispositiu dins de les instal·lacions i l'altre a 45 metres però a fora de la nau. Els períodes d'enviament de missatge (0'1, 0'2, 0'3 i 0'4 segons)s'han elegit durant l'experiment, hem començat amb el període de 0'1 segons i observant els resultats hem triat el següent període.

Els resultats es mostren en les següents taules:

Enviem missatge cada 0'1 segons

Distància entre dispositius (metres)	0	5	10	15	20	25	30	40	45 sense visibilitat
Missatges que no es poden enviar	64	64	64	64	64	64	64	64	64
Missatges rebuts	36	36	36	36	36	36	36	36	36
Missatges no rebuts	0	0	0	0	0	0	0	0	0
Percentatge (missatges enviats/missatges rebuts	100%	100%	100%	100%	100%	100%	100%	100%	100%

Enviem missatge cada 0'2 segons

Distància entre dispositius (metres)	0	5	10	15	20	25	30	40	45 sense visibilitat
Missatges que no es poden enviar	37	37	37	37	37	37	37	37	37
Missatges rebuts	62	62	62	62	62	62	62	62	62
Missatges no rebuts	1	1	1	1	1	1	1	1	1
Percentatge (missatges enviats/missatges rebuts	98'4%	98'4%	98'4%	98'4%	98'4%	98'4%	98'4%	98'4%	98'4%

Enviem missatge cada 0'3 segons

Distància entre dispositius (metres)	0	5	10	15	20	25	30	40	45 sense visibilitat
Missatges que no es poden enviar	10	10	10	10	10	10	10	10	10
Missatges rebuts	87	87	87	87	87	86	86	86	86
Missatges no rebuts	3	3	3	3	3	4	4	4	4
Percentatge (missatges enviats/missatges rebuts	96'7%	96'7%	96'7%	96'7%	96'7%	95'5%	95'5%	95'5%	95'5%

Enviem missatge cada 0’4 segons

Distància entre dispositius (metres)	0	5	10	15	20	25	30	40	45 sense visibilitat
Missatges que no es poden enviar	0	0	0	0	0	0	0	0	0
Missatges rebuts	100	100	100	100	100	100	100	100	100
Missatges no rebuts	0	0	0	0	0	0	0	0	0
Percentatge (missatges enviats/missatges rebuts	100%	100%	100%	100%	100%	100%	100%	100%	100%

Aquest gràfic mostra els resultats obtinguts entre els 0 i 20 metres:

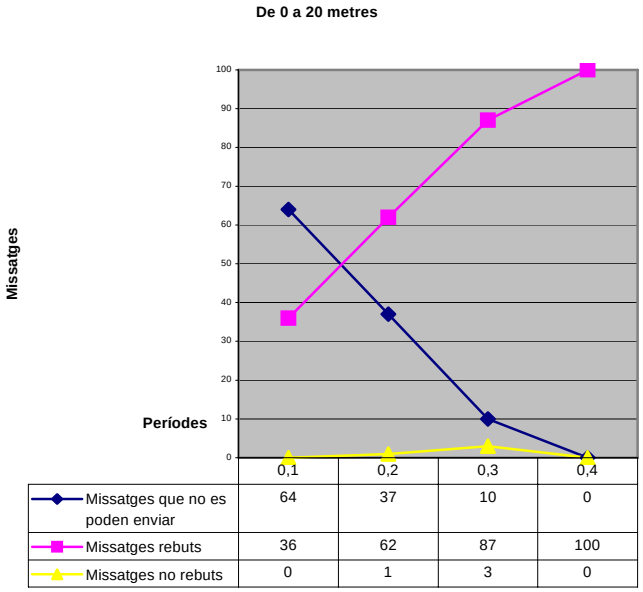
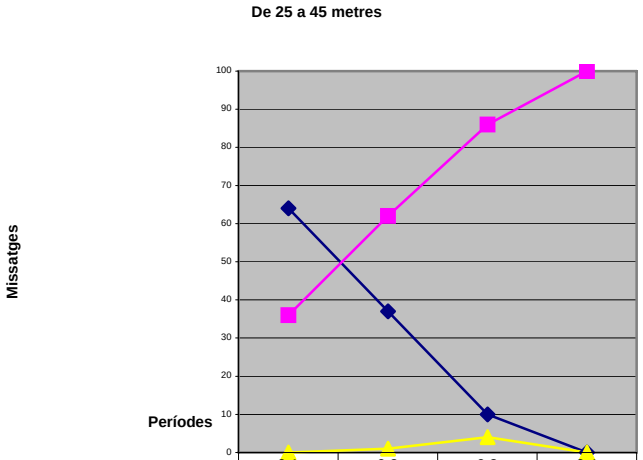


Figura 9.2.1 Transmissions de 0 a 20 metres

Aquest gràfic mostra els resultats obtinguts entre els 25 i 45 metres:



	0,1	0,2	0,3	0,4
Missatges que no es poden enviar	64	37	10	0
Missatges rebuts	36	62	86	100
Missatges no rebuts	0	1	4	0

Figura 9.2.2 Transmissions de 25 a 45 metres 122

9.3 Conclusions

Com s'observa, la taxa de transferència queda limitada per la capacitat que té el CC2430 en processar els missatges que s'han d'enviar, no pas per la capacitat que té per rebre missatges ni per la distància.

Així ens trobem que amb un període de 0'1 segons el 64% dels missatges no es poden enviar, en canvi es reben el 100 % dels missatges que si s'aconsegueixen enviar. A mesura que augmentem el període es compensa la quantitat de missatges que no es poden enviar, amb els que si que s'envien, els que es reben i els que es perden.

En referència a la distància de separació entre dispositius, cal dir que fins als 40 metres de separació no hi ha hagut cap variació en les taxes de missatges, en excepció del període de 0'3 segons (que a partir dels 25 metres es perd un paquet més que en distàncies inferiors). El que si que hem detectat és que al treballar amb 45 metres de distància i sense visibilitat (ja que un dels dispositius es trobava situat fora de les instal·lacions) en algun moment costava que els dispositius es detectessin, però un cop detectats es mantenen les taxes de comunicació.

Veient els resultats, s'ha d'elegir un període de transferència que permeti un bon resultat en quant a la relació de missatges que s'intenten enviar, els que realment s'envien, els que es reben i que, encara que no hagi de ser en temps real, tampoc s'hagi d'agafar un període excessivament llarg. Així que veient el que hem obtingut en l'experiència realitzada optem per elegir com a període d'enviament de missatges de correcció del cicle de treball del PWM 0'3 segons.

Apèndix 10. Literatura

En aquest apartat farem un llistat de referències que aporten informació addicional al tema que estem estudiant:

- Més informació sobre ZigBee al següent enllaç:
  http://en.wikipedia.org/wiki/ZigBee#_note-chipcost
- Comparativa ZigBee i Bluetooth
  http://spanish.bluetooth.com/Bluetooth/Learn/Technology/Compare/
- Més informació sobre el protocol CSMA
  http://en.wikipedia.org/wiki/Carrier_Sense_Multiple_Access
- Més informació sobre la tècnica de modulació DSSS:
  http://en.wikipedia.org/wiki/Direct-sequence_spread_spectrum
- Més informació sobre les modulacions BSPK i QPSK en els següents enllaços:
  http://en.wikipedia.org/wiki/BPSK
  http://en.wikipedia.org/wiki/Phase-shift_keying
  http://es.wikipedia.org/wiki/Modulaci%B3n_por_desplazamiento_de_fase

Figures

Llistat de les figures dels apèndixs:

Memòria elaborada per Xavier López Puig,
que la signa per donar-ne fe.

Aquest memòria de projecte final de carrera és un estudi i implementació bàsica de l'estat actual en quant a les tecnologies/protocols sense fils existents, en particular centrat en ZigBee sobre la plataforma CC2430 de Texas Instruments, per a una aplicació industrial amb una taxa de transferència de dades baixa, que no presenta un alt grau de complexitat, però que requereix gran versatilitat i fiabilitat.

Esta memoria de proyecto final de carrera es un estudio e implementación básica del estado actual de las tecnologías/protocolos inalámbricos existentes, en particular centrado en ZigBee sobre la plataforma CC2430 de Texas Instruments, para una aplicación industrial con una tasa de transferencia de datos baja, que no presenta un grado alto de complejidad, pero que requiere gran versatilidad i fiabilidad.

This degree thesis' memory is a research and a basic implementation of the up-to-date wireless technologies/protocols, particularly focused on the Texas Instruments solution CC2430, for a low data rate industrial application, which is not very complex, but demands a high versatility and reliability level.