Implementació protocol TSCH al hardware OpenMote

Albert Creixell Antolín
Màster Universitari en Enginyeria de Telecomunicacions

Pere Tuset Peiró

09/01/2015

FITXA DEL TREBALL FINAL

Títol del treball:	*Implementació protocol TSCH al hardware OpenMote*
Nom de l'autor:	*Albert Creixell Antolín*
Nom del consultor:	*Pere Tuset Peiró*
Data de lliurament (mm/aaaa):	*01/2015*
Àrea del Treball Final:	*TFM-Telemàtica*
Titulació:	*Màster Universitari en Enginyeria de Telecomunicacions*

Resum del Treball (màxim 250 paraules):

El projecte ha desenvolupat el codi C per implementar comunicacions TSCH tal com especifica la norma IEEE802.15.4e, utilitzant el hardware OpenMote per executar aquest codi. Les comunicacions TSCH permeten comunicacions deterministes via radio a 2,4GHz, aportant estalvi energètic i insensibilitat a interferències.

El TSCH es basa en la utilització de dos tècniques, una de divisió per temps en ranures i una altra de salt de freqüència.

La primera utilitza una sincronització de temps entre totes les remotes que formen la xarxa per activar la radio només en els instants d'escriptura o lectura, de forma que la radio està desactivada la majoria del temps estalviant energia i a la vegada existint un ordre de comunicacions que assegura comunicacions deterministes.

La segona tècnica es basa en utilitzar els 16 canals disponibles segons un algoritme que determina en cada ranura quin canal utilitzar, de forma que es minimitza l'efecte d'alguna interferència en un canal concret.

Utilitzant aquest codi, el projecte ha desenvolupat una aplicació exemple en la que s'utilitzen dues OpenMote, que comuniquen les dades de temperatura, humitat i llum d'un a l'altre via TSCH, i el receptor envia les dades rebudes per RS232 a una aplicació de pc.

Abstract (in English, 250 words or less):

The project has developed the C code to implement communications TSCH as specified in the standard IEEE802.15.4e using OpenMote hardware to run this code. TSCH allow deterministic communications with 2,4 GHz radio, providing energy saving method and insensitive to interference.

The TSCH is based on the use of two techniques, a division of time in slots and frequency hopping.

The first uses a time synchronization among all the remotes to turn up the radio network only in the moments to read or write, so that the radio is off saving energy and the order of communication that ensures deterministic functionality.

The second technique is to use the 16 available channels according to an algorithm that determines which channel to use in each slot, thus minimizing the effect of interference on a particular channel.

Using this code, the project has developed an application that uses the sample of two OpenMote, communicate the data of temperature, humidity and light each other via TSCH, and the receiver sends the received data to an pc application for RS232.

Paraules clau (entre 4 i 8):

IEEE 802.15.4e; TSCH; OpenMote; Timeslotted Channel Hopping; Salt de freqüència a temps ranurat

Índex

1. Objectiu, gestió del projecte i resultat esperat

1.1 Context i justificació del Treball

El creixement d'Internet a les Coses ha generat la necessitat d'unes electròniques de molt baix consum amb gran flexibilitat en les comunicacions, i unit a aquestes, un software i protocols de comunicacions que cobreixin aquestes necessitats descrites.

El present projecte s'ha desenvolupat en aquest context i parteix d'un hardware especialment adaptat per Internet a les Coses, dotat d'un microcontrolador de baix consum, programable en C i amb una radio de 2,4GHz.

El projecte volia desenvolupar un protocol que proporcioni comunicacions radio fiables i de baix consum, objectius alineats amb els establerts pel hardware i que pugui ser utilitzat per gran quantitat d'aplicacions d'Internet a les Coses.

1.2 Objectius del Treball

Els objectius del projecte és llisten a continuació:

- Entendre els procediments establerts en la norma 802.15.4e per les comunicacions TSCH.
- Programar el codi en C que desenvolupi les comunicacions TSCH
- Programar un programa exemple que utilitzi i demostri el funcionament del codi realitzat.
- Compilar i provar el codi desenvolupat en el hardware OpenMote

1.3 Enfocament i mètode seguit

El projecte fonamentalment s'ha tractat com un projecte de TIC, en el que calia definir clarament l'objectiu desitjat i desenvolupar aquest objectiu determinant un abast clar. Un cop definit l'abast, es van marcar unes fites amb una planificació detallada de tasques i fites a aconseguir, amb dates màximes clares. Es va considerar que per aquesta fase de inicialització i planificació, la millor estratègia a seguir era redactar l'acta de inicialització de projecte segons els procediments establerts al PMBOK, i posteriorment redactar un document que defineixi l'abast del projecte amb claredat, ja que són els procediments més acceptats per executar projectes TIC. L'acta d'inicialització s'adjunta en els annexos i la definició en el document preparat per la PAC1, també en els annexos.

Per desenvolupar el projecte, l'estratègia a seguir va ser la marcada a la planificació, primer calia preparar l'entorn de programació, amb els

editors i compiladors necessaris. Posteriorment, desenvolupar programes de proves per conèixer el hardware. Calia realitzar programes de proves de control dels leds, enviar i rebre missatges via radio, proves amb el rtc i enviar i rebre missatges via RS232. Gràcies a aquests programes es va adquirir coneixement del comportament del hardware que serviria per definir el programa i es van eliminar possibles riscos del projecte al plantejar el codi de forma errònea i topar un cop fet en que el hardware no podia executar alguna tasca necessària.

El codi nou es va desenvolupar partint de cero, i aprofitant únicament les llibreries que aporta el firmware de l'OpenMote. Abans de començar a programar, es van redactar els documents d'especificació de com comunicaran i es sincronitzaran les remotes. Aquests documents es van enviar al tutor per tal de tenir l'aprovació i correcció. Un cop es tenia definida la funcionalitat es va a començar el procés de programació del codi. Gràcies a seguir aquesta estratègia de treball es va aconseguir definir molt be l'operativa desitjada i minimitzar les correccions del codi.

Durant tot el procés de programació, el codi es va anar provant, i la metodologia de seguiment i control es va basar en la verificació del correcte funcionament de cada un dels estats i de les seves transicions. Posteriorment, com a prova final, s'havia de programar una aplicació real que utilitzes el codi per enviar trames TSCH, i així comprovar el funcionament general i el control de retràs de temps.

Gràcies a aquestes estratègies de definició, planificació i proves detallades el software s'ha programat en el temps establert i de forma operativa i funcional.

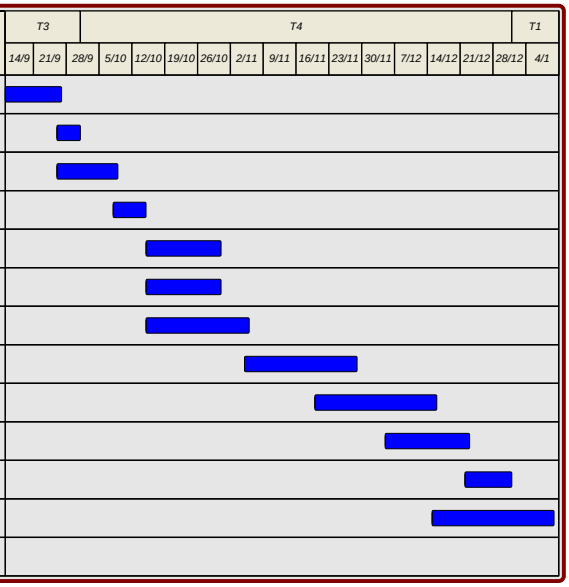
1.4 Planificació del Treball

Per executar el projecte, s'ha dissenyat un Diagrama de Gantt amb les següents fites:

Fita	Data realització
Tancament de la definició del projecte	27/09/2014
Recopilació d'informació de la norma IEEE 802.15.4e	30/09/2014
Definició de l'abast del projecte i planificació. **AQUESTA FITA ÉS LA PAC 1**	08/10/2014
Recollida de material hardware i màquines virtuals	14/10/2014
Redacció document de la màquina d'estats. **PART DE LA DOCUMENTACIÓ DE LA PAC 2**	30/10/2014
Redacció del document de sincronització. **PART DE LA DOCUMENTACIÓ DE LA PAC 2**	05/11/2014
Programació sistema simple per enviar trames nomes amb llibreria radio. **PART DE LA DOCUMENTACIÓ DE LA PAC 2**	30/11/2014

Programació llibreries protocol TSCH i sincronització de l'ASN. **PART DE LA DOCUMENTACIÓ DE LA PAC 3**	15/12/2014
Proves de comunicacions i realització de codi exemple. **PART DE LA DOCUMENTACIÓ DE LA PAC 3**	22/12/2014
Redacció de memòria tècnica, diapositives i documentació. **FITA DE L'ENTREGA DE LA MEMÒRIA**	03/01/2015
Presentació del projecte davant del tribunal	10/01/2015

Id.	*Nombre de tarea*	*Comienzo*	*Fin*	*Duración*
1	Decisió del projecte a executar	15/09/2014	26/09/2014	10d
2	Recopilació de dades norma IEEE802.15.4e	26/09/2014	30/09/2014	3d
3	Redacció acta inicialització, i la PAC 1 (amb l'abast i planificació del projecte)	26/09/2014	08/10/2014	9d
4	Preparació entorn programació amb màquina virtual	08/10/2014	14/10/2014	5d
5	Realització prgrames proves radio, RTC, leds i port sèrie	15/10/2014	30/10/2014	12d
6	Preparació document procés comunicació	15/10/2014	30/10/2014	12d
7	Preparació document procés sincronització	15/10/2014	05/11/2014	16d
8	Desenvolupament codi TSCH	05/11/2014	28/11/2014	18d
9	Desenvolupament codi sincronització	20/11/2014	15/12/2014	18d
10	Programació aplicació exemple	05/12/2014	22/12/2014	12d
11	Proves del sistema finals	22/12/2014	31/12/2014	8d
12	Redacció memòria final	15/12/2014	09/01/2015	20d
13	Presentació projecte	19/01/2015	19/01/2015	1d



El projecte ha tingut una sèrie d'entregables que es detallen a continuació:

- Document de la màquina d'estats per gestionar l'enviament i recepció de les dades del TSCH, segons IEEE 802.15.4e. Aquest document és part de l'apartat 3.3
- Document explicatiu del procediment de sincronització entre remotes. Aquest document és part de l'apartat 3.3
- Llibreria TSCH escrita en C i provada en la plataforma OpenMote
- Document de les proves realitzades i dels seus resultats. Aquest document és part de l'apartat 3.4
- Programa exemple que utilitzi la llibreria desenvolupada, que sincronitzi un coordinador i un esclau, i comuniqui dades utilitzant el protocol TSCH
- Memòria final, incloent explicació del protocol TSCH, explicació de l'organització lògica de la llibreria, codis exemple i el codi font de la llibreria.
- Presentació final del projecte per fer la defensa. Dispositives en PowerPoint.

1.5 Breu sumari de productes obtinguts

El projecte ha obtingut un codi en C++, que es pot carregar en la plataforma OpenMote que implementa comunicacions TSCH.

El projecte també proporciona una aplicació C++ que envia dades de temperatura, humitat i llum a una altra OpenMote, i el receptor envia les dades a un pc mitjançant el port sèrie.

I finalment, un programa en .Net Framework, que rep les dades pel port sèrie, les mostra en pantalla i fa unes gràfiques en temps real.

1.6 Breu descripció dels altres capítols de la memòria

La present memòria consta d'un capítol 2 on es fa menció al concepte Internet a les Coses, s'explica el hardware utilitzat, els processos de sincronització, lectura i escriptura TSCH i l'aplicació exemple desenvolupada.

Posteriorment, s'expliquen unes conclusions i en els annexes s'inclouen l'acta d'inicialització feta, el document de la PAC 1, el codi TSCH i el codi de l'aplicació .Net Framework.

2. Estat de la tecnologia

2.1. Introducció a Internet a les Coses

a. Introducció

Internet a les coses, és un concepte que es refereix a la interconnexió d'objectes a Internet, per tal d'obtenir i rebre informació. El concepte va ser proposat per Kevin Ashton en el Auto-ID Center del MIT al 1999, on es fan estudis sobre la identificació per radiofreqüència en xarxa i tecnologies de sensors.

En primera instancia es pot pensar que pocs elements poden ser susceptibles de connectar-se, com ara sensor o màquines, però existeixen gran quantitat d'elements quotidiants o no, que poden oferir avantatges pel fet d'estar connectats. Per exemple, es facilitaria enormement tota la gestió d'stocks, de garanties, de caducitats, de propietats, entre altres millores.

Donat que un gran nombre de productes poden ser susceptibles de ser connectats, és important disposar d'electròniques de molt baix consum, amb comunicacions sense fils i de molt baix cost, per tal que aportin les avantatges dites a la vegada que no incrementi sensiblement el cost del producte.

Unit a aquest hardware, s'han de desenvolupar protocols de comunicacions i software molt pensats per aquestes aplicacions, en les que les quantitats d'informació a moure són molt petites i l'autonomia de la solució molt important, generant un nou tipus de sistema amb uns requeriments molt diferents als entorns pc o mòvil convencionals.

b. Utilitats i aplicacions

Les utilitats de connectar una gran quantitat d'aparells i sensors en una gran xarxa són quasi infinites. El fet de disposar d'una gran quantitat de fonts d'informació sobre molts paràmetres permet establir sistemes de presa de decisió, que poden ser manuals, automàtiques o mixtes. Aquestes decisions poden influir en elements molt locals, com la calefacció d'una vivenda fins a tot el sistema de gestió del trànsit d'una gran ciutat. Habitualment la presa de certa decisió va lligada a l'execució d'algunes accions, tornant a entrar d'utilitat de Internet a les coses, IoT d'ara en endavant, ja que per executar accions cal que certs actuadors s'activin, i es imprescindible poder comunicar de forma eficient amb ells.

A continuació es posen alguns exemples d'aplicacions:

- A nivell molt local, es podria llegir la dada de temperatura d'una vivenda, decidir si es vol activar la calefacció o aire condicionat i donar l'ordre des de qualsevol lloc d'activació d'algun element.

- A nivell d'un edifici d'oficines, es podria automatitzar l'adquisició de l'estat de les llums i presencia de persones en totes les sales, i automatitzar l'encesa o apagada de les llums segons la presencia o no de personal.

- A nivell metropolità, es poden adquirir l'estat dels aparcaments disponibles, i distribuir aquesta informació en panells informatius per tal de guiar el trànsit, i així reduir la quantitat de vehicles que busquen aparcament al disminuir els temps de cerca al disposar d'informació.

Tal com s'observa, adquirir informació i actuar sobre elements, ofereix una immensa possibilitat d'implementació de nous sistemes que poden canviar les metodologies establertes de treball i actuació en molts àmbits.

c. Estàndards utilitzats

Els sistemes de IoT solen requerir de sistemes de comunicacions que connectin els sistemes electrònics i permetin l'enviament i recepció de dades. Donada la naturalesa dels sistemes IoT, se sol optar per sistemes flexibles que requereixin una mínima infraestructura, sent les electròniques de baix consum alimentades amb piles o bateries i les comunicacions radio, les més utilitzades per aquesta funció.

El tipus de comunicacions dels sistemes IoT tenen unes característiques peculiars, ja que requereixen d'elevades autonomies, alta flexibilitat, comunicacions robustes i multicamí, i la quantitat de dades a transmetre és reduïda.

Tenint en compte aquests requeriments, la industria ha desenvolupat diversos protocols inalàmbrics de curt abast (per dotar l'elevada autonomia a l'emetre amb baixa potencia), amb velocitats de transmissió mitjanes i petites (per no requerir d'electròniques potents) i amb capacitats multicamí per poder cobrir extensions elevades repetint missatges de remota en remota.

Els protocols estàndards més comuns són el Zigbee, el IEEE802.15, el 6loWPAN i el WirelessHart.

2.2. Hardware per Internet a les Coses

a. Requisits del hardware per IoT

Tal com s'ha comentat anteriorment les aplicacions d'IoT tenen un requisits molt particulars en el hardware a utilitzar que es llisten a continuació:

- Consums molt reduïts, que proporcionin autonomies molt elevades amb sistemes d'alimentació mitjançant piles o bateries. Sol ser aconsellable que els processadors permetin passar a un mode de repòs (sleep mode) en el que pràcticament tot el hardware s'atura, excepte certes interrupcions.
- Capacitat de processament petites, habitualment les aplicacions IoT no solen requerir d'elevats ratis de processament, motiu pel qual solen funcionar amb rellotges de kHz, no MHz. Velocitats baixes de rellotge permeten majors autonomies.
- Capacitat per adquirir dades de l'exterior i actuar, que pot ser mitjançant senyals digitals, analògiques o des de busos de comunicacions. Això permet que les aplicacions IoT interactuïn amb l'entorn.
- Capacitat per comunicar entre diferents equips, sent ideal les comunicacions sense fils per guanyar flexibilitat
- Costos de fabricació reduïts, ja que costos petits permetran la instal·lació d'electrònica en objectes quotidians i de baix cost, sense que el preu final quedi afectat en excés.

b. Hardware disponible al mercat

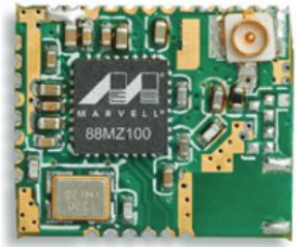
Els diversos fabricants han desenvolupat solucions específiques per les aplicacions de IoT, que habitualment solen ser microcontroladors que integren en el mateix xip una CPU, memòria RAM, ROM, comunicacions sense fils i hardware d'entrades-sortides. Aquests xips estan optimitzats per treballar amb consums molt petits de pocs mA, i incorporen funcions de repòs que disminueixen en consum a pocs uA.

Alguns dels xip que es troben al mercat poden ser els següents:

- La gama Texas Instruments CCXXXX, que incorpora una CPU ARM, RAM, ROM, entrades i sortides i una radio de 2,4GHz. Disposa de diferents models segons els protocols possibles en que pot comunicar la radio, podent ser Wi-FI, Bluetooth o IEEE802.15 / Zigbee.

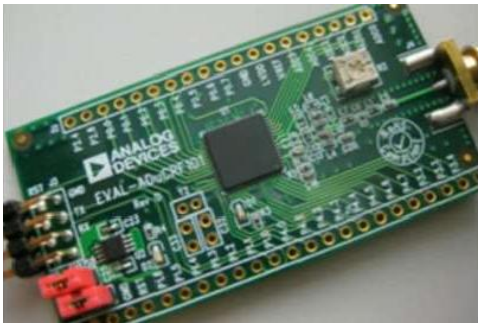
- La gama ST Microeectronics STM32W, que incorpora una CPU ARM, RAM, ROM, entrades i sortides i una radio de 2,4GHz que permet comunicacions IEEE802.15 / Zigbee.

- La gama Marvell 88MXXX, incorpora una CPU ARM, RAM, ROM, entrades i sortides i una radio de 2,4GHz que segons versió pot comunicar amb Wi-Fi, Blueetooth o comunicar IEEE802.15 / Zigbee.



- Intel disposa de la gama Edison/Quark, pensada per IoT, però que incorpora processadors x86 bastant potents i aposta més per les comunicacions Wi-Fi.

- La gama Analog Devices ADuCXXX, incorpora una CPU ARM, RAM, ROM, entrades i sortides i una radio de 431 a 464MHz



Tal com s'observa la industria evoluciona a oferir xips que incorporin tota la circuiteria necessària incorporada, inclòs la radio per les comunicacions sense fils.

També es bastant comú la utilització de l'arquitectura ARM, pràcticament tots els fabricants, excepte Intel, aposten per aquesta arquitectura que permet fabricar xips prou potents i amb uns costos molt continguts.

Per altra banda, s'observa les dues games de productes, les que implementen hardware radio Wi-Fi, que solen muntar amb un rellotge més ràpid, i les que munten radios compatibles amb Bluetooth o IEEE802.15/ Zigbee, que solen utilitzar velocitats de rellotge més reduïdes. Cada tipus de comunicacions, té uns requeriments de hardware i implica uns costos o altres. Per aplicacions amb un enorme nombre d'equips i poca necessitat de processament, les solucions radio IEEE802.15 / Zigbee, resulten més econòmiques i amb més autonomia, però requereixen de passarel·les per convertir la informació a protocols TCP/IP i comunicar amb sistemes pc.

2.3. Protocols de comunicacions per Internet a les Coses

Tal com s'enunciava a l'apartat 2.1, en l'entorn IoT s'estan marcant una sèrie de protocols com estàndards, i la majoria de fabricants els permeten en el seu hardware. La majoria d'aquests protocols permeten comunicacions sense fils, de forma que s'augmenta la flexibilitat en el seu muntatge al no requerir d'infraestructura.

Els protocols més comuns es mostren en les imatges anterior, i cada un proporciona diferents avantatges. A continuació analitzarem cada un d'ells, amb les avantatges e inconvenients que proporciona.

Wi-Fi: És una tecnologia de xarxa local sense fils que permet a un dispositiu electrònic intercanviar dades ja sigui a 2.4 GHz o 5 GHz. L'aliança Wi-Fi ho descriu com qualsevol "producte wifi de xarxa local basat en l'estàndard 802.11a/b/g/n/ac de la IEEE". Hi ha molts dispositius habilitats per utilitzar comunicació Wi-Fi: ordinadors, impressores, videoconsoles, smartphones, càmeres digitals, tablets i reproductors digitals multimèdia. El rang que cobreix el sistema wifi té uns 20 metres en interior fins a alguns centenars en camp obert, i arribar a diversos kilòmetres utilitzant antenes direccionals. Poden incorporar sofisticats mètodes d'encriptació per proporcionar comunicacions segures. En l'àmbit IoT, sobre aquesta tecnologia es solen muntar comunicacions TCP/IP, i per tant solen ser molt fàcils d'integrar amb grans xarxes sense cap passarel·la de conversió de protocol, sent la seva principal avantatge. Lamentablement sol requerir d'una CPU relativament ràpida per gestionar les comunicacions i no disposa de

mètodes d'estalvi d'energia, fet que dificulta aconseguir molt baix consum utilitzant aquesta tecnologia.

Wi-Fi	
Avantatges	Inconveniets
Comunicacions segures	No disposa de sistemes d'estalvi d'energia
Fàcil integració TCP/IP	Requereix de processadors relativament ràpids
Velocitat de transferència elevada	Cost elevat de l'electrònica comparat amb altres protocols
Rang en exteriors de centenars de metres	
Fàcil integració d'elements sense passarel·les específiques	
Altament suportat per la industria	

Bluetooth: És una especificació industrial per les Xarxes d'Àmbit Personal (PAN, Personal Area Network) sense fil, amb la que podem obtenir una forma de connectar i intercanviar informació entre dispositius com ordinadors de butxaca, telèfons mòbils, ordinadors portàtils, ordinadors, impressores i càmeres digitals a través d'una forma segura, de baix cost a través d'ones de ràdio de baixa freqüència. El Bluetooth permet a aquests dispositius comunicar-se quan es troben a l'abast, encara que no estiguin a la mateixa habitació, fins a un límit de 100 metres entre ells depenent de la classe de potència del producte. Els productes poden estar disponibles en alguna d'aquestes tres classes de potència, Classe 3 (1 mW) és la més rara, i ens permet una transmissió de 10 centímetres a un màxim d'1 metre, Classe 2 (2.5 mW) és la més comuna i ens permet una distància de fins a 10 metres, i Classe 1 (100 mW) té l'abast més gran, de fins a 100 metres tot i que el consum és més elevat. A nivell de seguretat, permet incorporar encriptacions de 128bits, suficients per proporcionar seguretat i no requerir hardware potent. En l'àmbit IoT, no se solia utilitzar massa aquesta tecnologia, ja que l'abast que té és petit amb els classe2 i els classe 1 consumeixen massa energia, no sent idoni per la majoria d'aplicacions IoT. Ara bé, el sector IoT l'està començant a utilitzar implementant la variant del protocol en versió de baixa potencia (Bluetooth low energy, Bluetooth LE, BLE, Bluetooth Smart). La versió 4.0 de l'especificació Bluetooth ja l'incorpora BLE i es preveu que la majoria del hardware Bluetooth de manera molt progressiva vagi adquirint aquests sistemes de baixa potència.

Bluetooth	
Avantatges	Inconveniets
Comunicacions mitjanament segures	No disposa de sistemes d'estalvi d'energia. En un futur si els tindrà
Possibilitat d'implementar una pila TCP/IP	Abast de les comunicacions petit
Velocitat de transferència mitjana. La versió de baixa potencia implementa velocitats baixes.	Requereix de passarel·les específiques
Cost moderat de l'electrònica comparat amb altres protocols	
Creixent suport de la industria IoT amb la versió Low Power	

ZigBee/IEEE802.15: ZigBee és un protocol basat en l'especificació IEEE802.15. Els dos es basen en comunicacions sense fils de baixa velocitat i baix consum, amb un rang de fins a 100m i molt optimitzats per permetre muntar la xarxa en malla, de forma que sigui factible incrementar el rang a base d'anar repetint els missatges d'un equip en un altra. Els dos protocol operen en diverses bandes de freqüència 2.4 GHz en tot el mon, 784 MHz a Xina, 868 MHz a Europa i 915 MHz a EEUU i Australia. Les velocitats de transmissió varien entre 20 kbit/s (banda de 868 MHz band) fins a 250 kbit/s (banda de 2.4 GHz). Les electròniques radio estan molt pensades per requerir de pocs components i tenir un preu molt reduït, a la vegada que proporcionin uns consums molt baixos. A nivell de seguretat, els protocols permeten encriptacions de fins a 128 bits, que proporcionen seguretat suficient sense requerir d'electròniques complexes. Tal com s'observa, aquest protocol és el que més s'ajusta als requeriments dels sistemes IoT, i es per això que està sent àmpliament utilitzat en aquest entorn, on el baix cost i l'elevada autonomia són vitals.

ZigBee/IEEE802.15	
Avantatges	Inconvenients
Comunicacions mitjanament segures	Velocitats de transmissió baixes
Possibilitat d'implementar una pila TCP/IP	Abast de les comunicacions petit
Disposa de sistemes d'estalvi d'energia.	Requereix de passarel·les específiques
Cost baix de l'electrònica comparat amb altres protocols	
Rang ampliable utilitzant tipologies de xarxa on els nodes fan de repetidor	
Suport de la industria IoT	

Existeixen altres protocols que utilitzen les radios de l'estàndard IEEE802.15, com el 6LoWPAN o WirelessHART. El primer afegeix una capa per sobre de les comunicacions IEEE802.15 per implementar una pila IPv6, fent més fàcil la connexió d'equips a grans xarxes utilitzant una passarel·la específica. El WirelessHART, és un protocol molt utilitzat en instrumentació, i és una implementació de les trames HART cablejades sobre radio, de forma que s'obtingui major facilitat de connexió i menor necessitat de cablejat i infraestructura.

2.4. Software per Internet a les Coses

a. Programació dels microcontroladors

La majoria de fabricants permeten la programació dels seus equips en C o C++, de forma que sigui factible la reutilització del codi i a la vegada no sigui necessari programar en llenguatge assemblador. Això permet programar de forma molt més ràpida.

La majoria de fabricants proporcionen exemples per poder direccionar el hardware, aportant llibreries exemple per fer algunes funcions bàsiques, però no solen incorporar funcions complexes, obligant als usuaris a desenvolupar molt de codi per fer les funcions desitjades.

Altres fabricants, com OpenMote Technologies, implementen en les seus productes xips de grans fabricants, i aporten com a valor afegit un firmware més elaborat que proporcionen llibreries orientades a objecte per controlar pràcticament tot el hardware, llibreries per implementar sistemes de temps real i moltes aplicacions exemple.

Finalment, donat que la majoria de xips tenen arquitectura ARM, s'han desenvolupat petits sistemes operatius que ja tenen implementats moltes funcions de control de hardware, llibreries avançades i protocols de comunicacions. Aquest tipus de solucions han anat proliferant els últims anys, ja que els requisits dels sistemes IoT no s'acaben de cobrir amb els sistemes operatius convencionals, i això ha generat la oportunitat dels nous sistemes, molts d'ells han estat generats per l'entorn universitari sota llicencia OpenSource. A continuació s'analitzaran quatre de representatius, el OpenWSN, el Contiki, TinyOS i el RiOT.

b. Contiki

Es un sistema operatiu Open Source molt complet, pensat per proporcionar un entorn de desenvolupament d'aplicacions, amb gran nombre de llibreries, protocols de comunicacions ja programats, sistemes de simulació i compatibilitat amb un gran nombre de xips hardware.

El llistat de hardware compatible fins el moment és el següent:

Xip	Radio	Plataforma	Suport simulació Cooja
RL78	ADF7023	EVAL-ADF7023DB1	-
TI CC2538	Integrada	cc2538dk	-
TI MSP430x	TI CC2420	exp5438, z1	Sí
TI MSP430x	TI CC2520	wismote	Sí
Atmel AVR	Atmel RF230	avr-raven, avr-rcb, avr-zigbit, iris	-
Atmel AVR	TI CC2420	Micaz	Sí
Freescale MC1322x	Integrada	redbee-dev, redbee-econotag	-
ST STM32w	Integrada	mb851, mbxxx	-

Xip	Radio	Plataforma	Suport simulació Cooja
TI MSP430	TI CC2420	sky, jcreate, sentilla-usb	Sí
TI MSP430	TI CC1020	msb430	-
TI MSP430	RFM TR1001	esb	Sí
Atmel Atmega128 RFA1	Integrada	avr-atmega128rfa	-
Microchip pic32mx795f512l	Microchip mrf24j40	seed-eye	-
TI CC2530	Integrada	cc2530dk	-
RC2300/RC2301	Integrada	sensinode	-
6502	-	apple2enh, atari, c128, c64	-

Tal com s'observa, aquest sistema operatiu és compatible amb molt xips, fet que proporciona certa independència del hardware a les aplicacions desenvolupades sobre ell.

A més aquest sistema operatiu, proporciona l'entorn instant Contiki que integra en un únic paquet descarregable l'entorn de programació en C i el simulador Coojan, que permet emular i provar certs programes desenvolupats.

Per altra banda, també disposa del següent llistat de característiques que aporten funcions molt pensades per sistemes IoT:

- Gestió de memòria optimitzada, utilitzant pocs kbytes i permeten gestió de memòria
- Comunicacions IP programades, implementant piles IP, i poden utilitzar comunicacions TCP,UDP, HTTP i inclòs 6lowpan, RPL i CoAP.
- Implementa sistemes d'estalvi d'energia, per augmentar autonomies
- Càrrega de mòduls dinàmica quan s'està executant el programa
- Permet programació per fils (multi-threaded)
- Permet implementar una línia de comandaments (shell)
- Implementa piles per reenviar missatges a remotes veïnes, ampliant així els rangs de cobertura de les xarxes.

Sense cap mena de dubte aquest sistema operatiu és molt potent, **però lamentablement no té implementat el sistema de comunicacions TSCH**.

c. OpenWSN

OpenWSN no es pot considerar un sistema operatiu, és més una implementació de protocols de comunicació, que permeten que les aplicacions C desenvolupades amb ell comuniquin segons tot una sèrie de protocols estàndards.

Els diversos protocols implementats són els següents:

- IEEE802.15.4e, **incloent el nou sistema TSCH**.
- IETF 6TiSCH, implementant una pila IPv6 sobre comunicacions IEEE802.15.4e TSCH
- IETF 6LoWPAN, implementant piles IPv6 sobre trames IEEE802.15.4
- IETF ROLL, que implementa el protocol d'enrutació RPL, que permet el multi salt dels missatges a través dels esclaus.

Aquestes llibreries funcionen en els xips llistats a continuació:

- TelosB
- GINA
- WSN430
- Z1
- OpenMoteCC2538
- OpenMoteSTM
- SAM R21 Xplained Pro
- USP Mote MC13213
- USP Mote CC2538
- IoT-LAB_M3
- AgileFox

d. TinyOS

Aquest sistema operatiu està dissenyat per proporcionar abstracció del hardware al dissenyador, gestionant la memòria, permetent programació per fils, aportant sistemes d'adquisició de senyals i comunicant amb protocols estàndards, incorporant una pila IPv6 6lowpan/RPL.

El hardware compatible amb el TinyOS són els microcontroladors Texas Instruments MSP430, Atmel's Atmega128, Atmega128L i Atmega1281, i la Intel px27ax. S'està treballant pr soportar el Cortex M3.

Els xips radio que suporta són el Texas Instruments/ChipCon CC1000 i CC2420, el Infineon TDA5250, el Atmel RF212 i RF230, i el Semtech XE1205.

e. RiOT

Aquest sistema operatiu disposa de les següents característiques:

- Programació en C i C++
- Minimització de la dependència del hardware
- Codi operatiu en xips de 8 bits, 16 bits i 32 bits
- Programació per fils
- Gestió de memòria
- Protocols implementats 6LoWPAN, IPv6, RPL, TCP i UDP
- Protocols CoAP i CBOR
- Línia de comandaments i rutines SHA256

f. Ús en el projecte

Tal com s'observa, existeixen diverses opcions de sistemes operatius, però només un d'ells té desenvolupades les llibreries TSCH.

En el projecte es treballarà directament sobre el firmware del producte, i no s'utilitzarà cap sistema operatiu existent.

2.5. Introducció al IEEE 802.15.4e

a. Introducció

El 16 d'abril de 2012 la IEEE va publicar la norma IEEE 802.15.4e, en la que es definia millor que en normes anteriors un nou protocol de control d'accés al medi (MAC) i el nou mode TSCH (Timeslotted Chanel Hopping), que en català pot traduir-se com salt de freqüència a temps ranurat.

Aquest nou mode és compatible amb tot el hardware compatible IEEE 802.15.4, però afegeix un mode de funcionament basat en sincronització de temps, per realitzar comunicacions controlades que permeten activar l'electrònica radio només en certs instants, fet que permet un estalvi energètic important, i en un sistema de salts de canal de comunicacions, pensat per augmentar la fiabilitat de les comunicacions, ja que els 16 canals disponibles en la banda de 2,4GHz són utilitzats per diversos protocols, fet que pot generar interferències, i el fet de anar alternant de canal afavoreix la immunitat a interferències.

El protocol TSCH s'ha dissenyat per permetre una gestió per capes, de forma que únicament es centra en la capa MAC, permeten utilitzar per sobre altres sistemes, sent les piles IPv6 les que segurament més utilització tindran en el futur.

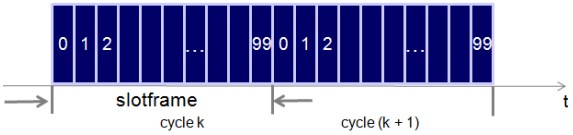
Cal destacar que el TSCH proporciona procediments per comunicar de forma òptima energèticament i de forma fiable, però no implementa sistemes de detecció de veïns, detecció de tipologia de xarxa o sistemes d'enrutació. Aquestes funcions les executen altres protocols que estan fora de l'abast del present projecte.

b. TSCH

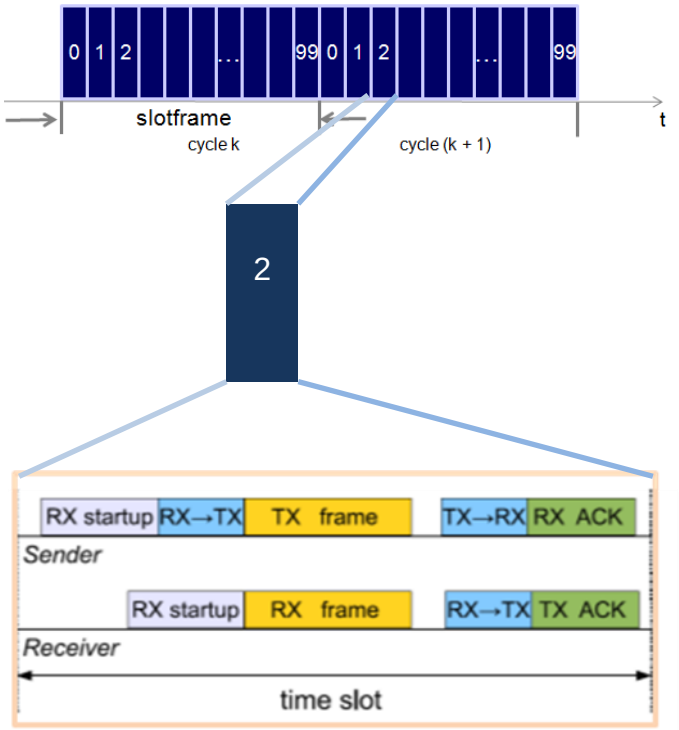
Aquest protocol es basa en dos característiques que s'utilitzen a la vegada, el Timeslotted (TS) i el Channel Hopping (CH).

El TS, separa el temps en ranures, de forma que cada remota només comunica en la ranura que té assignada, tenint la radio desconectada la resta del temps, i aconseguint un important estalvi energètic. Donat aquest sistema, el protocol és determinista, ja que no hi ha col·lisions entre remotes i es pot saber exactament quan arribarà un missatge. Totes les remotes van sincronitzades en temps i disposen de sistemes de correcció de petites diferències temporals entre remotes.

El sistema defineix una trama (slotframe), que està formada per un nombre de ranures, i que es repeteix en el temps. Cal dir que el protocol no determina el nombre de ranures que s'utilitzen en un slotframe, sent l'usuari o protocols superiors qui defineixen aquest paràmetre.



Cada trama està formada per ranures, que internament està formada per diverses fases, que en termes generals es poden agrupar en fases de preparació, enviament o recepció i confirmació de recepció.



El CH, salt de freqüència, té en compte dos valors per decidir quin dels 16 canals disponibles a la banda de 2,4GHz s'utilitza per comunicar a cada ranura, el primer és l'ASN, un comptador de trames que segons la norma es forma amb 5 bytes, per tant pot arribar fins a 2^{40}, i un paràmetre de canal offset. La fórmula utilitzada és sumar primer l'ASN i el canal offset, i dividir el resultat entre 16, la resta de la divisió determina el canal a utilitzar. Donat que l'ASN s'incrementa en cada ranura, la freqüència utilitzada en cada comunicació va variant, aportant una gran fiabilitat a les comunicacions front les interferències. Cal tenir en compte que els canals en la banda de 2,4GHz, són 16 i van del canal 11 al canal 26. El paràmetre channel offset, permet que es facin comunicacions simultànies entre diverses remotes a la vegada i en la mateixa ranura, ja que utilitzaran canals diferents.

La norma estableix que un coordinador proporcionarà l'ASN als nous membres de la xarxa per tal que es puguin sincronitzar, ja que el nombre ASN són absolutament necessaris per poder comunicar amb aquest sistema.

3. Desenvolupament del projecte

3.1. Descripció dels objectius del projecte

L'objectiu principal del projecte és desenvolupar una llibreria que implementi el protocol TSCH sobre la plataforma OpenMote, programat directament amb llenguatge C++, sense utilitzar cap sistema operatiu encastat. Aquesta llibreria aprofitarà una llibreria existent de comunicacions radio, que envia una trama pel canal de comunicacions seleccionat.

La llibreria desenvolupada ha de permetre la implementació futura d'altres piles de comunicacions sobre ella, per tant ha de disposar de dues funcions principals, una d'enviament i una altre de recepció de trames. Aquestes funcions no han d'implementar cap protocol d'enrutació, només enviar trames implementant els mètodes de ranures de temps i selecció de canal, comportant-se de forma transparent per les piles superiors.

A part, la llibreria desenvolupada haurà de disposar d'una rutina que s'executi de forma temporitzada, per un contador o per una interrupció, que s'encarregui de gestionar el control de l'ASN, la gestió de ranures i canals, i les tasques de sincronització entre remotes.

El projecte haurà de desenvolupar i provar un codi exemple d'utilització d'aquesta llibreria desenvolupada capaç de comunicar dues remotes OpenMote que la UOC ha prestat.

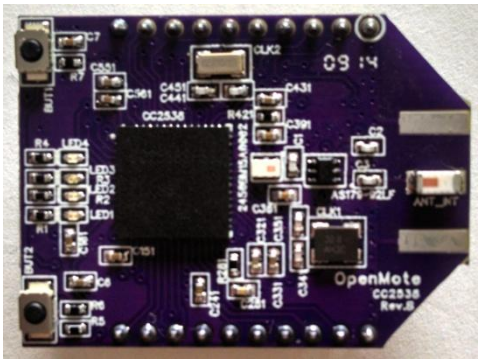
3.2. Plataforma OpenMote

a. Introducció

El projecte es desenvoluparà sobre la plataforma hardware OpenMote, que es pot subdividir en els 3 hardwares específics següents:

- OpenMote-CC2538

Aquest és el component principal de tota la solució, ja que implementa el cor de la solució, al incloure la CPU, la radio, el cristall oscil·lador, l'estabilitzador de tensió, 4 leds de colors, 2 botons i una antena integrada. De fet els altres dos components de hardware representen únicament plaques que permeten certes ampliacions, com certs ports de comunicació o diferents sistemes d'alimentació.



Font: www.openmote.com

- OpenBase

Aquest hardware proporciona ports addicionals i un sistema d'alimentació mitjançant un cable USB a l'OpenMote-CC2538. Concretament proporciona un port JTAG per poder fer depuracions, un port USB direccionable amb un pc com a port sèrie gràcies al xip FTDI FT232R, facilitant les comunicacions ampliacions i les programacions, i finalment un port 10/1000Mbps Ethernet per donar connexió a xarxes.

L'objectiu d'aquest hardware és doble, un es per donar una placa de programació i depuració senzilla i totalment necessària per desenvolupar aplicacions, i la segona es permetre executar tasques de passarel·la entre diverses xarxes amb protocols diferents, sent un element molt útil quan es molt utilitzar l'OpenMote connectat com a coordinador i passarel·la per transmetre informació. Tal com s'observa, aquesta placa disposa d'uns connectors que permeten allotjar directament els pins del OpenMote, sent fàcilment intercanviables.

Font: www.openmote.com

- OpenBattery

El tercer hardware està totalment pensat pel muntatge de xarxes distribuides de sensors alimentades amb petites bateries o piles i amb autonomies de mesos o anys. Incorpora un allotjament per piles AAA i tres sensors, un de temperatura i humitat, un altre d'acceleració i un luxòmetre. Gràcies a aquest hardware es poden implementar gran xarxes de sensors amb una o diverses funcionalitats i transmetre o rebre informació útil d'ells. Tal com s'observa, aquesta placa disposa d'uns connectors que permeten allotjar directament els pins del OpenMote, sent fàcilment intercanviables.

Font: www.openmote.com

b. Detall del hardware

Es detallarà amb molta més informació el hardware OpenMote, ja que es la base de tot el sistema i en el que es pot aprofundir molt més. Concretament un OpenMote està format pels següents elements:

- CC2538, un microcontrolador Cortex-M3 de 32 bits amb l'emisor radio CC2520
- TPS62730, conversor DC-DC que permet convertir les tensions de 3V a 2,1V
- ABM8G, cristall oscil·lador de 32MHz
- ABS07, cristall de 32768kHz de major precisió utilitzat pel rellotge de temps real
- 4 Leds de colors
- 2 botons
- Antena integrada de 0dBi, incloent connector per antena externa opcional

Microcontrolador

El microcontrolador que munta el OpenMote és un Texas Instruments CC2538, un microcontrolador d'arquitectura ARM de 32 bits que segueix

el disseny Cortex-M3. Aquest processador està desenvolupat per proporcionar unes elevades prestacions, amb baix consum i a un preu molt contingut, considerant-se ideal per aplicacions d'Internet a les Coses. Funciona mitjançant un rellotge de 32kHz (velocitat clarament orientada a optimitzar l'estalvi energètic més que no pas un rendiment molt elevat), i implementa mitjançant interrupcions diversos modes de funcionament que permeten deixar inactiu algunes parts o tota la CPU per aconseguir mitjanes de consum molt reduïdes.
A més implementa tot una sèrie de perifèrics que el permeten interactuar amb l'exterior, aconseguint i comunicant informació, entre els que destaquen 4 temporitzadors, 8 entrades conversores analògic digital de 12 bits, 2 UART, 2xSPI i 1 port USB.
D'especial menció és la unitat de radiofreqüència que incorpora que es detalla a continuació, però que aporta al microcontrolador capacitat per comunicar amb ones electromagnètiques, aportant una gran flexibilitat pel muntatge de xarxes amb aquest microcontrolador, sense requerir infraestructura. A més, el mateix microcontrolador disposa de hardware específic per fer encriptacions i establir comunicacions segures de forma òptima i amb consums d'energia molt reduïts.

Emisor radio 2,4GHz

El mateix microcontrolador implementa a l'interior el xip CC2520 de Texas Instruments, que és un emissor i receptor de radiofreqüència que treballa a 2,4Ghz (vàlid per tot el mon). Aquest mòdul permet establir comunicacions segons les normes IEEE 802.15.4 i Zigbee, utilitzant els 11 canals disponibles a la banda de 2,4Ghz, amb unes velocitats màximes de 250kbps, uns consums ajustats i amb possibilitat de posar en suspensió el xip per baixar encara més el consum.

c. Detall del firmware

El hardware OpenMote està preparat i provat per carregar diversos sistemes operatius o nuclis de software ja desenvolupats que proporcionen el suport i llibreries per poder programar aplicacions a un nivell més alt, ja que part de les funcions més bàsiques ja es troben ja implementades. Els sistemes operatius que es capaç de carregar són el OpenWSN, Contiki i RIOT. Tots ells es distribueixen gratuïtament sota llicencia OpenSource.

A part del sistemes indicats anteriorment, el fabricant OpenMote Technologies, proporciona un firmware amb una sèrie de llibreries ja implementades que permet programar aplicacions en C++. Aquest firmware no constitueix un sistema operatiu, és més un conjunt de llibreries que ens permeten direccionar el hardware molt més fàcilment des d'una aplicació. A més, aquest firmware, incorpora el kernel FreeRTOS, que permet programar tasques que s'executin cada x ms, pensat per desenvolupar aplicacions amb fils (threads).

Tot el firmware de OpenMote és pot descarregar de la direcció https://github.com/OpenMote/firmware.

Resulta especialment interessant la carpeta projects, on es troben codis exemple molt simples, que mostren com direccionar els diversos components del hardware o el kernel FreeRTOS. En aquest projecte, s'han utilitzat especialment els següents codis:

- ✓ Test-radio, que mostrava com enviar i rebre informació per la radio de 2,4GHz
- ✓ Test-uart, que mostrava com enviar trames pel port RS232 de l'OpenBase
- ✓ Test-sleepTimer, que mostra com treballar amb el rellotge RTC de precisió

El firmware també conté els codis Python, per tal d'enviar programes compilats a un OpenMote muntat sobre una OpenBase per un port USB. L'operativa a seguir per programar s'especifica en l'apartat següent.

d. Preparació de l'entorn de programació

Per disposar un entorn de programació caldrà seguir els passos següents:

- Descarregar el sistema operatiu Ubuntu 14.04 LTS.
- Instal·lar en una màquina física o virtual el sistema operatiu amb la configuració estàndard

Un cop tinguem el sistema operatiu en marxa, caldrà instal·lar les eines per compilar codi C/C++ per les plataformes ARM Cortex i l'entorn Python que utilitzen les rutines de programació per port sèrie. Per tant caldrà connectar la màquina a Internet, obrir un terminal i executar els següents comandos:

- sudo add-apt-repository -y ppa:terry.guo/gcc-arm-embedded sudo apt-get update sudo apt-get dist-upgrade
- sudo apt-get install build-essential gcc-arm-none-eabi libnewlib-arm-none-eabi python python-serial
- wget http://mirrors.kernel.org/ubuntu/pool/universe/libs/libstdc++-arm-none-eabi/libstdc++-arm-none-eabi-newlib_4.8.3-11ubuntu1+4_all.deb sudo dpkg -i libstdc++-arm-none-eabi-newlib_4.8.3-11ubuntu1+4_all.deb

Un cop acabades les instal·lacions, caldrà descarregar el firmware. Això es pot fer a mà descarregant un zip o amb la instrucció següent des d'un terminal.

- git clone --recursive https://github.com/OpenMote/firmware.git OpenMote/firmware

I amb això s'haurà acabat la preparació de la màquina i es podria reproduir l'entorn de programació utilitzat en aquest projecte.

Finalment caldria copiar la carpeta del projecte amb el codi font entregat a una carpeta dintre de firmware/projects i executar les dues instruccions següents, una per complicar i l'altra per enviar el codi pel port sèrie. És important executar la segona instrucció com a root, o assegurar-se que l'usuari emprat té permís per accedir al port USB, si no fallarà la càrrega.

```
cd ruta del projecte
make TARGET=cc2538 all
make TARGET=cc2538 bsl
```

Es important que l'OpenMote tingui el boot loader carregat abans d'enviar el programa, ja que sinó no acceptarà el programa nou. Per posar-lo en aquest mode, cal fer un pont entre el terminals ON/SLEEP i GND i preme el botó reset. Si no s'encenen els quatre led's, caldrà fer un pont del terminal RTS/AD6/DIO6 i GND i preme el botó de reset.

3.3. Implementació IEEE 802.15.4e TSCH

a. Introducció

Un cop detallat les característiques del protocol a implementar i de les possibilitats del firmware, es detallaran les tasques per implementar el protocol en el Openmote.

La configuració de les remotes es farà automàticament, els esclaus només requeriran una identificació numèrica única, i el coordinador serà el que tingui el número de ranures i el channel offset. Quan un esclau demana sincronitzar-se al coordinador, aquest envia l'ASN, el nombre de ranures de la trama i verifica la matriu de connexions i li assigna una ranura lliure i un channel offset.

Donat que el procés de comunicació està format per una sèrie de processos que l'executen de forma seqüèncial, la manera més senzilla per realitzar el programa es implementar una màquina d'estats, que executi diferents tasques dintre d'una ranura.

Aquesta màquina d'estats anirà saltant d'una tasca a l'altra de forma sincronitzada utilitzant les interrupcions del temporitzador RTC. El funcionament de forma sincronitzada és un requeriment del protocol, ja que per tal de baixar el consum energètic les comunicacions, els equips radio s'han de connectar i parar en moments molt concrets.

La màquina d'estats anirà commutant d'un estat a un altra segons les interrupcions del RTC, i això permet implementar sistemes correctius de temps que facin que totes les remotes sincronitzin els rellotges interns

per tal de connectar i desconnectar la radio en el moment precís.

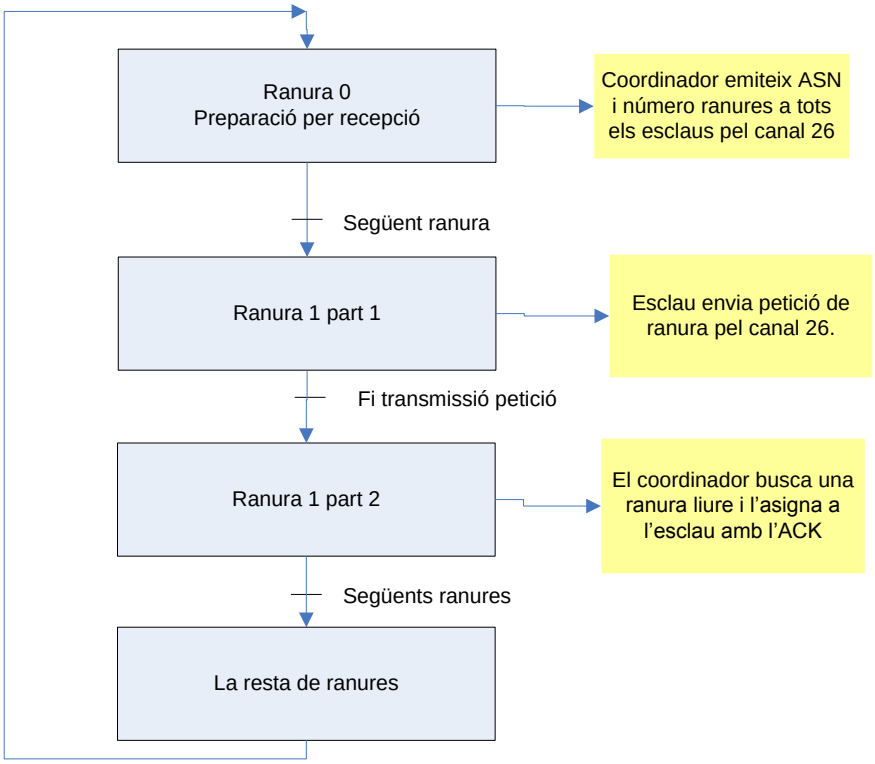
El projecte farà que el coordinador no realitzi cap correcció de temps, sent els esclaus els que realitzaran els ajustos quan han de comunicar amb el coordinador, de forma que es compensin petits errors de rellotge de cada remota.

b. Procés de sincronització

El coordinador per la seva naturalesa sempre estarà sincronitzat, i serà l'equip que tindrà el rellotge i l'ASN mestre per tota la xarxa. Els esclaus es sincronitzaran amb el coordinador corregint el seu RTC intern i adquirint l'ASN d'ell.

Per fer aquest procés, s'utilitzaran les dues ranures inicials de la trama per fer el procés de sincronització.

- A la ranura 0, el coordinador enviarà pel canal 26 en emissió broadcast els 5 bytes de l'ASN i el nombre de ranures que conté tota la trama. L'esclau si no està sincronitzat, escoltarà els valors i els carregarà a les seves variables internes. Si l'esclau està sincronitzat, utilitzarà l'inici de la recepció del paquet per fer una correcció del seu RTC. Si no rep cap paquet significarà que el contador de ranures està malament i tornarà a activar el procés complert de sincronització.
- A la ranura 1 el coordinador està a l'espera, i un esclau que no estigui a la xarxa enviarà sol·licitud per connectar indicant el seu identificador. El coordinador farà una cerca per la matriu per localitzar ranures lliures, i assignarà la primera ranura perquè l'esclau comuniqui amb el coordinador i la següent perquè el coordinador comuniqui amb l'esclau, juntament amb el channel offset a aplicar. Els esclaus ja assignats no faran res en aquesta ranura.

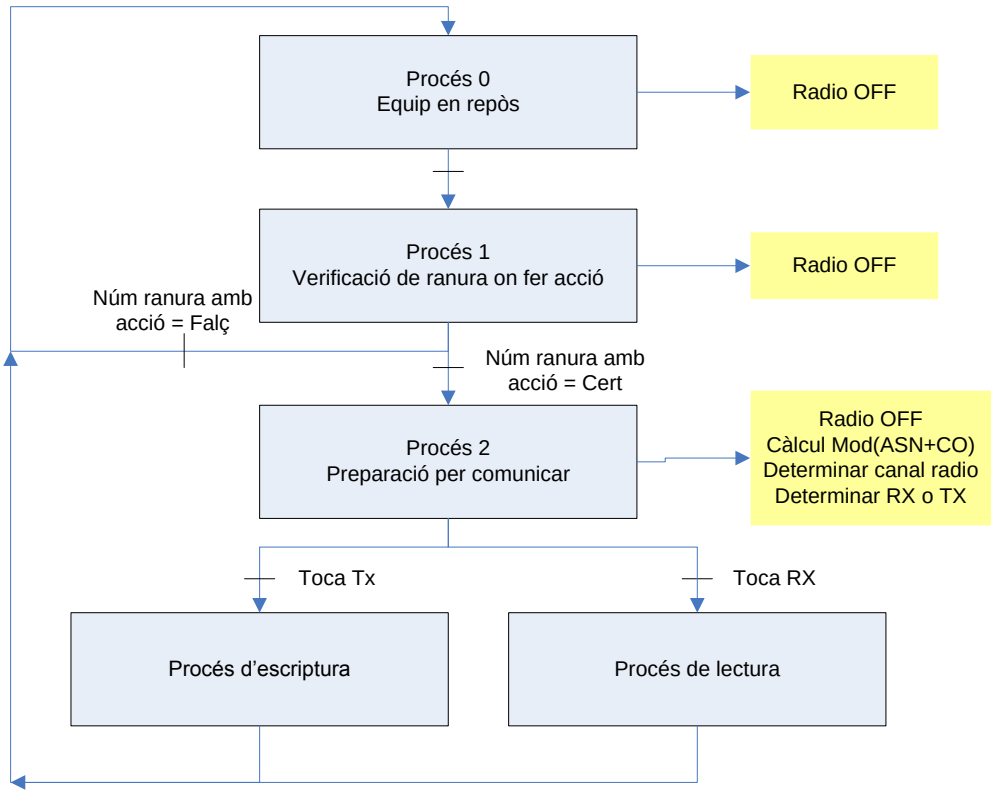


c. Enviament i recepció

Cada ranura tindrà una sèrie de tasques, i s'anirà commutant d'una a altra segons una variable que va registrant l'estat.

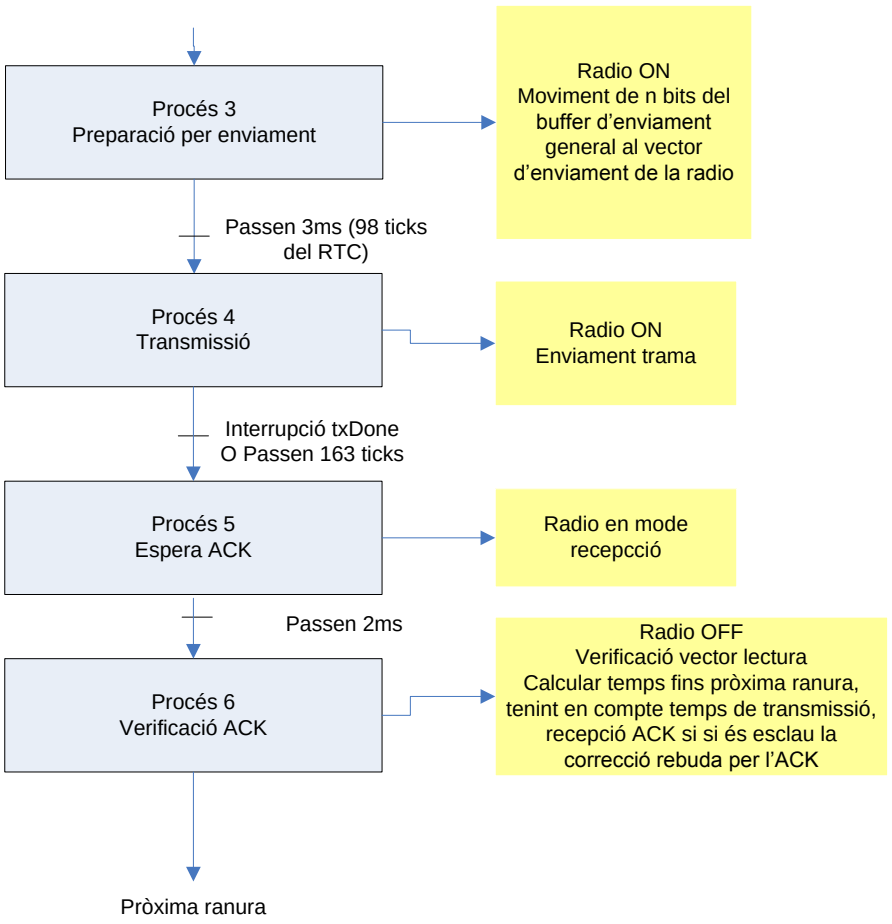
El programa donarà un temps de 12ms a cada ranura, que sól considerar-se la duració típica, ja que una trama de 127 bytes necessita 4,6ms per enviar-se (a una velocitat de 250kbps), i l'ACK sobre 1ms, deixant 6,4ms per gestió de comunicacions i correcció de retards.

Per dibuixar la màquina d'estats de forma clara, s'utilitzarà el grafcet següent:

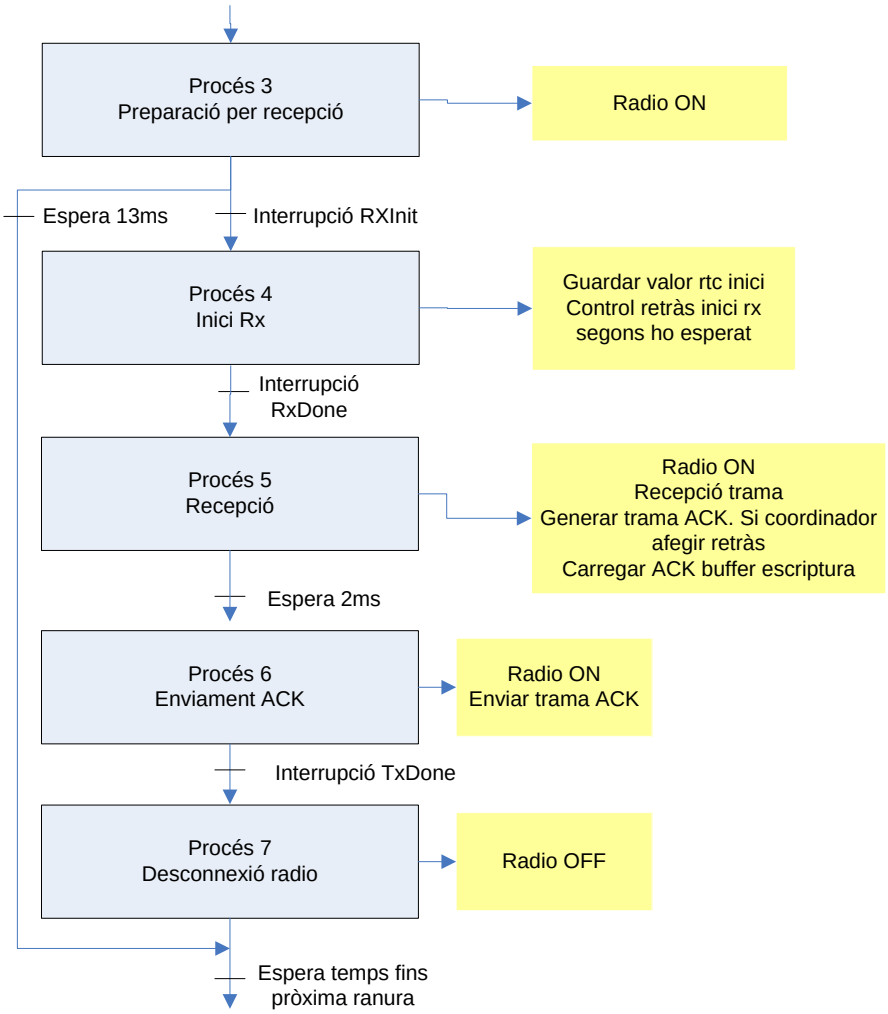


Tal com s'observa es defineixen dos subprocessos (lectura i escriptura) que es detallen a continuació:

Procés d'escriptura



Procés de lectura



d. Explicació detallada

El protocol s'ha programat utilitzant el cristall ABS07, que oscil·la a 32768kHz amb un error de 10ppm, que la cpu utilitza com a RTC. La raó d'utilitzar aquest component és la gran precisió necessària per obrir les ranures en el moment just. El RTC proporciona una mesura fiable amb una precisió de 30us, de forma que es podran fer deteccions i correccions de temps amb aquesta precisió, mantenint el sincronisme de tota la xarxa.

El RTC es capaç de generar una interrupció cada una sèrie de ticks que es programin, i això juntament amb unes variables de control d'estat, permet muntar tot el sistema de sincronització que requereix el TS.

Quan s'inicia el programa s'inicialitzen les variables les següents variables:

- **Sincronitzat** en fals, que indica que la remota no té sincronitzat

l'ASN
- **Assignat** en fals, que indica que no té ranura assignada
- **Estat** a 0, que és el que controla la màquina d'estat.

En cas de ser coordinador, les variables sincronitzat i assignat passen a cert, ja que és el que controla la xarxa.

Quan s'executa la interrupció, el sistema executa les tasques de l'estat 0, que inicialitza la ranura, definint quina acció s'ha d'executar, llegir, escriure o no fer cap acció. En cas de no fer cap acció es carrega la pròxima interrupció en 393 ticks del rellotge, que equival a 12ms.

Si la ranura és d'escriptura, la variable estat adquireix el valor 2, carrega les dades a enviar en el buffer de la radio, dóna el valor 3 a la variable estat i genera una interrupció RTC en 98 ticks (3ms). A la següent interrupció, l'estat igual a 3, fa que s'executi la rutina que transmet el missatge per la radio i canvia l'estat a 4. Quan el paquet acaba d'enviar-se la radio activa la interrupció txDone, que en l'estat 4, desactiva la radio, carrega l'estat 5 i genera una interrupció RTC en 2ms. En l'estat 5, s'activa la radio a l'espera de rebre l'ACK i l'estat 6. Quan la radio acaba de rebre un paquet, s'activa la interrupció rxDone que activa l'estat 7, que analitza l'ACK rebut i aplica la correcció de temps enviada pel coordinador en cas de ser esclau. Com el temps d'enviament depèn de la quantitat de dades a enviar, en l'estat 2 es guarda el valor del RTC i al acabar es torna a llegir, per tal de quadrar el temps a 12ms, es passa l'estat a 0 i es genera una interrupció del RTC a 393-(valor RTC final – valor RTC inicial).

Si la ranura és de lectura, la variable estat adquireix el valor 9 i s'activa la radio en mode recepció. Si no rep cap missatge, la radio es desactivarà i s'executarà la següent ranura. Si rep un missatge, s'executarà la interrupció rxDone, que activa l'estat 11, que envia la trama rebuda pel port sèrie, carrega l'ACK en el buffer de la radio i genera una interrupció en 54 ticks. En l'estat 12, es transmet l'ACK i un cop acabat s'executa la interrupció txDone i es genera una interrupció del RTC igual a 393-(valor RTC final – valor RTC inicial).

e. Planificació de comunicacions

A part de les variables que configuren el nombre de ranures i duració d'elles, es necessari carregar la matriu que determina els ordres de comunicació establerts. Per això, el programa disposa d'una variable que indica el id de cada estació, i una matriu on s'especificarà l'ordre de comunicacions. Aquesta matriu tindrà tantes files com ranures previstes i 3 columnes.

uint8 idesta

uint8 ordrecom [num ranures] [3]

L'element ordrecom[n] [0] = identificador estació emissora
L'element ordrecom[n] [1] = identificador estació receptora
L'element ordrecom[n] [2] = offset de canal a aplicar

La remota quan demana assignació en la ranura 1 al coordinador, aquest li retorna quina ranura ha d'utilitzar per enviar i rebre, així com el channel offset.

Al començament de cada ranura, cada estació verificarà aquesta taula per determinar si ha de comunicar, rebre o quedar-se suspesa.

3.4. Aplicació de proves del sistema

a. Introducció

Per verificar el funcionament de la comunicació s'ha programat una aplicació que utilitza el protocol TSCH desenvolupat, que té una doble funció. La primera es provar com una remota es connecta al coordinador, se li assigna una ranura i envia de forma continua en cada ranura el valor de temperatura, humitat i llum. La segona prova, monitoritzarà el retràs que s'inclou en l'ACK.

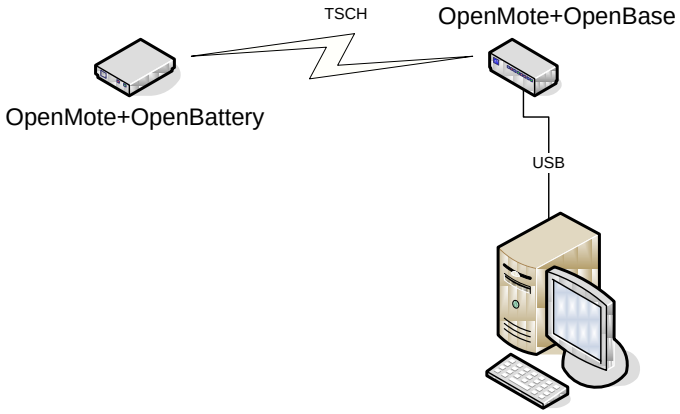
La primera, serveix per verificar la funcionalitat principal del protocol, que és enviar informació, la segona serveix per verificar que la correcció de microsegons que s'envien en l'ACK es manté estable i per tant que funciona correctament.

b. Programa verificació protocol

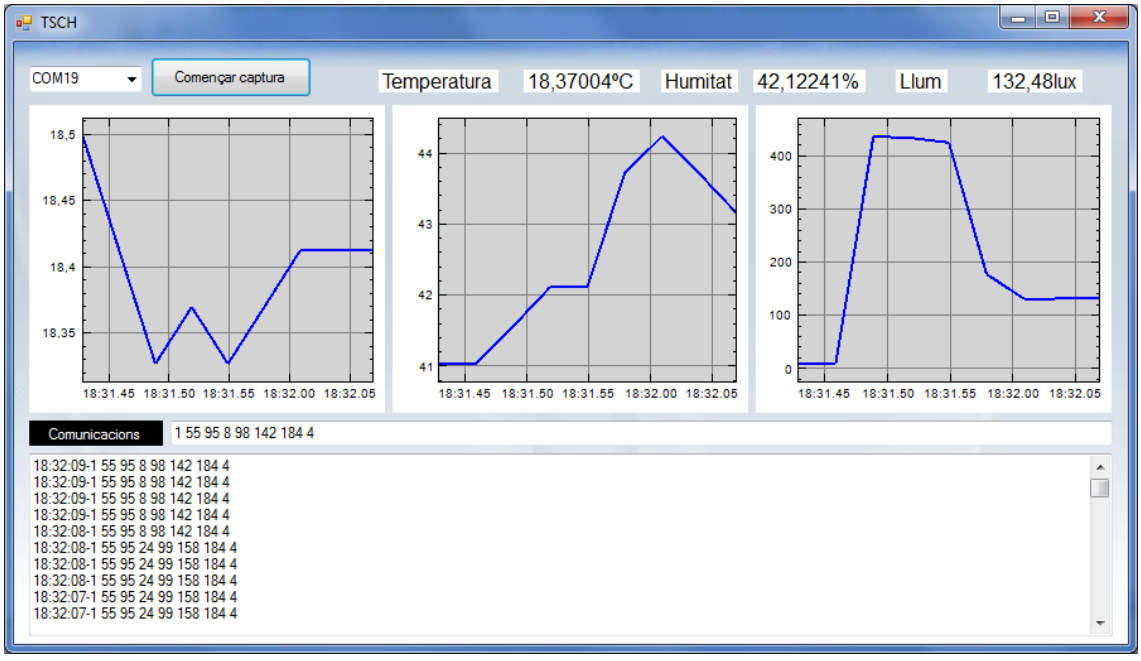
Per comprovar la correcta sincronització i comunicació entre dos OpenMote s'han desenvolupat aquesta prova, pensada en gran mesura per simular un entorn de funcionament estàndard del codi TSCH.

L'objectiu desitjat és muntar un OpenMote amb un OpenBattery que adquireixi les dades de temperatura, humitat i lluminositat. Aquesta remota tindrà carregat el codi TSCH i estarà configurat com esclau. El sistema tindrà un altre OpenMote amb el OpenBase amb el mateix codi TSCH però configurat com a coordinador, i enviarà les dades rebudes per comunicació sèrie, on un software rebrà, escalarà i visualitzarà les dades. Aquesta aplicació doncs, és un sistema estàndard de recolecció d'informació de sensors que utilitza un coordinador per centralitzar la informació i el protocol TSCH desenvolupat en les comunicacions radio.

El diagrama del sistema seria el següent:



Tant el codi de l'exemple com el codi de l'aplicació del pc estan als annexes i en les versions digitals entregades, i l'aspecte que té la interfície gràfica de l'aplicació pc és la següent:

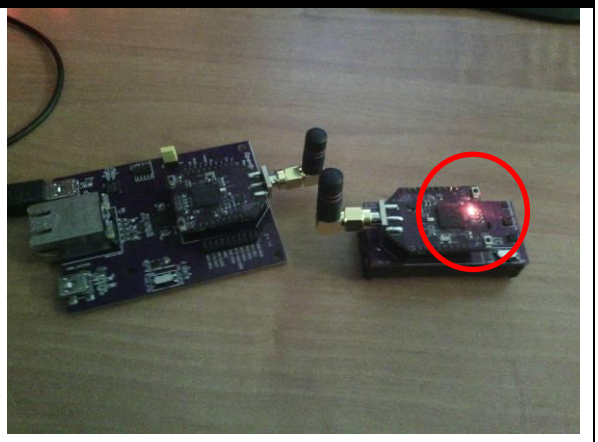
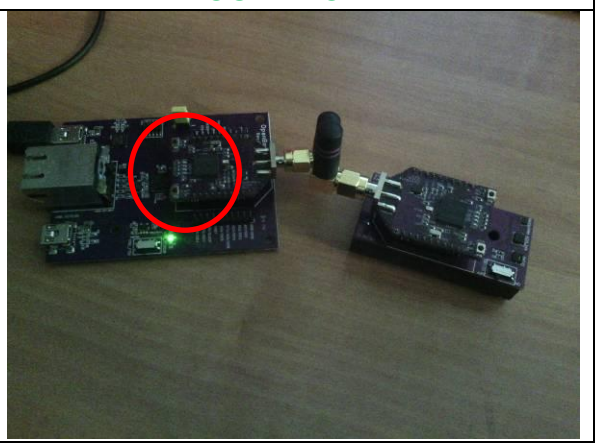


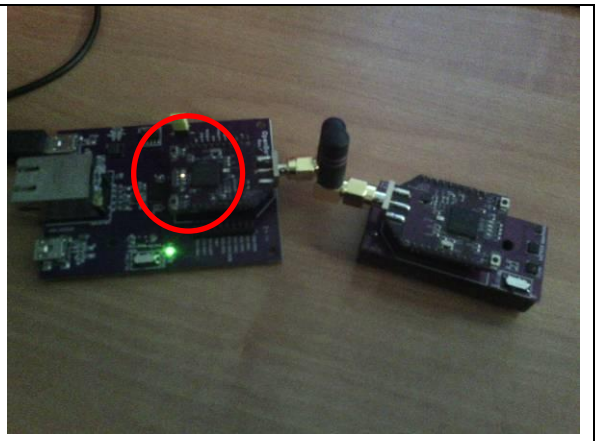
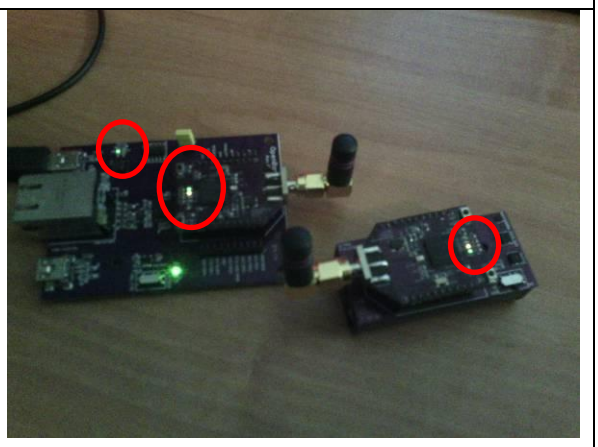
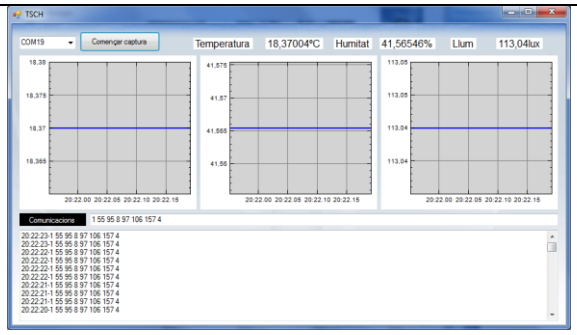
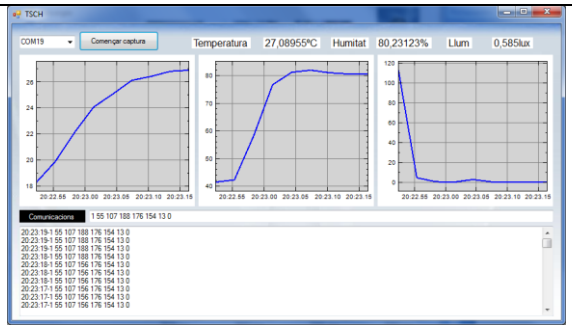
Aquest programa s'executa sobre Microsoft Net.Framework 2.0 o superior i s'ha programat amb l'entorn de desenvolupament SharpDevelop 3.4. Els dos es poden descarregar sense cap cost d'Internet.

Les dades de cada paràmetre s'envien com a 2 bytes separats, i segons queda determinat en els datasheets dels sensors SHT21 i Max44009, les operacions que cal fer per escalar els valors són les següents escrites en llenguatge Visual Basic:

tempe = bytes2(2)
tempe = (tempe << 8) + bytes2(3)
tempe2 = -46.86 + 175.72 * (tempe / (2 ^ 16)) 'Dada convertida
humi = bytes2(4)
humi = (humi << 8) + bytes2(5)
humi2 = -6 + 125 * (humi / (2 ^ 16)) 'Dada convertida
mantisa = bytes2(6)
exponent = bytes2(7)
llum = (2 ^ exponent) * mantisa * 0.045 'Dada convertida

Com l'objectiu es utilitzar aquesta aplicació per fer proves, es seguirà el següent procediment per verificar que funciona.

Acció	Resultat esperat	Resultat
Connectar l'esclau	Com el coordinador no està connectat, la mote no estarà sincronitzat i quedarà a l'espera de rebre l'ASN i la ranura. S'ha d'encendre el led vermell indicant aquest estat.	 CORRECTE
Connectar coordinador amb esclau desconnectat	El coordinador envia a la ranura 0 l'ASN a la xarxa pel canal 26. Això implica que s'encendrà el led taronja parpellejant.	

		 CORRECTE
Connectar esclau i coordinador	L'esclau rebrà l'ASN, demanarà ranura i començaran a comunicar. El led vermell s'apagarà i parpellejaran els leds verd(lectura) i taronja (escriptura) de les dues remotes. El led Rx de la OpenBase també parpellejarà al enviar les dades pel port sèrie.	 CORRECTE
Activar l'aplicació del pc	Han d'aparèixer dades molt estables al pc	 CORRECTE
Posar el dit a sobre del sensor de temperatura i del lúxmetre	La temperatura ha de pujar, l'humitat també es veurà afectada i el lúxmetre haurà de marcar un valor proper a 0.	 CORRECTE

El sistema funciona de forma estable, fiable, i tal com ha de funcionar, demostrant que l'aplicació exemple resulta totalment funcional.

Com a conclusió de la prova, es pot definir clarament que el sistema opera tal com s'esperava, complint els requisits marcats i demostrant el correcte funcionament tant del codi de comunicacions com el de l'exemple.

En els annexes, s'ha inclòs un video del funcionament d'aquesta prova.

c. Programa prova verificació retard

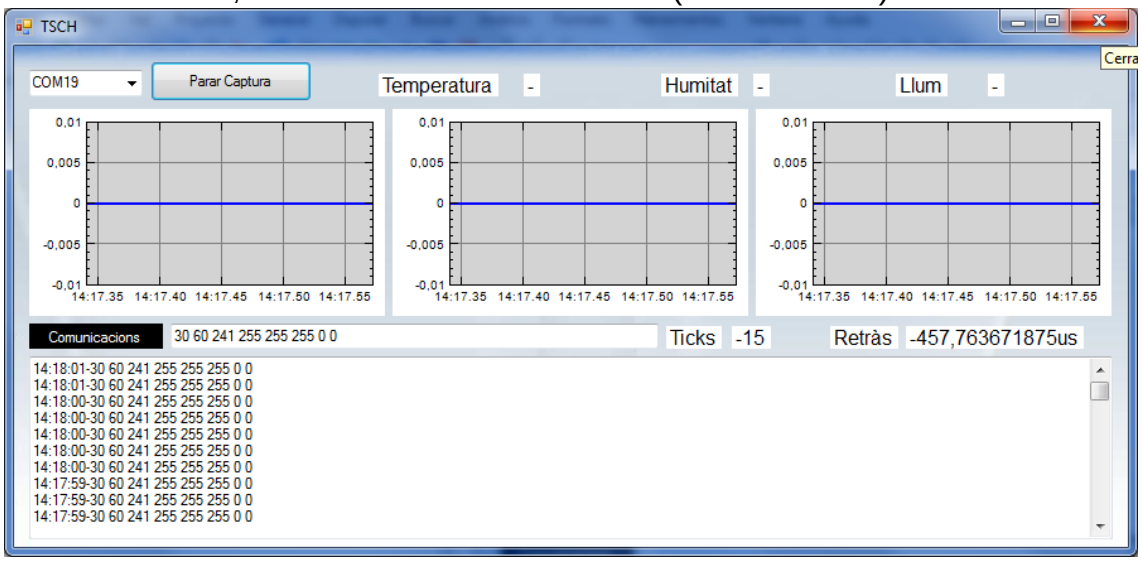
El sincronisme del sistema és vital, ja que com cada remota té que anar totalment sincronitzada amb el coordinador per començar tots les ranures en els instants que toquen, fet que requereix fer correccions de temps als esclaus. Les correccions es fan en dos processos, un en la ranura 0, on els esclaus utilitzen l'emissió de l'ASN per corregir el rellotge intern i l'altre a rebre l'ACK del coordinador.

Per poder veure com es comporten les comunicacions, s'ha modificat el programa perquè enviï pel port sèrie un trama amb el retard de comunicacions, en lloc de la informació rebuda i s'ha modificat el programa de l'apartat anterior per gestionar aquestes trames, indicant els ticks de retards i els microsegons que impliquen.

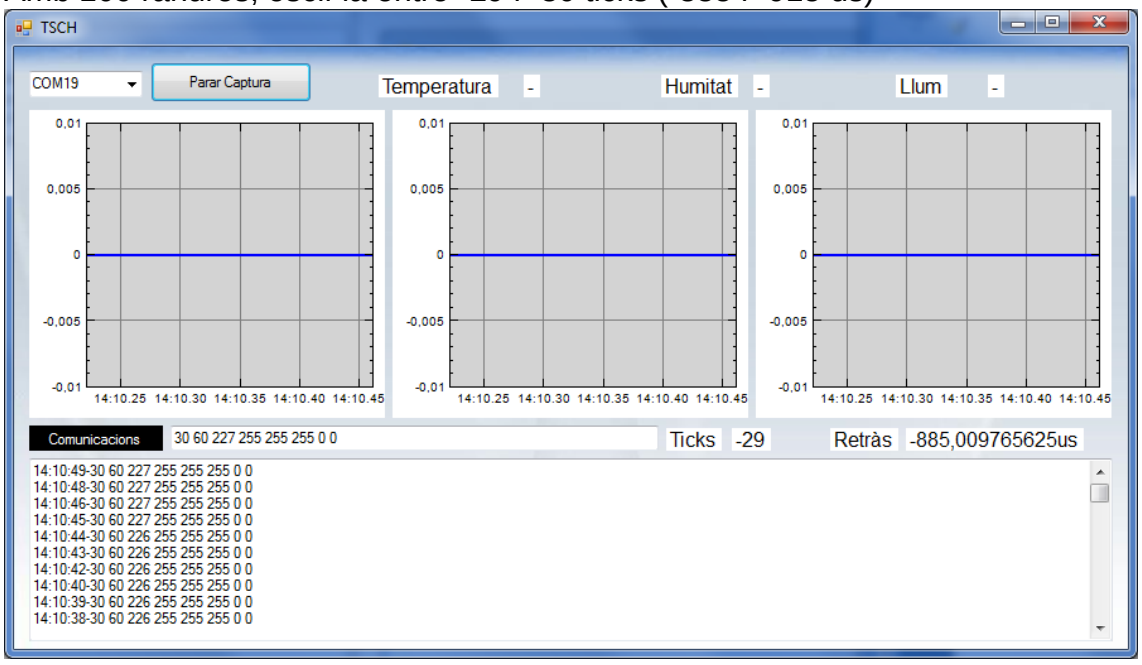
Els resultats han estat els següents:

Retard mesurat en la ranura 0, emissió de l'ASN
El retard mesurat dependrà del nombre de ranures, ja que quantes més ranures menys sincronitzacions i més error per tolerància es genera. Per tant farem una prova amb 20 ranures i una altra amb 100 ranures.

Amb 20 ranures, és manté entre -14 i -15 ticks (-427 i -457 us)

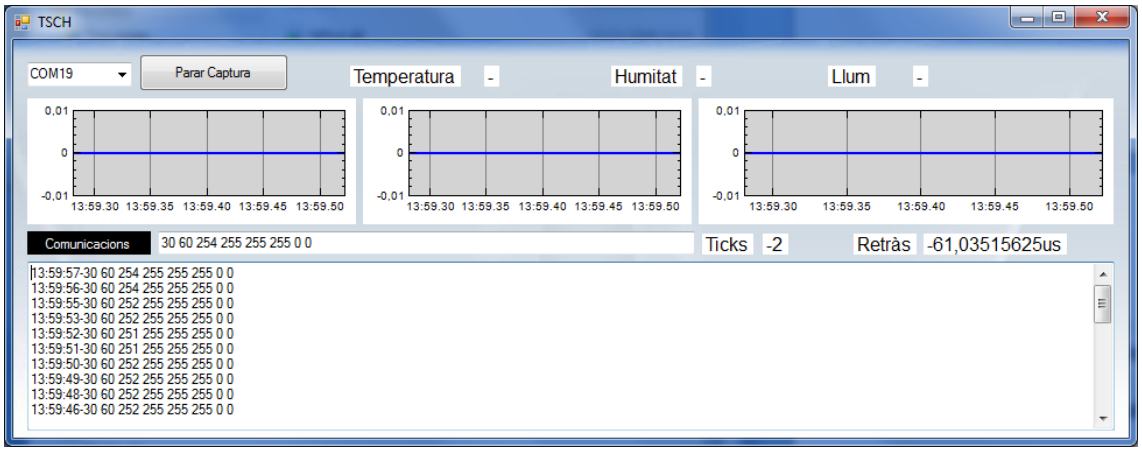


Amb 100 ranures, oscil·la entre -29 i -30 ticks (-885 i -915 us)



Tal com s'esperava al incrementar el nombre de ranures els errors augmenten, fet normal, ja que és disminueix el nombre de correccions que fan els esclaus.

Retard mesurat al rebre l'ACK



Aquesta prova dóna uns retards molt baixos perquè la correcció és fa a la ranura 0, i el ACK es rep a la ranura 2, de forma que dóna molt poc temps per acumular-se error de temps. El retard mesurat oscil·la entre -2 i -4 ticks, que signifiquen entre -61 i -122 microsegons, mantenint-se estable en aquests valors. Aquests retards no impliquen cap problema en les comunicacions i signifiquen que el codi realitza les correccions correctament.

Com a conclusió de la prova, es pot afirmar que el sistema mantè estables els retards i sempre dins les toleràncies de 3ms que té el sistema.

4. Conclusions

4.1. Lliçons apreses

El projecte ha aportat diverses lliçons que permetran desenvolupar altres projectes d'una forma més eficaç i més controlada.

A nivell tècnic el projecte a proporcionat a l'estudiant un major coneixement de la plataforma OpenMote i el microcontrolador Texas CC2538, de forma que en futurs projectes es podran executar els programes més ràpidament. Un dels majors problemes que s'han trobat durant el projecte, ha estat la gestió de les interrupcions del microprocesador, ja que si s'activava el sleepmode 2, les interrupcions de la radio no s'executaven. Per altra banda, també es van tenir problemes per accedir al buffer de lectura de la radio des de segons quines parts del codi. Aquests impediments van dificultar el desenvolupament i endarrerir una mica la planificació.

A nivell de gestió, el projecte s'ha desenvolupat en termes generals de forma ordenada i complint l'abast i planificació establerts. D'especial mención, es el fet de generar els documents d'inicialització, abast i de detall de sincronisme i processos de lectura/escriptura abans de començar a programar res, ja que va permetre establir clarament les funcionalitat i la forma d'executar-les des d'un inici, reduint en gran mesura les reprogramacions. S'ha demostrat la importància d'executar correctament les tasques en les fases d'inicialització i planificació del projecte, per aconseguir una bona execució.

4.2. Consecució de l'abast

L'objectiu del projecte era implementar un codi que permetés comunicacions TSCH sobre la plataforma OpenMote, i es considera que aquest objectiu s'ha complert, al igual que els diferents entregables marcats al projecte, executats en temps propers a la planificació establerta i amb bona qualitat.

Ara bé, un objectiu secundari no s'ha aconseguit, que era implantar les trames exactes de comunicació que marca la norma IEEE802.15.4e. Els motius de no haver-ho implementat han estat dos. La primera la dificultat d'interpretar la norma, on barreja diversos tipus de trames diferents i on no aporta exemples clars, i el segon, ha estat els problemes mencionats amb les interrupcions del microprocessador que va provocar un retràs en l'execució del codi.

4.3. Metodologia utilitzada

En termes generals s'ha considerat la metodologia utilitzada com a correcte, ja que ha definit clarament la feina i com executar-la. Es podria haver documentat millor la gestió de riscos, per tal de generar una

documentació més elaborada que permetés un traspàs en qualsevol moment a un altre cap de projecte. D'altra banda, aquest projecte no disposava d'un equip de persones, per tant no s'ha fet cap gestió de la comunicació, però amb un equip de treball, la gestió hagués requerit de procediments adequats per transmetre la informació i de reunions de seguiment, així com la definició d'un organigrama, personal i responsabilitats.

4.4. Projectes futurs

Com a continuació dels projecte es pot pensar en tota la gestió de capes superiors del nivell OSI, a part d'implementar les trames que marca la norma IEEE802.15.4e.

Es considera el sistema TSCH com molt robust i flexible, permetent comunicacions entre equips, i si s'implementessin algoritmes d'enrutació que permetin utilitzar una remota com a repetidora de les altres, es podrien muntar xarxes de gran abast i baix cost.

5. Glossari

ACK: Acknowledgement. Trama que envia el receptor a l'emissor per indicar que ha rebut el missatge.

ASN: Absolute Slot Number. Comptador de ranura de 5 bytes que s'incrementa en cada ranura. S'utilitza per definir el canal a utilitzar en cada ranura.

C: Llenguatge de programació

IoT: Internet on Things. Internet a les coses, concepte que implica afegir processadors i comunicacions a molts elements perquè interactuïn amb l'entorn.

Mote: Paraula en anglés que pot ser traduïda com a remota.

RTC: Real Time Clock. És un oscil·lador d'alta precisió que té una tolerància d'error molt petita. S'utilitza per fer mesures de temps precises.

Ticks: Flanc de pujada d'una senyal digital.

TSCH: Timeslotted Chanel Hopping. Temps ranurat amb salt de freqüència.

6. Bibliografia

[1]
Títol: Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)
Autor: Grup de treball IEEE 802.15
Any: 2012
Font d'informació:
http://standards.ieee.org/getieee802/download/802.15.4e-2012.pdf
Resum: Aquest document és la definició estàndard del protocol 802.15.4e, proporcionant un descripció general dels protocols, definint les trames i procediments que segueixen el LLDN, la DSME, la LL-Beacons i la trama en ranures TSCH, que són totes les definides per la norma 802.15.4e.
També defineix constants que afegir a les trames a enviar per intercanviar informació entre equips i establir correctament les comunicacions, així com els sistemes de seguretat a implementar
Aspectes rellevants: Aquest ha estat la principal font d'informació pel projecte, ja que és el document que detalla a nivell normatiu com ha de ser el TSCH i que ha de complir.

[2]
Títol: Web OpenMote
Autor: OpenMote Technologies
Any: -
Font d'informació:
http://www.openmote.com/
Resum: Aquesta web és la del fabricant de la plataforma hardware que utilitzarà el projecte, disposa d'informació hardware i del software compatible, així com informació dels protocols de comunicació que comunicació que permet implementar.
Aspectes rellevants: Ha estat la principal font d'informació a nivell hardware pel projecte, ja que proporciona enllaços als datasheet dels diversos components i conèixer així les seves característiques i prestacions.

[3]
Títol: Web OpenWSN
Autor: OpenMote Technologies
Any: -
Font d'informació:
https://openwsn.atlassian.net/wiki/display/OW/Implementing+IEEE802.15.4e
Resum: Aquesta web forma part del sistema operatiu OpenWSN, que ha implementat el protocol TSCH dintre de les seves funcionalitats. En un tutorial detalla la seva implementació, juntament amb explicacions i consideracions del seu funcionament.
A la vegada es disposa del codi font desenvolupat en C de tot el sistema operatiu, dintre del qual es troben les llibreries que implementen el TSCH.
Aspectes rellevants: La informació detallada de com s'ha implementat el TSCH ha estat de gran utilitat per executar el projecte, ja que determina una

forma d'organitzar la feina a executar i les consideracions pràctiques que s'han de tenir.

[4]
Títol: A decentralized scheduling algorithm for time synchronized channel hopping
Autors: Andrew Tinka, Thomas Watteyne, Kristofer S. J. Pister, Alexandre M. Bayen
Any: 2010
Font d'informació:
http://bayen.eecs.berkeley.edu/sites/default/files/journals/mca11_v2.pdf
Resum: Aquest article explica com es va muntar una xarxa basada en TSCH on diversos sensors es sincronitzaven i enviaven informació entre ells, incloent algoritmes de planificació de comunicacions.
Aspectes rellevants: L'article ha servit per tenir una aplicació pràctica dels sistemes que utilitzaran les comunicacions TSCH, important per desenvolupar l'aplicació exemple.

[5]
Títol: Time Slotted, Channel Hopping MAC
Autors: Kris Pister, Chol Su Kang, Kuor Hsin Chang, Rick Enns, Clint Powell, José A. Gutierrez, Ludwig Winkel
Any: 2008
Font d'informació:
https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&ved=0CFIQFjAG&url=https%3A%2F%2Fmentor.ieee.org%2F802.15%2Fdcn%2F08%2F15-08-0581-02-004e-time-slotted-channel-hopping-mac.ppt&ei=GSw0VK6HBsSWaufagPgM&usg=AFQjCNH1PYMb94Jf4AfhtpXFARinGPlUTw&cad=rja
Resum: Aquesta presentació explica de forma detallada i visual els procediments a seguir en l'enviament i recepció de les trames, així com el sistema de ranures i el canvi de freqüència.
Aspectes rellevants: La presentació molt útil per assentar una bona idea del sistema a implementar.

7. Annexos

Llistat d'annexes inclosos:

- Acta de constitució de projecte
- Definició abast PAC 1
- Codi TSCH
- Codi OpenMote aplicació proves capítol 2.5, enviament dades sensors
- Codi aplicació pc recol·lecció dades sensors
- Vídeo aplicació enviament dades sensors funcionant