

# IMPLEMENTACIÓ PROTOCOL TSCH SOBRE LA PLATAFORMA OPENMOTE

Màster en enginyeria de telecomunicacions

Presentació treball final de master

Autor: Albert Creixell

Tutor: Pere Tuset



**Universitat Oberta  
de Catalunya**

**INTRODUCCIÓ INTERNET A LES COSES**

**OBJECTIU DEL PROJECTE**

**IEEE 802.15.4E - TSCH**

**OPENMOTE**

**IMPLEMENTACIÓ IEEE 802.15.4E TSCH**

**APLICACIONS EXEMPLE I PROVES**

**CONCLUSIONS I DUBTES**

# UOC INTRODUCCIÓ INTERNET A LES COSES

Internet a les coses, és un concepte que es refereix a la interconnexió d'objectes a Internet.

Se sol abreujar com IoT (Internet on Things).

- Molts elements són susceptibles d'estar connectats.

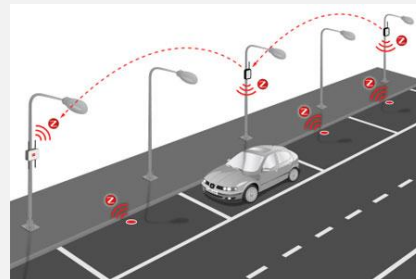


# UOC INTRODUCCIÓ INTERNET A LES COSES

La interconnexió de gran quantitat d'aparells i sensors permet disposar d'una gran quantitat de fonts d'informació, multitud de paràmetres, i establir sistemes de presa de decisió, que poden ser manuals, automàtiques o mixtes.

Exemples:

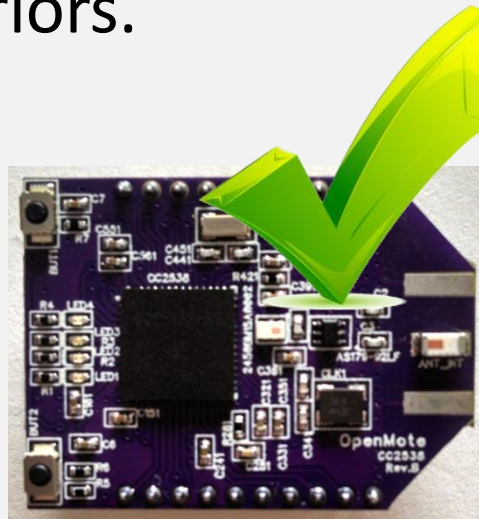
- Llegir la dada de temperatura d'una vivenda i activar la calefacció o aire condicionat
- Automatitzar el control de presència i dels serveis de un edifici
- Adquirir l'estat dels aparcaments disponibles, distribuir-los, i reduir la quantitat de vehicles que busquen aparcament



És important que el hardware que s'utilitzi per IoT compleixi:

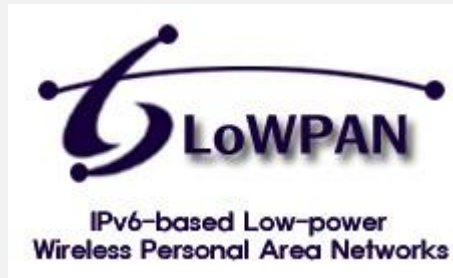
- ✓ Baix consum
- ✓ Comunicacions sense fils
- ✓ Baix cost.

El hardware OpenMote utilitzat en aquest projecte compleix els tres requisits anteriors.



Junt amb aquest tipus de hardware, s'han desenvolupat protocols de comunicacions i firmware molt pensats per IoT.

Aquests protocols estan pensats per donar resposta a les necessitats específiques d'aquestes xarxes. Alguns d'ells són el Zigbee, IEEE802.15.4, 6LoWPAN, IEEE RPL, IEEE CoAP, Wireless HART



Sistemes operatius com Contiki, FreeRTOS, OpenWSN, TinyOS o RiOT, han estat desenvolupats per utilitzar-se de suport per generar aplicacions IoT

Aporten els següents avantatges:

- ✓ Gestió de memòria optimitzada
- ✓ Comunicacions IP, TCP,UDP, HTTP i 6lowpan, RPL o CoAP.
- ✓ Implementa sistemes d'estalvi d'energia, per augmentar autonomies
- ✓ Càrrega de mòduls dinàmica quan s'està executant el programa
- ✓ Programació amb llenguatges d'alt nivell (C++) i llibreries multi-threaded

L'objectiu principal del projecte és:

- **Desenvolupar una llibreria que implementi el protocol TSCH, definit al IEEE 802.15.4e.**

Aquest objectiu s'ha d'aconseguir complint ho següent:

- ✓ La plataforma hardware serà OpenMote
- ✓ El programa serà en llenguatge C
- ✓ No s'utilitzarà cap sistema operatiu encastrat.



Publicat a la norma IEEE 802.15.4e, el mode TSCH (Timeslotted Channel Hopping), està pensat per comunicacions radio, aporta ho següent:

### Aplicacions objectiu

- Monitorització de sensors i procés
- Control no crític
- Diagnòs i manteniment predictiu

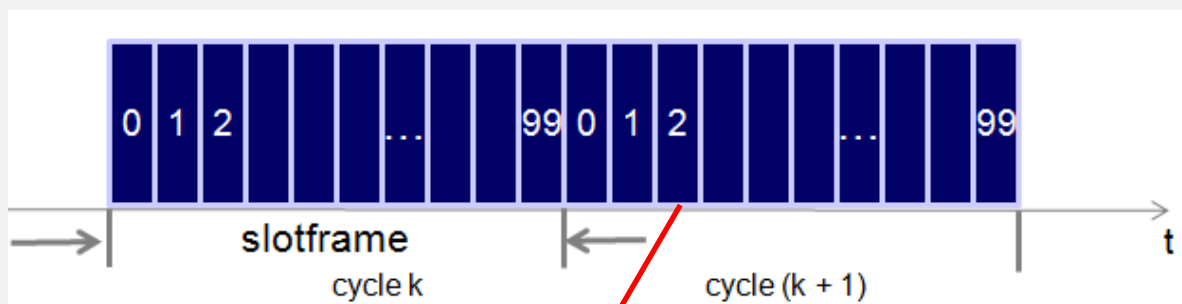
### Avantatges

- Comunicacions robustes, multicamí i multifreqüència
- Baix consum
- Determinístic
- Escalable

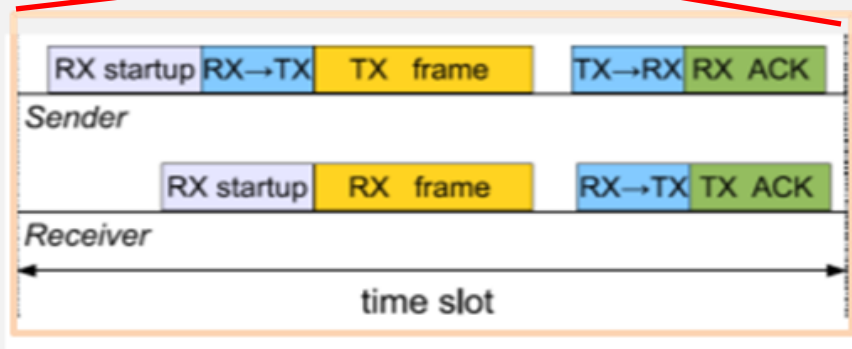
Aquest protocol es basa en dos característiques:

- Timeslotted (TS), que proporciona:
  - ✓ Determinisme
  - ✓ Estalvi energètic
- Channel Hopping (CH), que proporciona
  - ✓ Fiabilitat front interferències
  - ✓ Comunicacions simultànies mateixa ranura

El TS treballa amb trames formades per ranures.  
Cada ranura està formada per tres fases, preparació, enviament o recepció i confirmació de recepció.



2



## Consideracions sobre el TimeSlotted

- Totes les remotes requereixen sincronització de temps
- Un coordinador marca el temps de la xarxa
- Les remotes esclaus tindran sistemes per corregir el seu rellotge intern i estar sempre sincronitzades
- Totes les ranures estaran contingudes en la mateixa trama de ranures que s'anirà repetint
- Totes les comunicacions entre remota i remota tindran confirmació de recepció ACK

## Operacions a fer amb el TimeSlotted

- Totes les remotes mantindran sincronitzat el temps i l'ASN
- A l'inici de cada ranura cada remota determina si ha d'escriure, llegir o no fer res.
- Si ha de llegir o escriure, es prepararà i activarà la radio.
- Un cop llegit o escrit, gestionarà l'ACK
- Finalment, quan hagi acabat desconnectarà la radio per estalviar energia

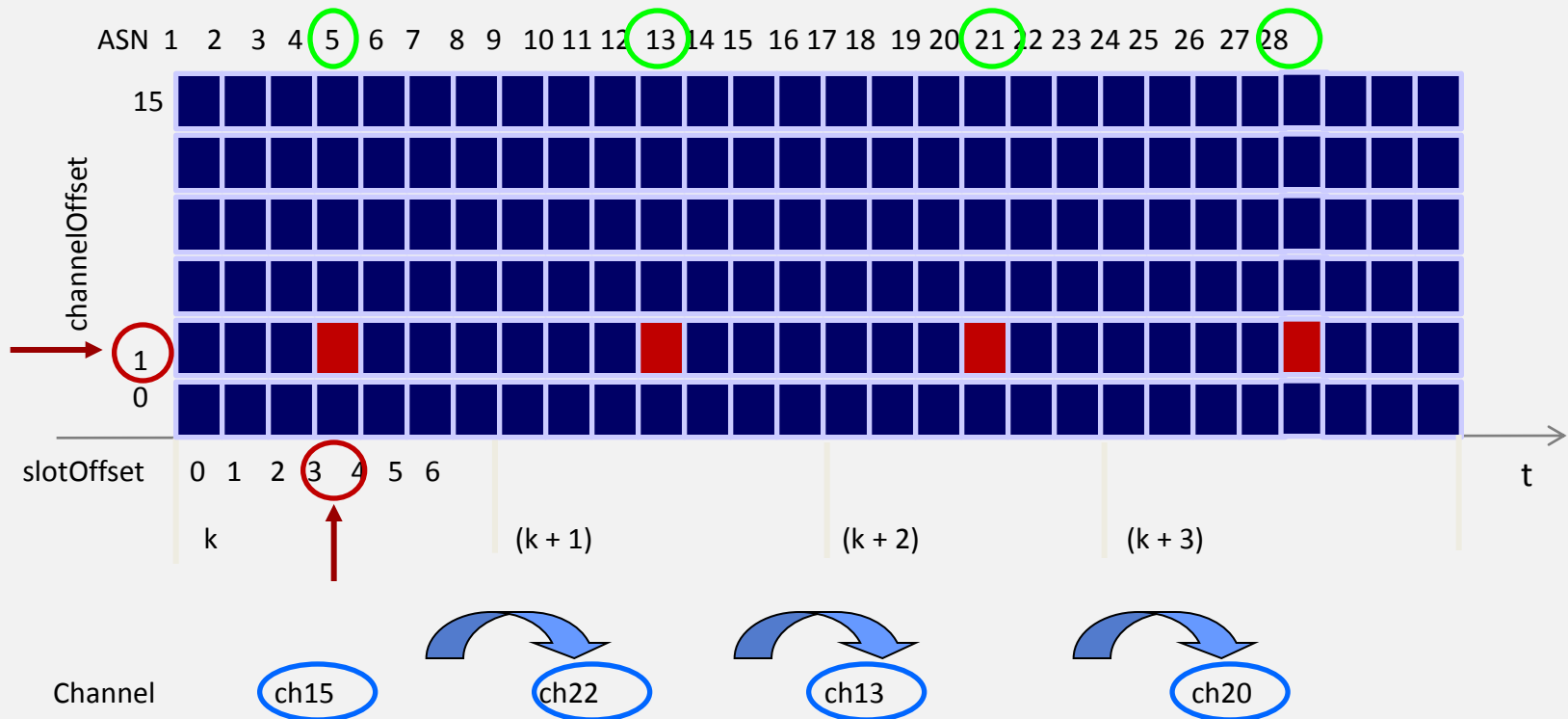
El CH, salt de freqüència, és basa en utilitzar un canal diferent en cada ranura, segons un comptador compartit entre tota la xarxa, el ASN, i un paràmetre anomenat channel offset.

Un coordinador proporcionarà l'ASN als nous membres de la xarxa per tal que es puguin sincronitzar.

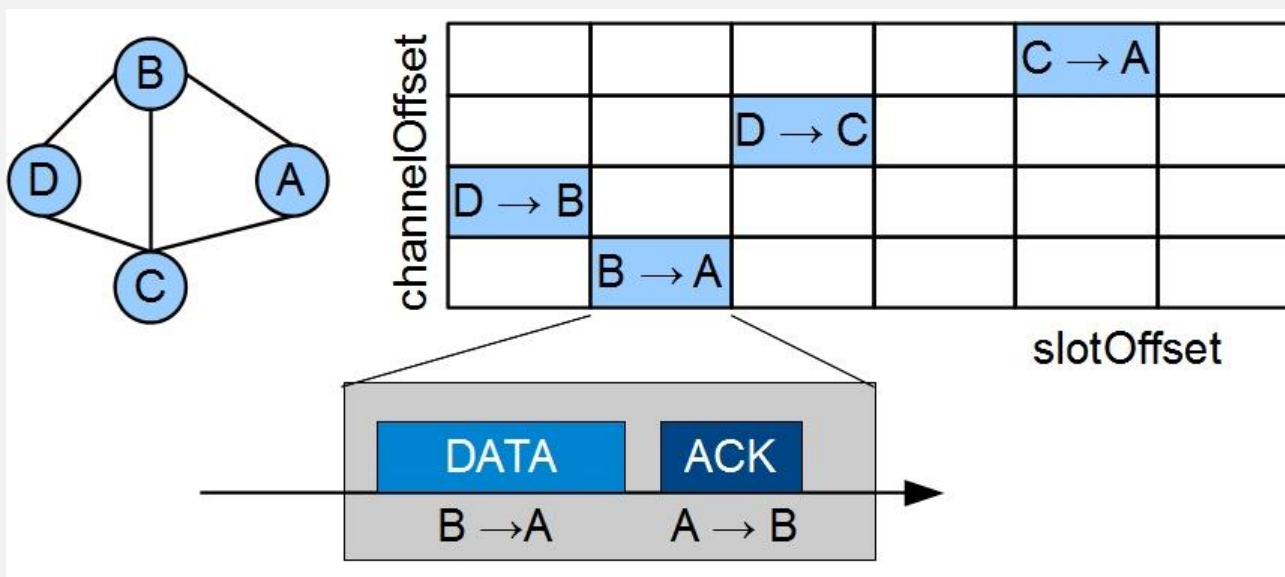
$$f = F \{ (ASN + chOf) \mod n_{ch} \}$$

Table I. Frequency Translation

$k$	ASN	chOf	$f$
0	4	1	5
1	11	1	12
2	18	1	3
3	25	1	10



El TSCH, uneix les seves característiques, per comunicar segons una planificació establerta, que defineix les comunicacions entre remotes. El channel offset permet que varies remotes comuniquin en la mateix ranura sense interferir-se





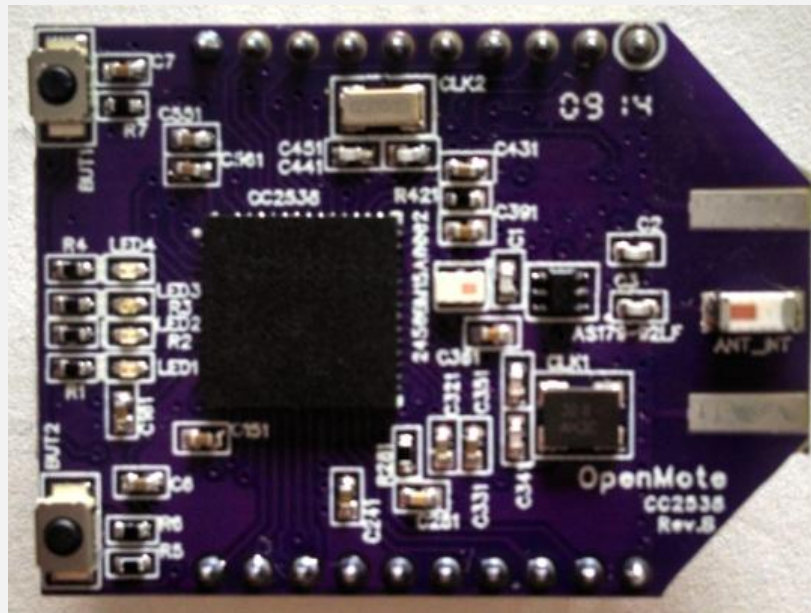
El projecte es desenvoluparà sobre la plataforma hardware OpenMote, que distribueix la companyia OpenMote Technologies.

La gama de productes és basa en els 3 hardwares específics següents:

- OpenMote-CC2538
- OpenBase
- OpenBattery

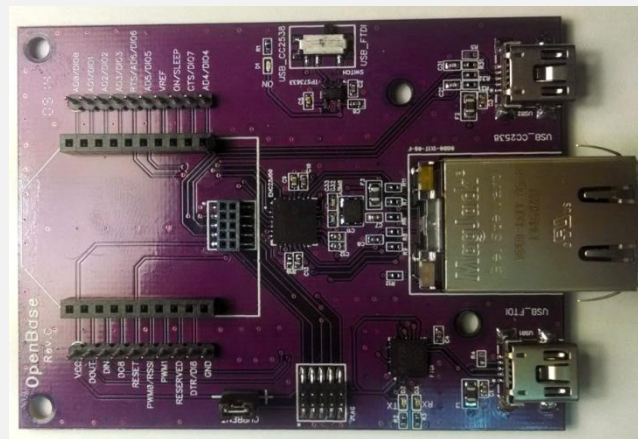
# OpenMote-CC2538

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.



## OpenBase

Placa pensada per connectar un OpenMote-CC2520, que incrementa sistemes d'alimentació, depuració i de comunicacions.



## OpenBattery

Placa pensada per connectar un OpenMote-CC2520, pensada per alimentar amb piles una unitat. A més disposa d'alguns sensors per adquirir dades. Aquest hardware està pensat sobretot per les unitats esclau de la xarxa.



## Firmware

Per programar l'aplicació s'ha utilitzat com a base el firmware que el fabricant posa a disposició a

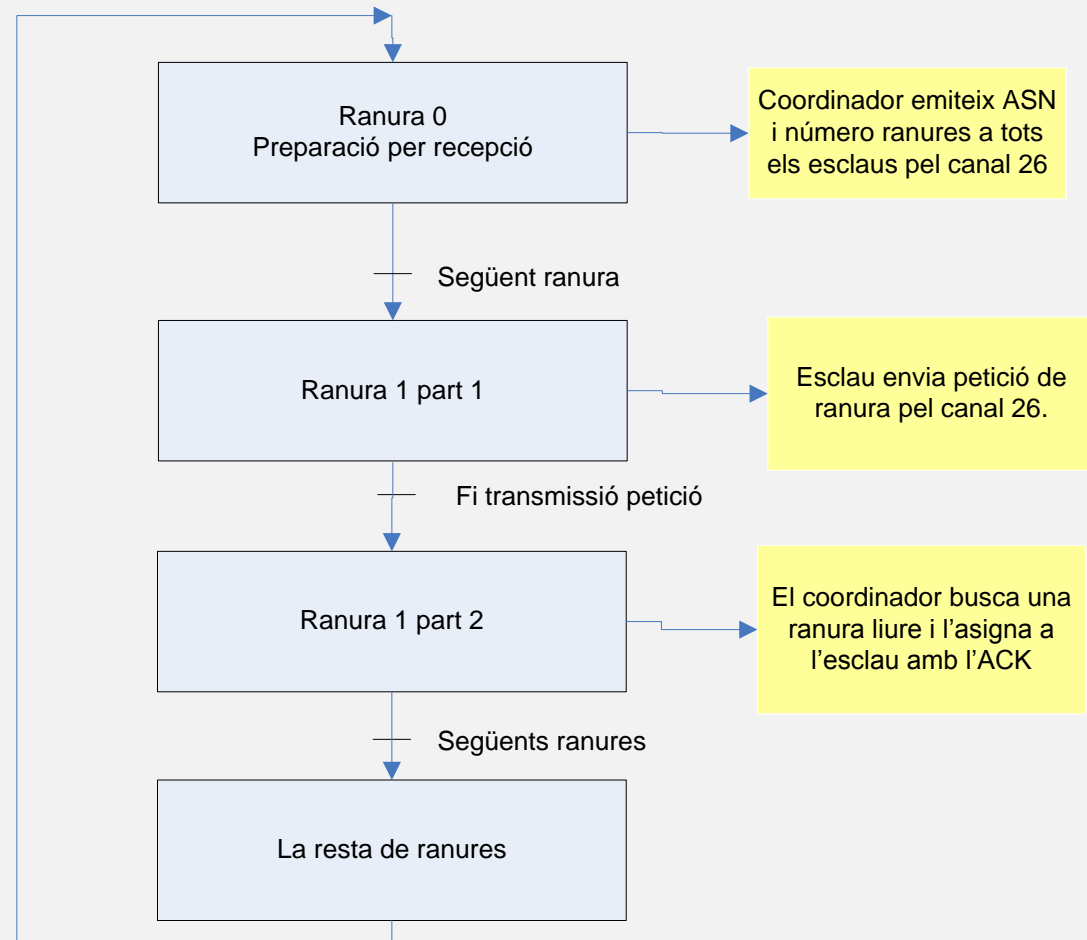
<https://github.com/OpenMote/firmware>

El firmware proporciona llibreries i kernels per direccionar el hardware més fàcilment, evitant haver de programar a molt baix nivell.

## Procés de sincronització

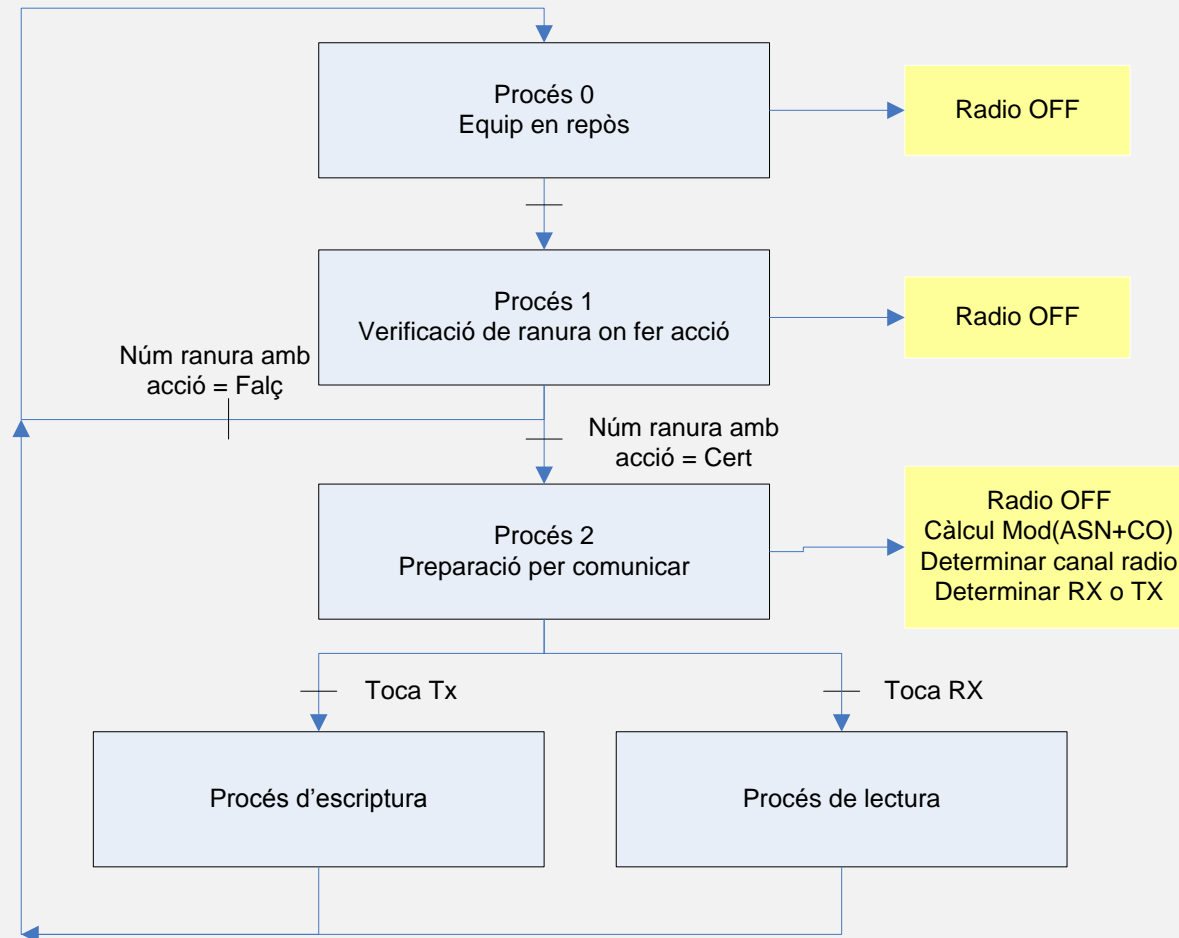
La comunicació s'inicia sincronitzant els esclaus amb el coordinador, sincronitzant l'ASN i el rellotge, i posteriorment assignant una ranura lliure.

Seguint el següent diagrama.



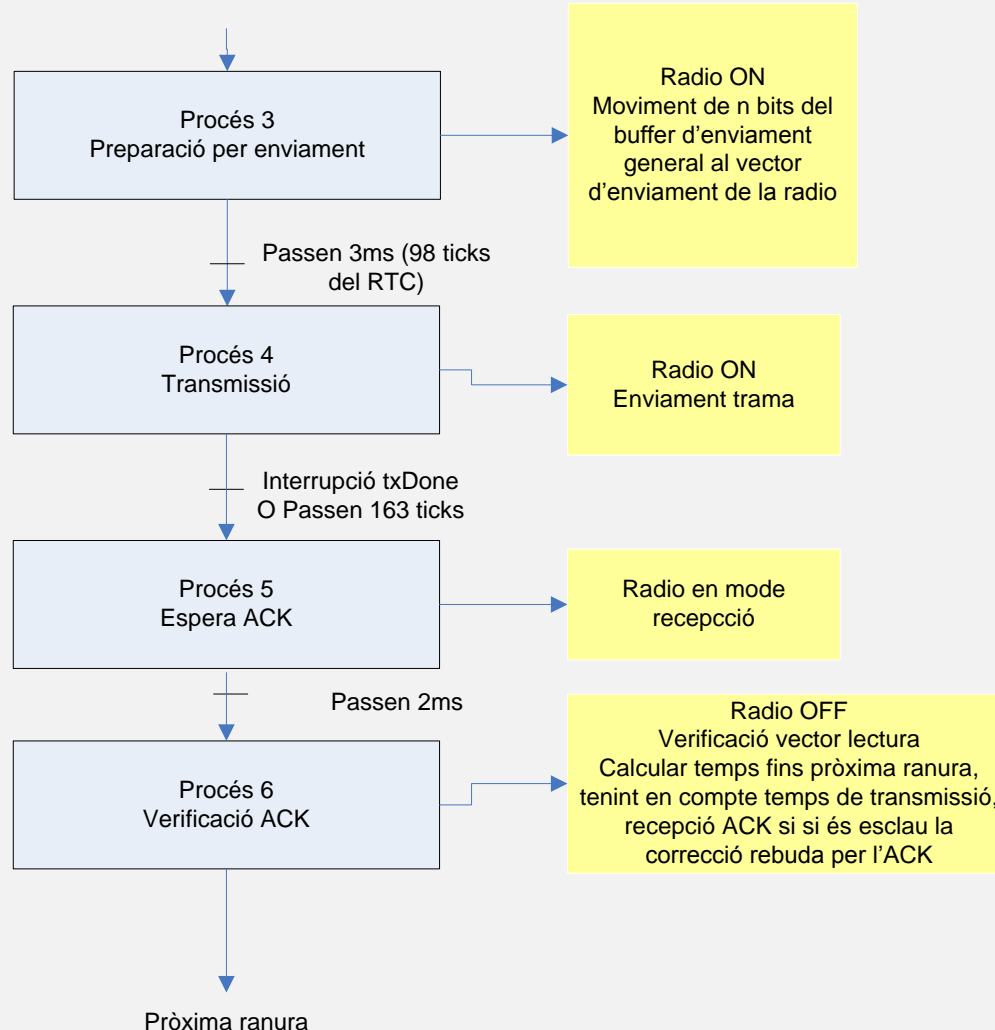
## Processos executats en una ranura

En cada ranura, s'executen diversos processos. Primer de tot, es determina si la ranura és de lectura, escriptura o passiva.



## Procés escriptura

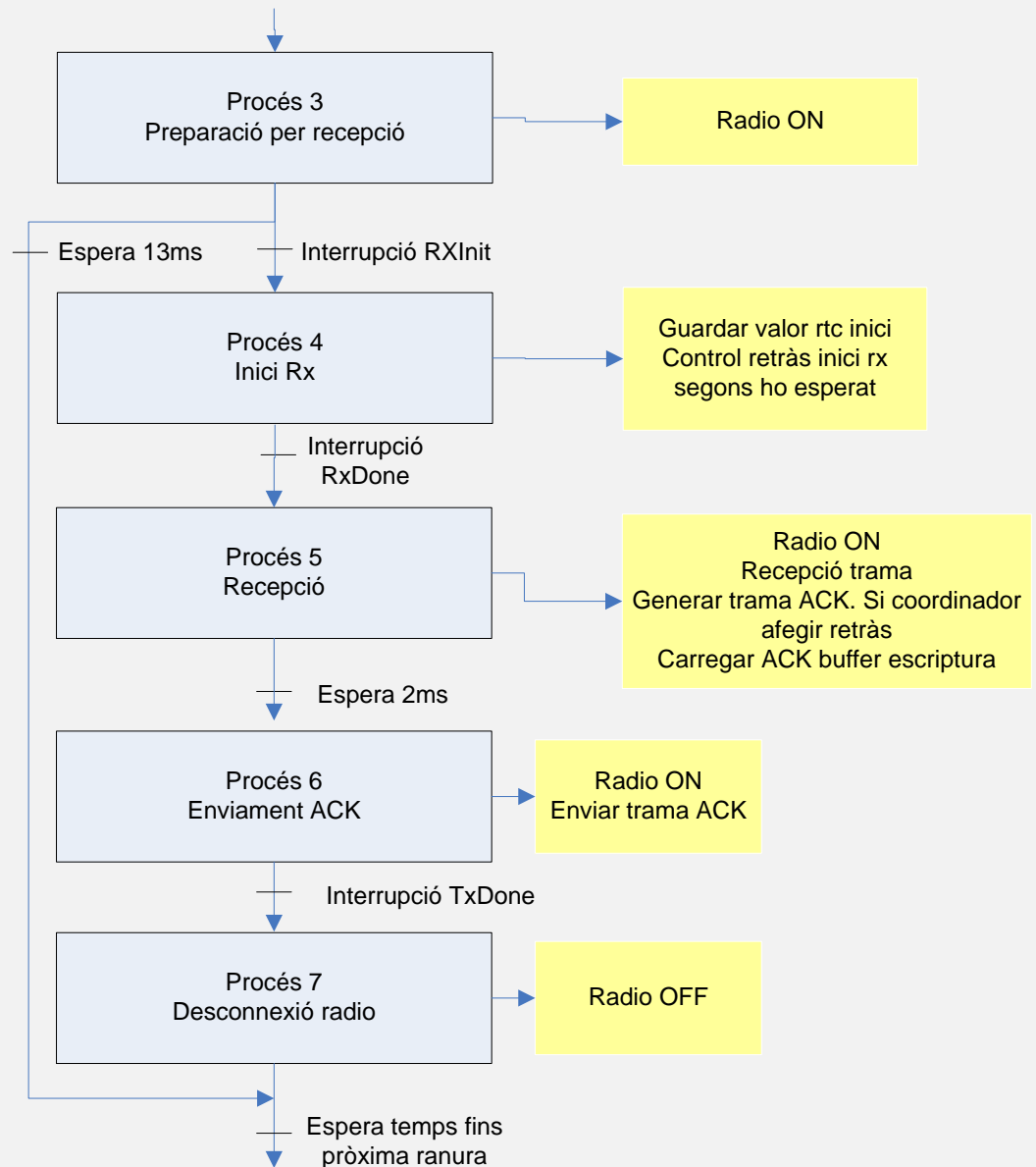
Si la ranura és d'escriptura, es seguirà la següent seqüència





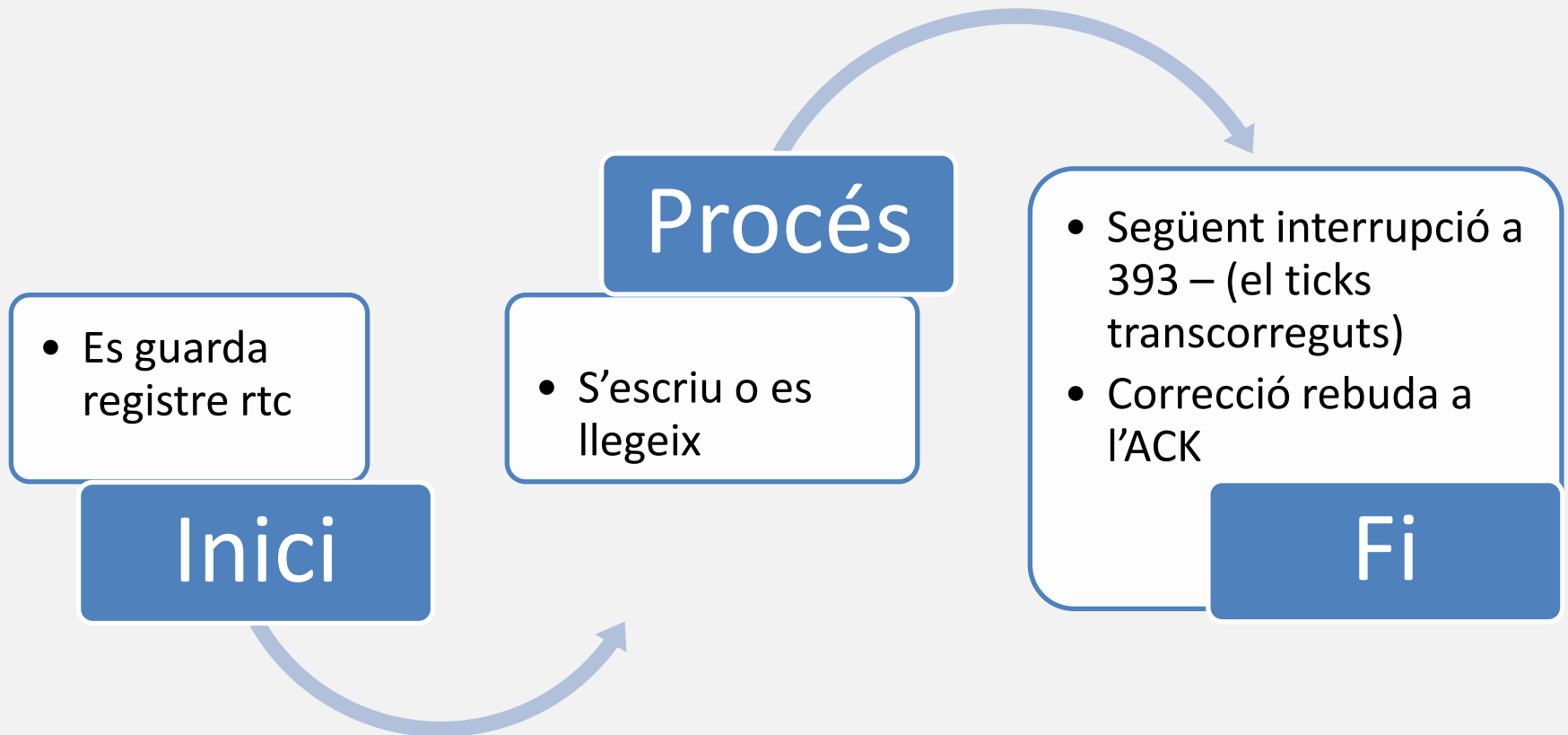
## Procés lectura

Si la ranura és de lectura, es seguirà la següent seqüència



## La clau, la gestió del RTC per controlar la sincronització

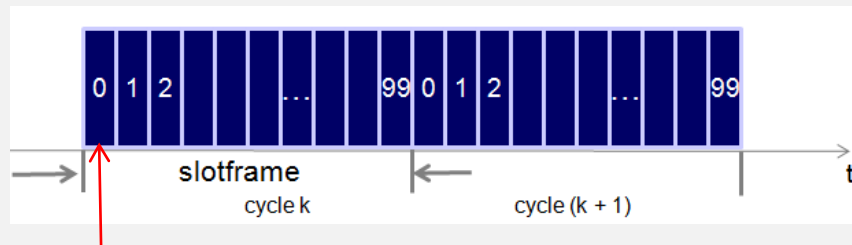
```
tic6=rtc.read();  
int32_t total;  
tic1=393-(tic6-tic5); //Resta als 12ms el temps transcorregut  
total=tic1-correccio; //Aplica la correcció marcada pel coordinador  
rtc.start(total);
```



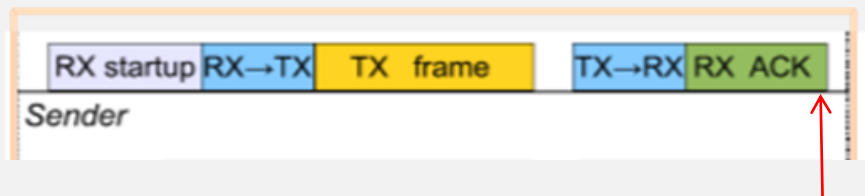
## Correccions del RTC

Els esclaus apliquen correcció de temps en **dos moments per trama**.

- En la ranura 0, al emetre l'ASN el coordinador. Ja que l'inici de recepció a la ranura 0, ha de ser 3ms després de començar, i això permet corregir el rellotge intern.



- Al rebre la correcció en el ACK



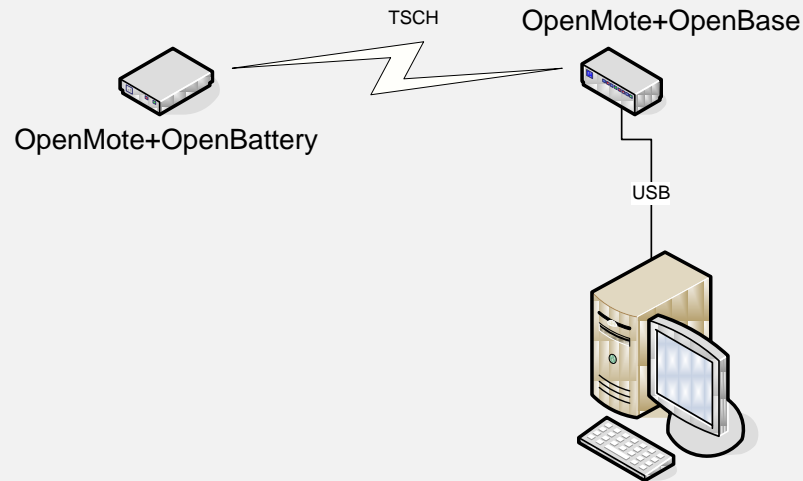
## Verificació de les comunicacions

Per verificar el funcionament del sistema, s'han dissenyat dos aplicacions, una destinada a emular una aplicació estàndard on es pot utilitzar el TSCH, i l'altre molt més de control de funcionament.

## Aplicació d'obtenció de dades de sensors

Es desitja una aplicació on s'adquireixin dades de temperatura, humitat i llum des d'una remota aïllada. Aquesta informació s'enviarà per TSCH al coordinador, i aquest les enviarà per port USB a una aplicació de pc.

L'esquema del sistema desitjat seria el següent:



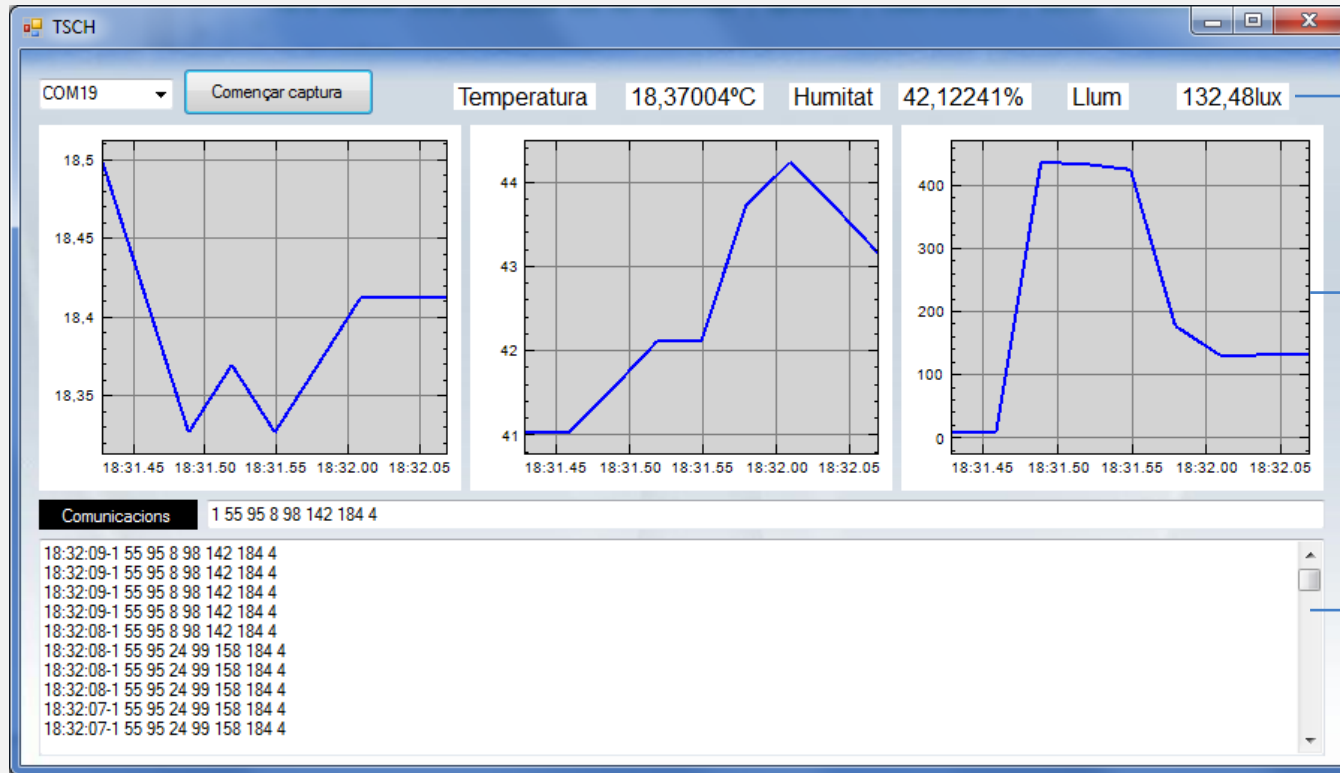
## **Aplicació d'obtenció de dades de sensors**

Objectius d'aquesta aplicació:

- ✓ Demostrar l'aplicació i funcionament del sistema
- ✓ Verificar el comportament del codi en tots els processos:
  - Procés de sincronització
  - Procés d'assignació de ranures
  - Procés de comunicació de dades

## Aplicació d'obtenció de dades de sensors

Aquesta aplicació utilitzarà un programa de pc programat per aquesta funció amb .Net framework 2.0



Últim valor llegit

Històric últims valors

Trames rebudes RS232

## Aplicació d'obtenció de dades de sensors

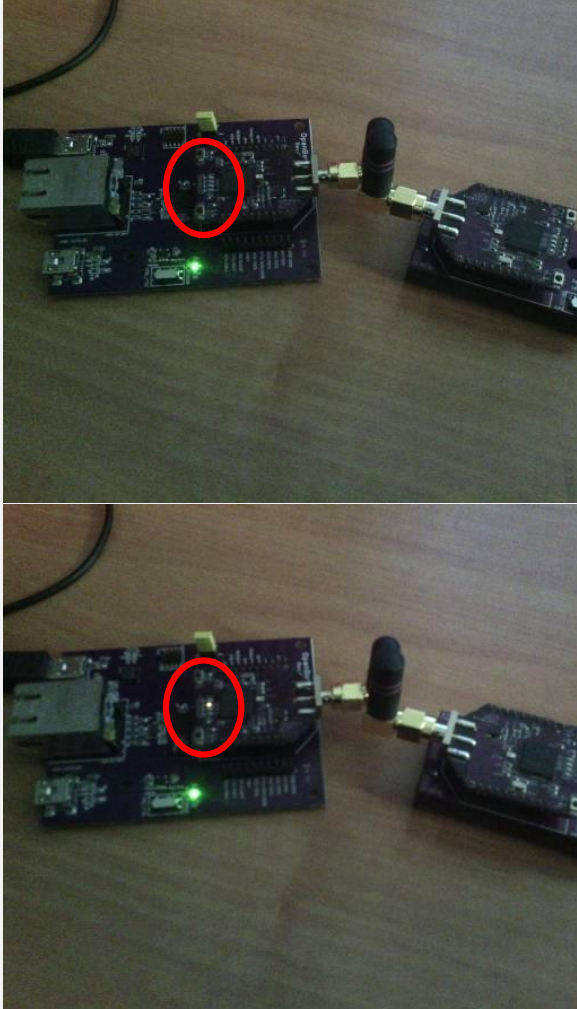
Per verificar el funcionament s'ha redactat un procediment amb un resultat esperat que s'ha verificat.

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.	 <p>CORRECTE</p>



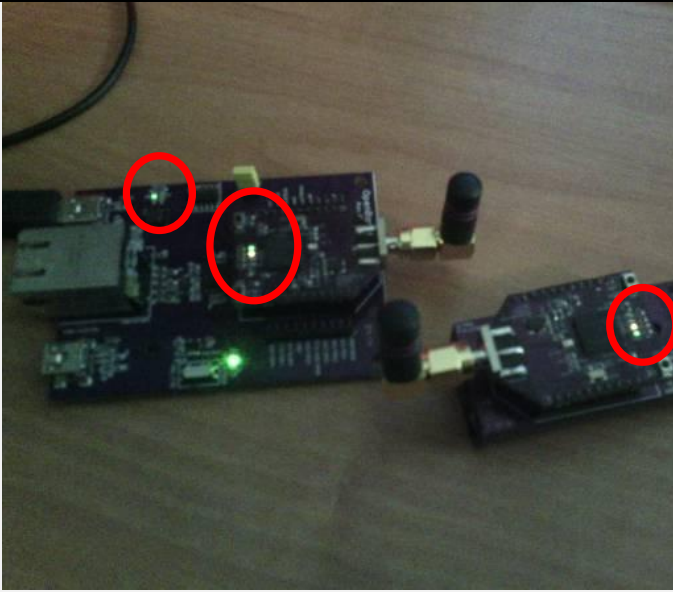


## Aplicació d'obtenció de dades de sensors

Acció	Resultat esperat	Resultat
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.	 <p>CORRECTE</p>

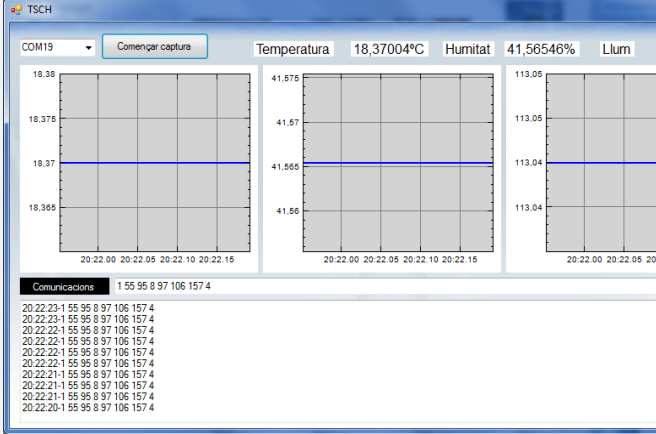
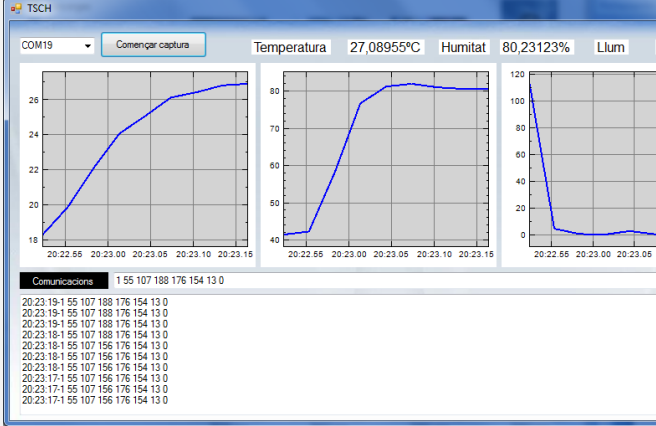


## Aplicació d'obtenció de dades de sensors

Acció	Resultat esperat	Resultat
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.	 <p>CORRECTE</p>



## Aplicació d'obtenció de dades de sensors

Acció	Resultat esperat	Resultat
Activar l'aplicació del pc	Han d'aparèixer dades molt estables al pc	 <p>CORRECTE</p>
Posar el dit a sobre del sensor de temperatura i del luxmetre	La temperatura ha de pujar, l'humitat també es veurà afectada i el luxmetre haurà de marcar un valor proper a 0.	 <p>CORRECTE</p>



## Aplicació d'obtenció de dades de sensors

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.

## **Aplicació de control dels retards de comunicacions**

El sincronisme del sistema és vital, i es requereix fer correccions de temps als esclaus.

S'ha modificat el programa perquè envii pel port sèrie una trama amb el retard de comunicacions, i així poder-lo monitoritzar

## Aplicació de control dels retards de comunicacions

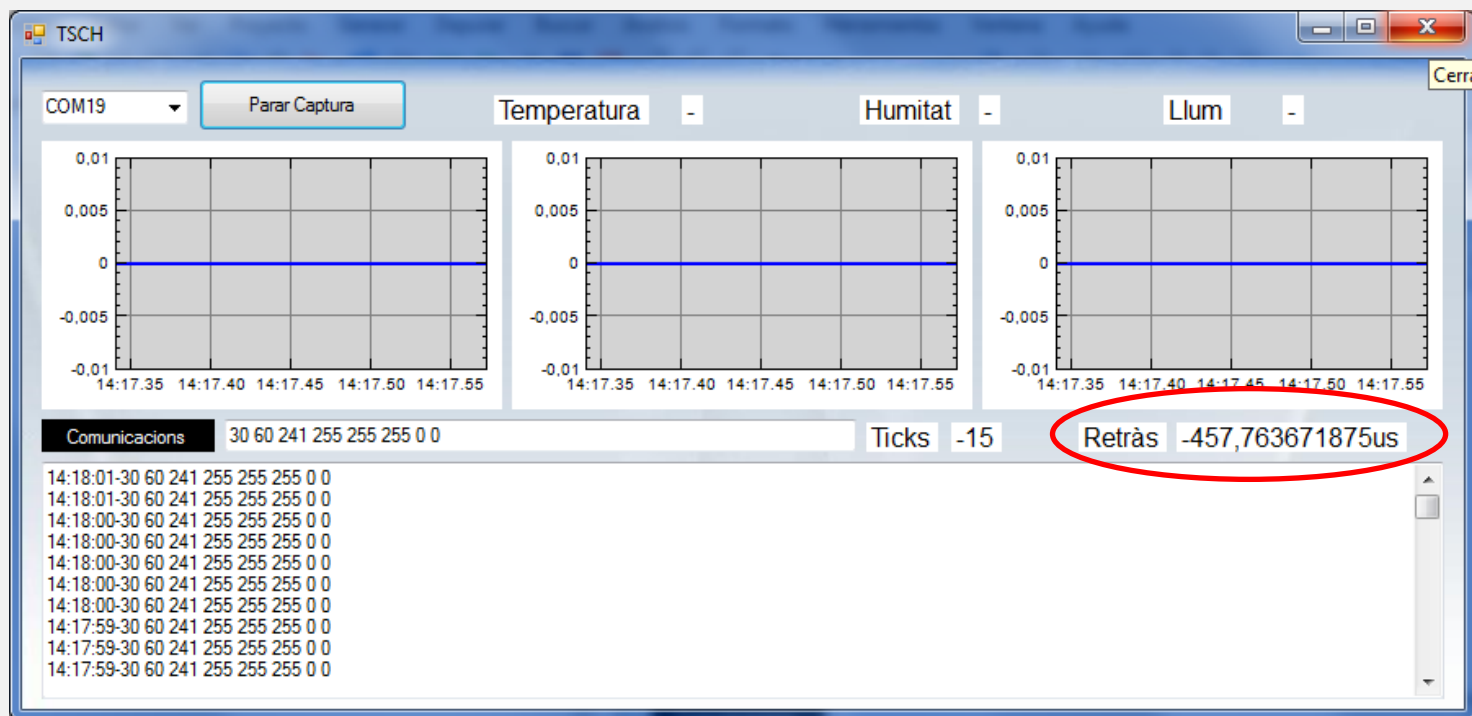
Les correccions es fan en dos processos, un en la ranura 0, on els esclaus utilitzen l'emissió de l'ASN per corregir el rellotge intern i l'altre a rebre l'ACK del coordinador.

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.

## Aplicació de control dels retards de comunicacions

### Retard mesurat en la ranura 0, emissió de l'ASN

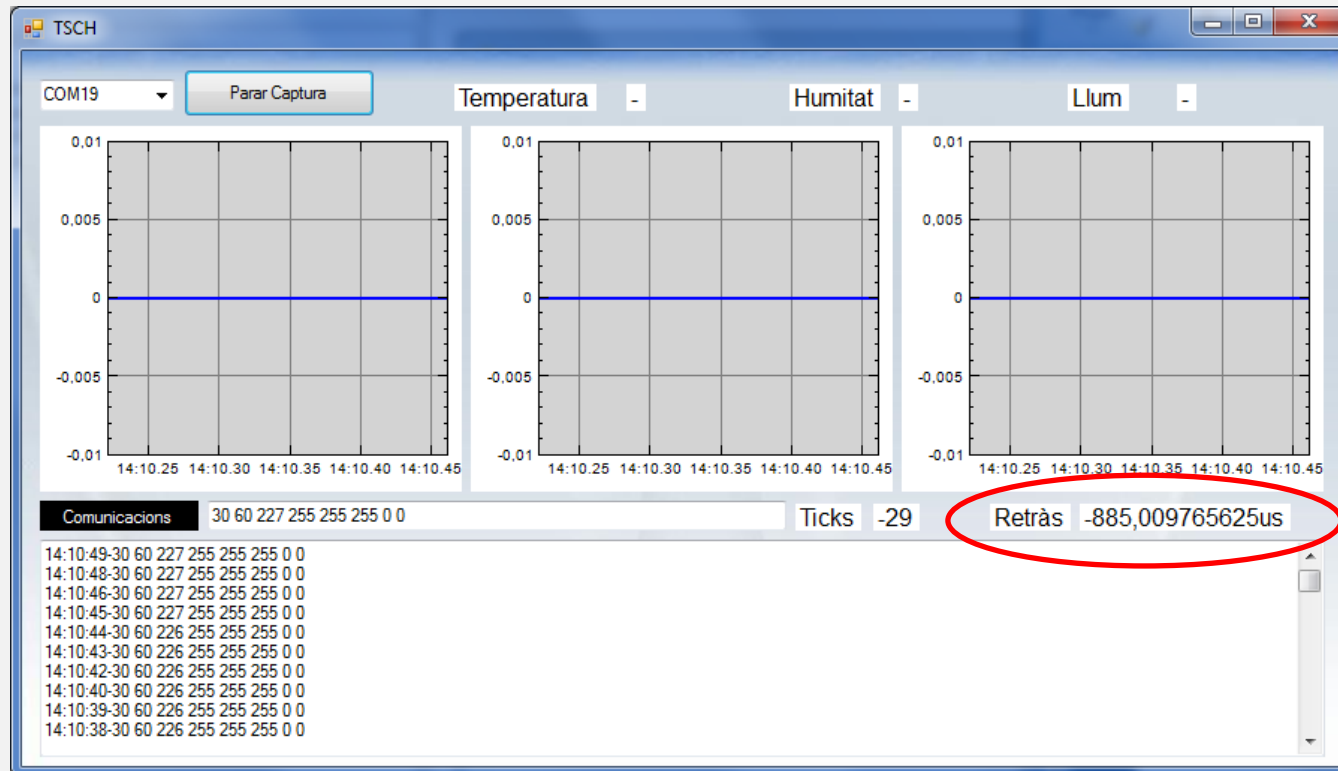
Amb 20 ranures, entre -14 i -15 ticks (427 i 457 us)



## Aplicació de control dels retards de comunicacions

### Retard mesurat en la ranura 0, emissió de l'ASN

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

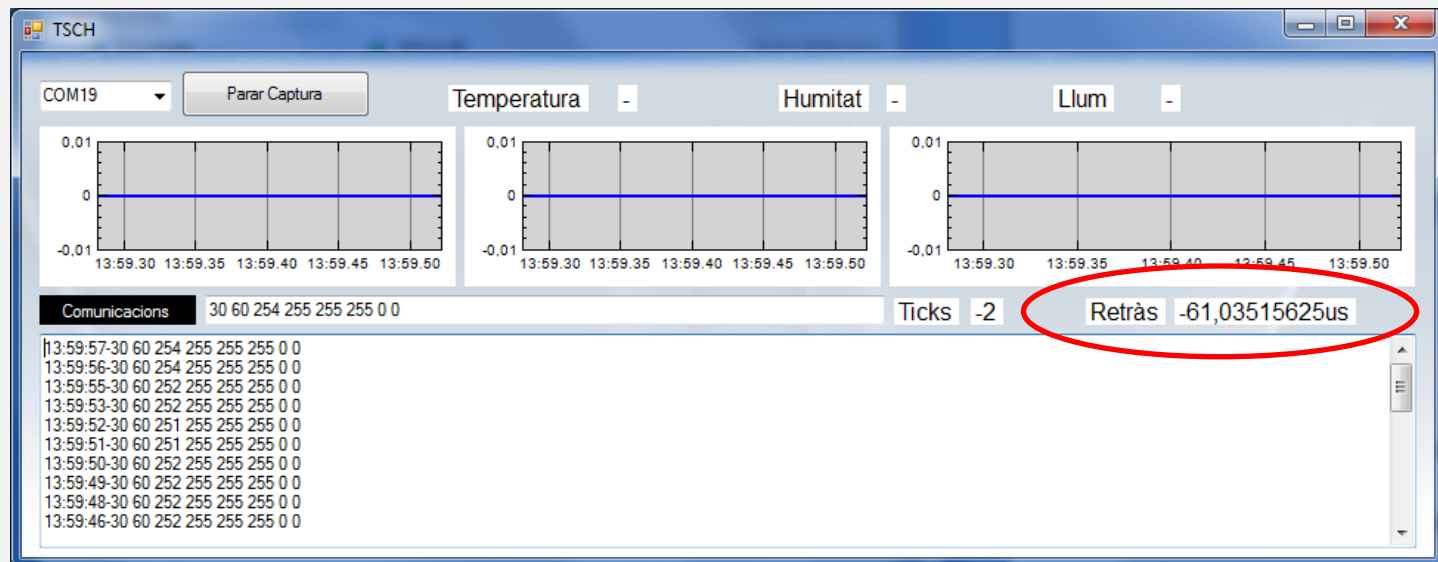




## Aplicació de control dels retards de comunicacions

### 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 entre -2 i -4 ticks (-61 i -122 us)



## **Aplicació de control dels retards de comunicacions**

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

## Conclusions

El sistema TSCH, es mostra com una alternativa viable per comunicacions de baixa capacitat, podent-se implementat en equips de baix cost i de forma molt flexible.

Es presenta com una alternativa molt viable per substituir xarxes RS485.

## Conclusions

El sistema TSCH, també és mostra com un bon sistema sobre el que muntar protocols més elaborats de comunicació, sobretot els basat en IPv6, per donar accés des d'Internet directament als equips.

# DUBTES



**MOLTES  
GRÀCIES PER  
LA SEVA  
ATENCIÓ**