Universitat de les Illes Balears

escola politècnica superior

Treball Final de Grau

GRAU D'ENGINYERIA ELECTRÒNICA INDUSTRIAL I AUTOMÀTICA

Implementació d'un node IoT de mesures ambientals amb connexió a Sentilo

MIQUEL HERNÀNDEZ NICOLAU

Tutor
Bartomeu Alorda Ladaria

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, 1 de setembre de 2016

Als meus pares i germans,
a en Dani, a n'Hèctor,
i a tots els que m'heu empès.

SUMARI

ACRÒNIMS

DHCP Dynamic Host Configuration Protocol

HTTP Hypertext Transfer Protocol

I^2C Inter-Integrated Circuit

IDE Integrated Development Environment

IoT Internet of Things

IP Internet Protocol

ISR Interrupt Service Routine

MV Màquina Virtual

PA Punt d'Accés

PCB Printed Circuit Board

RTC Real Time Clock

SMB System Management Bus

SMD Surface-mount Device

SO Sistema Operatiu

SSOP Shrink Small Outline Package

TCP Transport Control Protocol

TIC Tecnologies de la Informació i la Comunicació

UART Universal Asynchronous Receiver-Transmitter

Resum

El present document tracta el tema de les xarxes de sensors, emmarcat dins el camp de l'Internet of Things (Internet de les coses). En ell s'exposa el disseny a nivell físic i de software d'un node amb diferents sensors, així com la seva posada en funcionament.

El dispositiu dissenyat pren mesures de temperatura i humitat relativa, així com de nivell d'il·luminació i de contaminació. Inclou un botó de reset i una entrada de connector per poder realitzar la seva programació.

Totes les tasques són organitzades i comanades pel microcontrolador PIC, que executa el programa. Per al disseny d'aquest software s'hi han creat diferents llibreries i funcions específiques, que es van cridant des del programa principal.

Dins aquest programa principal s'executa un conjunt de configuracions inicials, per a després entrar en un bucle sense fi. En aquest bucle s'executa una tasca diferent cada minut, fins que al minut 15 es trameten les dades i torna a començar el cicle.

La comunicació del node es realitza a través de xarxa Wi-Fi. D'aquesta forma la informació presa és enviada a la plataforma Sentilo, que proporciona una base de dades entre d'altres funcionalitats.

CAPÍTOL 1

INTRODUCCIÓ

1.1 Proposta inicial

El context incial de la proposta de projecte presentada al cap d'estudis és el següent:

> «El creixement exponencial de la tecnologia sense fils basada en ZigBee fa que aprendre els seus secrets sigui de vital importància pel desenvolupament professional futur. Es tracta d'una tecnologia molt interessant amb multitud de propostes en el mercat i que en aquests moments es troba en expansió. Es tracta d'una pila de protocols per a comunicacions sense fils de baixa transferència de dades i dirigida especialment al camp de la domòtica. La seva integració amb dispositius avançats com els telèfons de darrera generació la fa especialment atractiva.»

Els objectius del projecte són:

> «Desenvolupar plataformes novedoses que permetin ser usades tant per monitoritzar com per a controlar espais industrials oberts mitjançant comunicacions sense fils basades en ZigBee.»

1.2 Enfocament final per a la realització

En dur a terme el projecte s'ha decidit emprar un protocol diferent de comunicació sense fils: Wi-Fi enlloc de ZigBee. No obstant això, el plantejament de la resta segueix sent molt semblant, ja que no s'arriba a sortir del camp de l'Internet of Things (IoT). L'objectiu ha estat modificat lleugerament, en tant que no només serviria per espais industrials, si no que estaria enfocat cap un ús més generalitzat.

L'objectiu final, per tant, és el disseny i desenvolupament d'un dispositiu per a la monitorització de l'espai en què es trobi, prenent mesures periòdiques de diferents

paràmetres ambientals. A més a més, es vol un funcionament inalàmbric via Wi-Fi, per tenir la capacitat d'enviar les mesures a una base de dades per tenir-ne un registre.

CAPÍTOL 2

ANÀLISI DE SOLUCIONS

2.1 El projecte dins del context actual

En la societat en què vivim les Tecnologies de la Informació i la Comunicació (TIC) s'estenen per tot cada cop més, esdevenint una necessitat en les persones deguda la seva funcionalitat. Dins el camp de les TIC s'hi troben les Smart Cities o Ciutats Intel·ligents. Aquest concepte es refereix a l'ús de les TIC com a mecanismes per millorar la gestió dels serveis d'una ciutat i, conseqüentment, la qualitat de vida dels seus habitants.

El medi de funcionament d'aquestes Smart Cities es basa en l'IoT. Aquest IoT planteja que diferents objectes puguin trobar-se interconnectats formant una xarxa d'objectes. Amb aquesta interconnexió podem saber quines són les seves condicions ambientals, si el seu funcionament és correcte, a quina posició es troba en un cert moment, etc.

El concepte d'IoT freqüentment va lligat a que els objectes tenen autonomia, és a dir, tenen font d'alimentació pròpia i la seva connexió a Internet és inalàmbrica. Per a realitzar aquesta transmissió sense fils actualment hi ha diferents tecnologies, que treballen majorment amb ones de ràdio: Wi-Fi, ZigBee, GPRS, LoRa, etc.

Per a la implementació de les TIC en les ciutats emprant IoT hi ha diferents opcions, i previsiblement cada cop n'hi haurà més degut a que el mercat de l'IoT és en expansió. Algunes d'aquestes opcions serien les que ofereixen Arduino (placa MKR1000) o Libelium (placa Waspmote), per exemple.

Per al projecte que s'ha desenvolupat no s'ha triat cap de les opcions que ja vénen confeccionades, si no que ha tractat d'experimentar i provar una forma diferent de posar en funcionament un node IoT. Les possibilitats són majors podent triar els components que conformen el node i partint des d'un nivell més baix de disseny.

2.2 Característiques principals del node

2.2.1 Descripció

El projecte que es du a terme és la realització d'un node de sensors a integrar dins una xarxa amb altres semblants, i que comptarà amb sensors de temperatura, humitat, il·luminació i contaminació. Dit node tendrà un transmissor Wi-Fi per fer els enviaments de les dades preses, bé cap a Internet, o bé a dins la mateixa xarxa local. L'alimentació es realitzarà mitjançant bateria.

2.2.2 Estructura

En la Figura 2.1 es mostra quina és la idea principal del node, del qual se n'ha prescindit finalment el Real Time Clock (RTC). S'hi poden observar les diferents funcionalitats dividides en blocs.

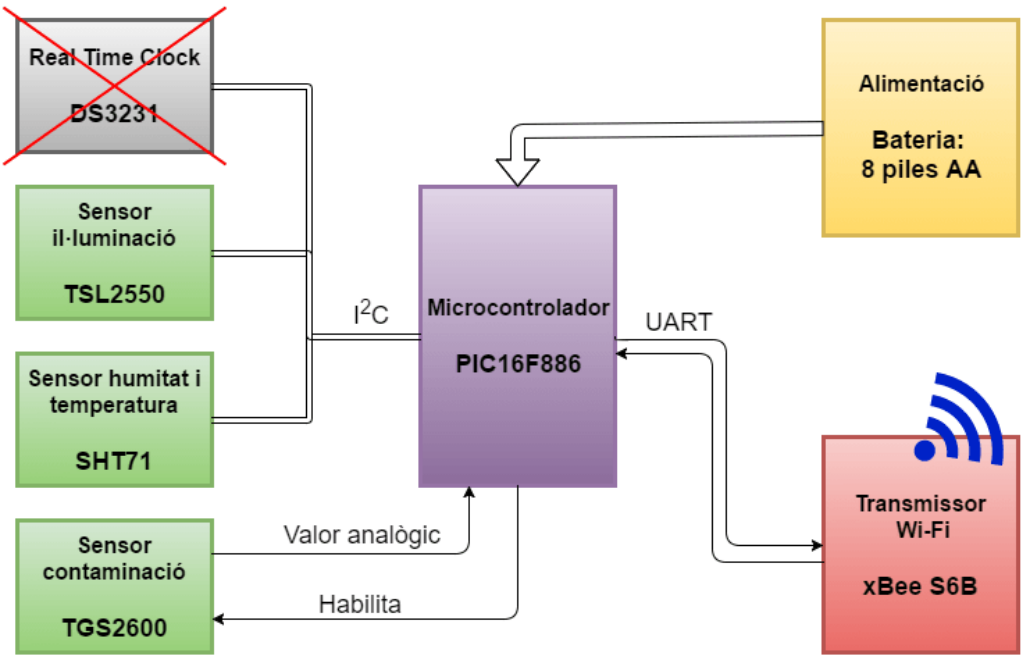

Figura 2.1: Diagrama de blocs del node.

També es veuen el tipus de comunicació entre cada bloc i el microcontrolador:

- Bus Inter-Integrated Circuit (I^2C) per comunicar amb el RTC i el sensor d'il·luminació. En aquest bus el microcontrolador fa de mestre en tot moment, mentre que el sensor i el RTC són esclaus. També s'hi troba connectat el sensor d'humitat i temperatura que, tot i no funcionar amb I^2C, és adaptable a aquest bus.
- Universal Asynchronous Receiver-Transmitter (UART) per comunicar amb el transmissor xBee. Aquesta comunicació permet canviar configuracions del transmissor, així com enviar dades directament cap a Internet degut al mode transparent del xBee.

- Senyal analògic del sensor de contaminació: Quan aquest component ha estat habilitat pel microcontrolador es pot llegir un valor analògic, que no és més que la sortida del sensor.

2.3 Eines externes al node

Són algunes parts externes al node, que són necessàries per al seu funcionament. Aquestes parts són la infraestructura Wi-Fi i la base de dades (veure Figura 2.2). S'han de tenir en compte abans de la posada en marxa del node, ja que s'hi han de fer les configuracions pertinents.

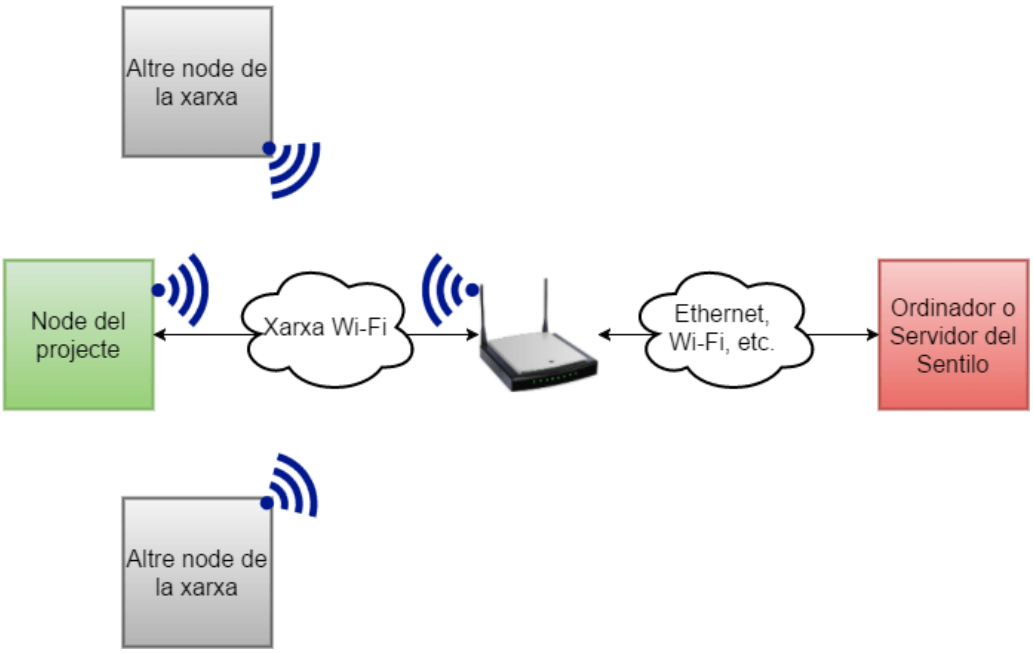

Figura 2.2: Connexió del node amb la base de dades.

2.3.1 Infraestructura Wi-Fi

Per a la connexió del node creat amb Internet és fonamental tenir una xarxa Wi-Fi amb accés permès i bon senyal. Això vol dir que el dispositiu no només dependrà d'ell mateix si no de la disponibilitat d'aquesta xarxa.

2.3.2 Plataforma de base de dades: Sentilo

El receptor de la informació és una base de dades. En aquest cas s'ha triat Sentilo, que és una plataforma que proporciona diferents eines per a tractament de dades i està enfocat a sistemes de sensors i actuadors en xarxa.

Proporciona una base de dades, una interfície gràfica, un catàleg de components i alarmes de lectures, entre d'altres funcions.[8] En general, és una eina que dóna moltes possibilitats al camp de les Smart Cities i l'IoT.

Per al seu funcionament, aquesta plataforma necessita ser instal·lada com a MV o com a Sistema Operatiu (SO) ordinari sobre un ordinador o servidor. Dins el Sentilo s'ha de donar d'alta el node amb els les característiques corresponents (Posició, sensors, unitats...). Òbviament, el Sentilo ha de ser en funcionament durant les trameses de dades, si no aquestes s'aniran perdent.

CAPÍTOL 3

DISSENY DE HARDWARE

3.1 Especificacions generals

A continuació s'exposen algunes de les característiques del hardware del node, per fer el pas de la teoria (els blocs de funcions) a la pràctica (el disseny del node a produir).

Per a l'**alimentació** del node es fan necessaris dos reguladors de tensió, ja que el sensor de contaminació només pot funcionar a 5V i el transmissor Wi-Fi només funciona a 3.3V. Es decideix que tot el node funcionarà amb un regulador de 3.3V, llevat del sensor de contaminació que tendrà el seu propi regulador de 5V.

Les característiques que se cercaven en el **microcontrolador** eren les següents:

- Capacitat de comunicar amb I^2C i UART, per poder interactuar amb els sensors digitals, el RTC i el transmissor.
- Ser capaç de llegir entrades analògiques, per la sortida del sensor de contaminació.
- Tenir entrades i sortides digitals per a possible control de reset d'alguns components, per encendre i apagar el sensor de contaminació, rebre algun senyal d'avís dels components, etc.
- Temporitzador intern, per poder realitzar interrupcions i comptar temps.
- Pin de reset: El microcontrolador s'ha de poder resetejar amb un botó.

En quant al **sensor de contaminació**, se cercava poder-lo encendre i apagar. Com aquest component no té directament aquesta opció, s'empra un regulador que en permeti l'apagada. D'aquest mode activant i desactivant el regulador de 5V es controla

també el sensor, sense afectar a la resta de node.

En quant als **sensors digitals** (il·luminació i temperatura/humitat) i **RTC**, només es requereix que puguin funcionar damunt un bus I^2C. La placa està dissenyada per poder rebre un valor digital d'alarma i poder efectuar un reset al sensor SHT31 de temperatura/humitat. Tanmateix, aquestes funcions han quedat inutilitzades en emprar el sensor SHT71 en lloc del SHT31.

3.2 Components

3.2.1 Microcontrolador: PIC16F886 [1]

Figura 3.1: Model de microcontrolador emprat.

Es tracta d'un microcontrolador PIC com el de la figura 3.1, del fabricant Microchip. És de 28 pins i 8 bits, amb el disseny Shrink Small Outline Package (SSOP), a soldar damunt superfície. Pot ser alimentat entre 2V i 5.5V

Aquest dispositiu disposa, entre d'altres, de les característiques que se cercaven, exposades a la secció anterior. Aquestes són: entrades analògiques i digitals, comunicació I^2C i UART, temporitzadors i pin de reset.

A l'apartat 4.1 s'exposa el procés de programació d'aquest microcontrolador, així com el programa complet que se li ha carregat a l'annex A.

3.2.2 Mòdul Wi-Fi: XBee S6B [2]

Aquest component és el transmissor Wi-Fi del node de sensors, fabricat per Digi, com el que es pot observar a la figura 3.2. S'ha d'alimentar a 3.3V per a un correcte funcionament. Dóna molta flexibilitat degut als seus paràmetres de configuració, a les entrades i sortides analògiques i digitals de les que disposa, als varis modes de comunicació, etc.

Com la major part dels valors de la seva configuració no han de canviar, aquesta es realitza per separat del node: Es connecta directament a l'ordinador i es carrega la configuració a través del programa XCTU. Veure apartat 4.2 per a més detalls del software.

Figura 3.2: Transmissor utilitzat.

En funcionament al node, es duen a terme dos tipus de comunicació per l'UART entre microcontrolador i transmissor: El microcontrolador envia comandes AT per demanar o modificar paràmetres de configuració, o bé envia dades per ser transmeses a l'Internet Protocol (IP) destí corresponent.

3.2.3 Sensor de temperatura i humitat: SHT71 [3]

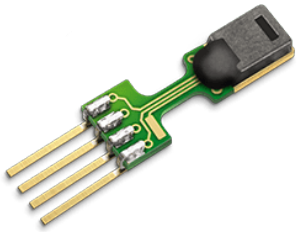

Figura 3.3: Sensor de temperatura i humitat relativa.

Fabricat per la companyia Sensirion, el SHT71 és un sensor digital com el que es mostra a la figura 3.3, que mesura temperatura i humitat. Transmet la informació mitjançant bus de dades amb un protocol propi (SENSI-BUS), però que és integrable a un bus I^2C. L'alimentació ha de ser entre 2.4V i 5.5V.

La temperatura té una precisió de ±0.4°C a temperatura ambient. Per altra banda, el sensor d'humitat té una precisió típica de ±3.0%.

3.2.4 Sensor d'il·luminació: TSL2550 [4]

Aquest sensor fabricat per TAOS (actualment AMS) té, de fet, dos fotodiodes (Ch0 i Ch1) amb dues respostes espectrals diferents: Un és sensible a la llum visible i infrarroja, i l'altre només a l'infrarroja, per compensar els efectes del primer. Per a obtenir la il·luminació corresponent a l'espectre visible s'ha d'aplicar una fórmula amb els valors obtenguts dels fotodiodes. Aquesta fórmula (veure equació 3.1) es proporciona al mateix datasheet del component i s'ha d'aplicar al programa del microcontrolador.

$$Light\ level\ (lux) = (Ch0\ counts) \cdot 0.46 \cdot e^{\left(-3.13 \cdot \frac{Ch1\ counts}{Ch0\ counts}\right)} \quad (3.1)$$

L'aplicació d'aquesta equació 3.1 fa que per als valors màxims dels fotodiodes (Chx counts = 4015) el nombre de lux es situi en 81 lux, com es mostra a l'equació 3.2. Això no vol dir que s'estigui a 81 lux, sinó que s'està molt per damunt però el sensor s'ha saturat. És un efecte advers d'aquest sensor.

$$Light\ level\ (lux) = 4015 \cdot 0.46 \cdot e^{\left(-3.13 \cdot \frac{4015}{4015}\right)} = 1846.9 \cdot e^{-3.13} = 80.7424...\ lux \quad (3.2)$$

Figura 3.4: Sensor d'il·luminació.

En quant al funcionament a la placa, de la mateixa manera que el SHT71, el TSL2550 és un sensor digital i es comunica amb el microcontrolador mitjançat un bus I^2C, emprant un protocol que és un subconjunt de l'I^2C: el System Management Bus (SMB). El component es pot observar a la figura 3.4. La seva alimentació ha de ser entre 2.7V i 5.5V.

3.2.5 Sensor de contaminació: TGS2600 [5]

El fabricant Figaro proporciona aquest sensor analògic de la figura 3.5. És sensible a diferents gasos contaminants com poden ser l'hidrogen, el monòxid de carboni, l'isobutà, l'etanol o el metà.

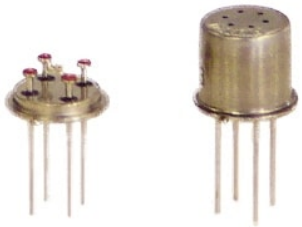

Figura 3.5: Sensor de contaminació per dins (esquerra) i amb l'encapsulat (dreta).

Consta d'un escalfador intern per poder realitzar les mesures a temperatura constant. Ambdós escalfador i sensor es troben alimentats a 5V, sent l'únic component del disseny que requereix aqueix voltatge i comptant amb un regulador per ell sol (veure apartat 3.2.7).

3.2.6 Regulador de 3.3V: NCP1117 [6]

Figura 3.6: Encapsulat del regulador de 3.3V.

Aquest component d'ON Semiconductor, semblant al de la figura 3.6, permet passar del voltatge d'alimentació proporcionat per la bateria als 3.3V d'alimentació de la majoria de components de la placa. A més a més, ofereix una major estabilitat del voltatge, esmorteint les oscil·lacions que es puguin produir a l'entrada.

3.2.7 Regulador de 5V: MIC5225 [7]

Fabricat per Micrel (ara de Microchip), aquest regulador dóna una sortida de 5V a partir de l'entrada d'alimentació. Es mostra a la figura 3.7.

Figura 3.7: Regulador de 5V.

Addicionalment consta d'un pin per habilitar-lo o deshabilitar-lo. Això és prou útil al node, en tant que així es pot desconnectar el sensor de contaminació (veure apartat 3.2.5) durant tot el temps que no s'està utilitzant. Cal recordar que dit sensor analògic incorpora un encalentidor que consumeix un corrent important, d'aproximadament 50 mA.

3.2.8 Alimentació: Piles d'1.5V

L'alimentació del node consta de vuit piles AA d'1.5V. Tot i així, aquesta entrada pot ser substituïda per qualsevol que proporcioni entre 6V i 16V en corrent continu.

3.2.9 Connector de programació

Es tracta d'una entrada RJ12 com la de la figura 3.8. A ella es connecta el programador del microcontrolador, el MPLAB ICD 2. (Més informació del ICD2 a l'apartat 4.1)

Figura 3.8: Connector per RJ12.

3.2.10 Components passius

El disseny inclou quatre resistències: Dues de 10kΩ per a pull-up del bus I^2C, una de 12kΩ que fa de divisor de tensió a la sortida del sensor TGS2600, i una altra de 10kΩ emprada per al botó de reset.

També es compta amb diferents condensadors per esmorteir els diferents senyals i estabilitzar-los. Al muntatge amb el qual s'han realitzat les proves sols se n'han inclòs a l'entrada i sortida del regulador de 5V, així com al botó de reset. El node ha funcionat correctament només amb aquests tres condensadors.

3.2.11 Components descartats

Hi ha un conjunt de components que s'han descartat finalment per diferents raons. Són els següents:

- **PIC18F2580 (Microcontrolador):** Era l'opció alternativa al PIC16F886. Tanmateix el PIC16F886 ja complia amb els requeriments del projecte, de mode que no va caldre emprar aquest model superior.
- **DS3231 (RTC):** Va deixar de ser necessari en tant que es va decidir emprar la Plataforma Sentilo per a base de dades, la qual ja inclou la data i hora en rebre dades.
- **CHIPCAP (Sensor temperatura-humitat):** Es va descartar aquest component degut a que la seva comunicació emprava protocol Manchester i no era compatible amb el bus I^2C. Substituït pel SHT31.
- **SHT31 (Sensor temperatura-humitat):** A l'hora de soldar-lo a la placa el component no havia arribat encara i es va prendre la decisió de substituir-lo per un altre de la seva família que era disponible: el SHT71.

3.3 Disseny dels circuits

Per al disseny del circuit emprat al projecte es va usar EAGLE[9], que és un software específic per això. A la figura 3.9 es pot veure la versió final del disseny i damunt d'ell quina és la distribució dels blocs funcionals del node. Es mostra així per tal d'identificar

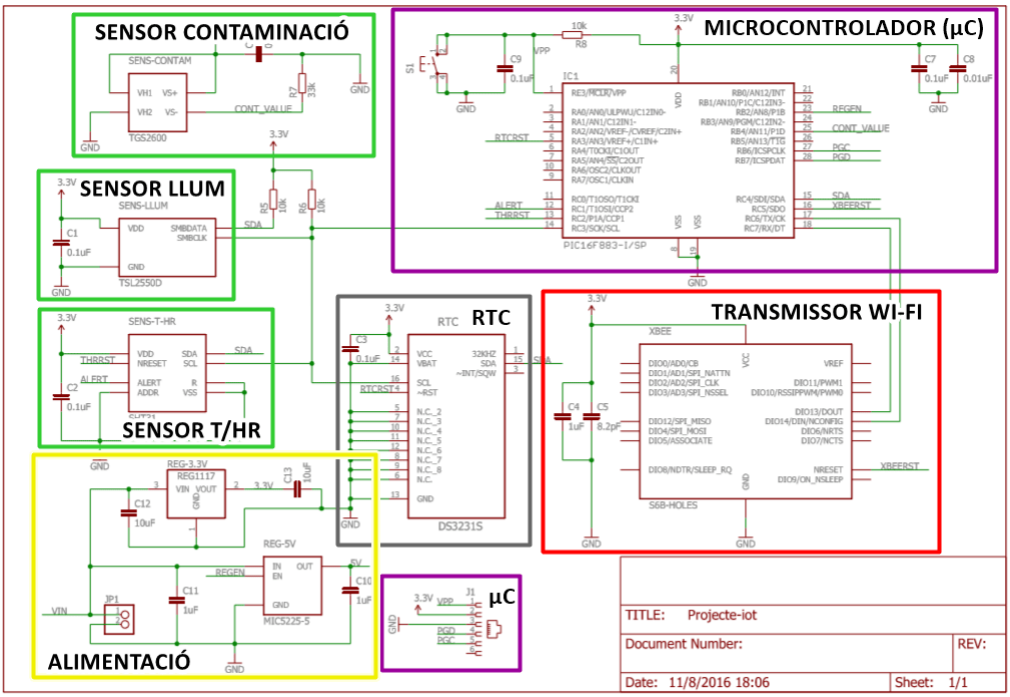

Figura 3.9: Diferents blocs sobre el circuit.

els blocs ràpidament i tenir-ne una primera idea.

En aquesta figura 3.9 cal fixar-se en què els dos resistors sense agrupar corresponen als pull-ups de l'I^2C, que és el bus que comunica els esclaus (sensors digitals i RTC) amb el mestre (microcontrolador). Per altra banda, el lligam entre microcontrolador i transmissor és l'UART, que intercomunica ambdos components.

També cal notar que a sota del RTC hi ha un bloc que també correspon al microcontrolador: el connector de programació exposat anteriorment. Es veu també que l'alimentació no només consta d'entrada, sinó també dels dos reguladors, ja que són aquests els que proporcionen realment els dos voltatges necessaris a la placa: 3.3V (apartat 3.2.6) i 5V (apartat 3.2.7).

A la figura 3.10 es pot observar més ampliada la versió final d'aquest disseny. S'ha de tenir en compte que en aquesta versió final hi ha alguns components que hi apareixen que no s'han arribat a incloure:

- El RTC DS3231 i el sensor SHT31, per les raons exposades a l'apartat 3.2.11.
- La major part de condensadors, ja que el node no ha presentat problemes de funcionament sense haver-los afegit.
- La resistència de 33kΩ per al sensor de contaminació finalment va ser una de 12kΩ. Això és degut a que els valors de voltatge amb 33kΩ podien superar els

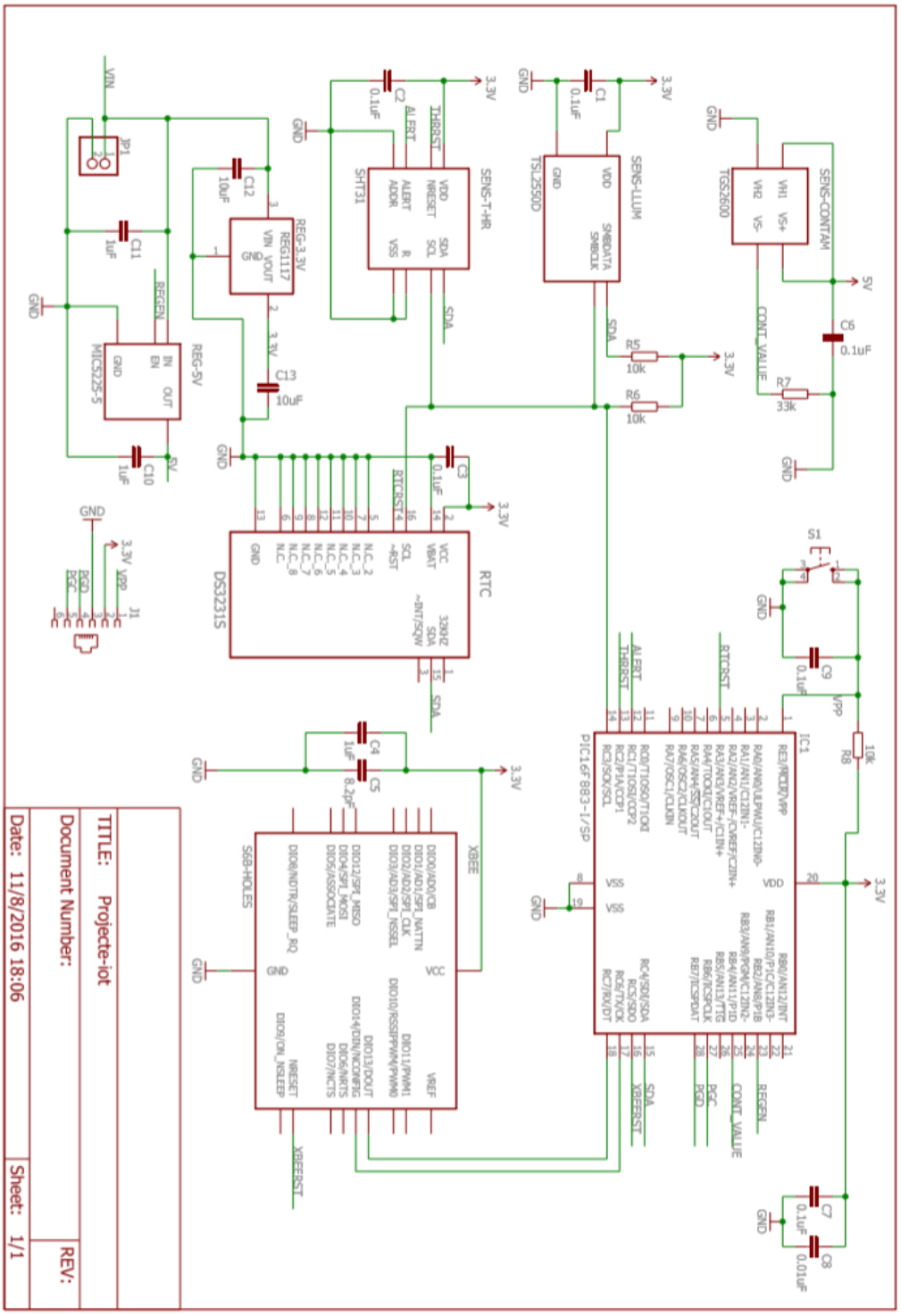

Figura 3.10: Circuit complet dissenyat amb Eagle.

límits del microcontrolador.

3.4 Disseny de la PCB

En aquest apartat es mostra com es va fer la disposició dels diferents components sobre la placa, i l'enrutat de les pistes. La majoria de footprints (que és l'espai físic a ocupar per un component) es troben a la cara superior (figura 3.11). Ara bé, els footprints del regulador de 5V (MIC5225) i els seus condensadors es troben a la cara inferior, així com la resistència i el condensador del sensor de contaminació (TGS2600). Es poden observar a la figura 3.12. A les dues imatges dels dissenys de PCB es tornen a mostrar els diferents blocs funcionals del node.

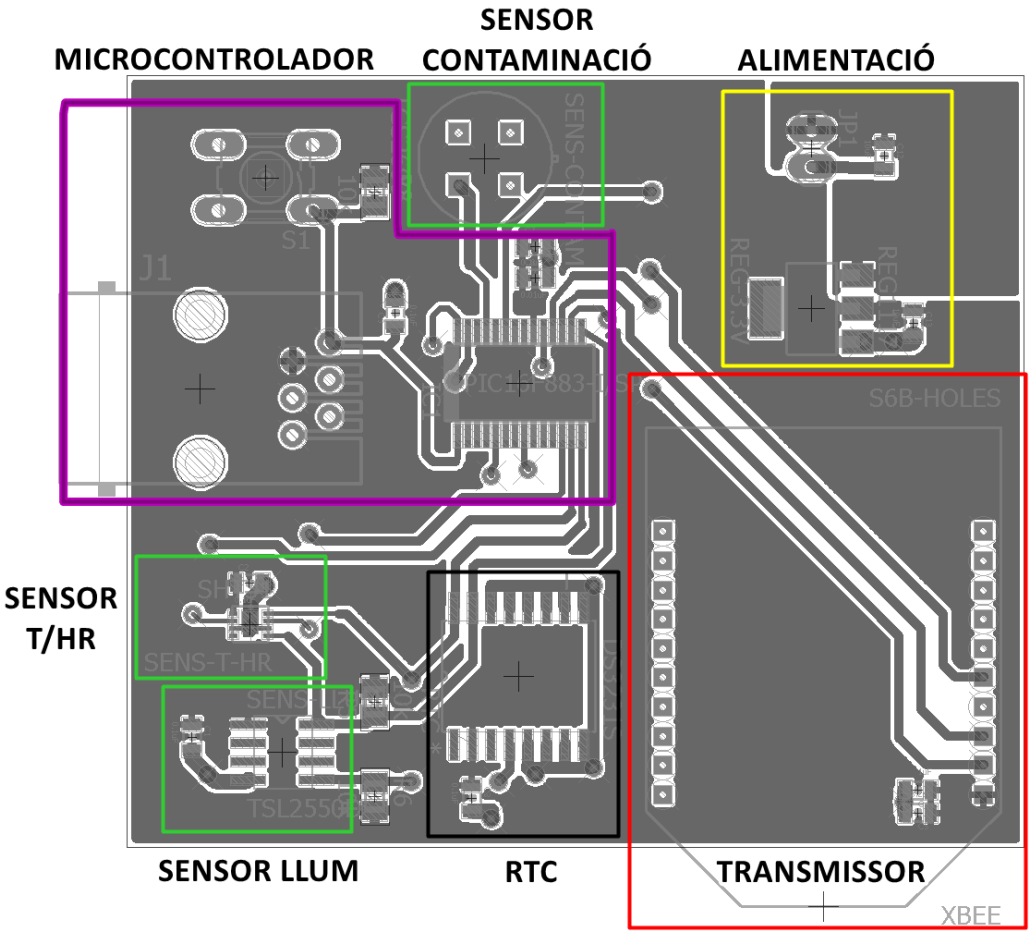

Figura 3.11: Disseny de la part superior de la PCB.

La distribució dels components sobre la placa s'ha realitzat seguint els següents criteris:

- El tranmissor xBee té la part d'antena sortint del pla de la placa i, per tant, lliure de cap component al davant.

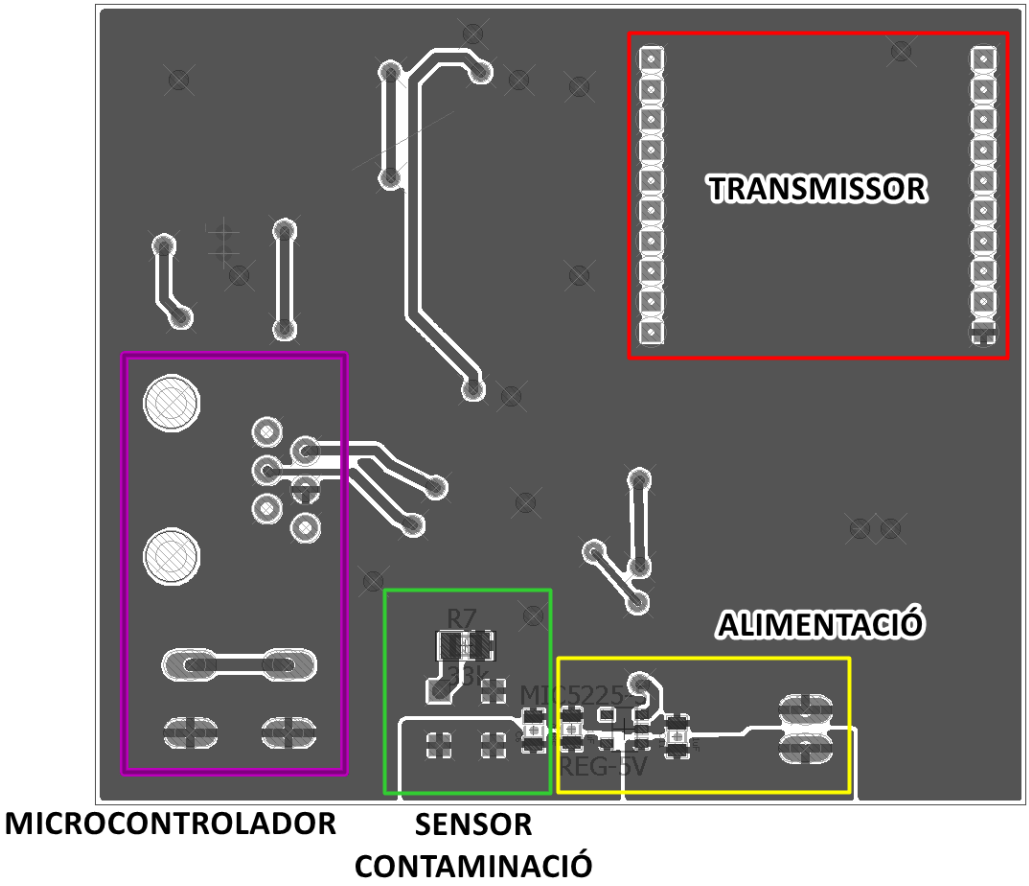

Figura 3.12: Disseny de la part inferior de la PCB.

- El connector de programació ha d'anar a un costat de la placa, orientat de mode que no hagi impediments per a poder connectar el node al programador.
- Els components comunicats amb bus I^2C es troben a la mateixa zona.
- Els condensadors són situats el més proper possible a les entrades d'alimentació dels seus respectius components.
- El sensor de contaminació, els reguladors i l'entrada d'alimentació es troben junts per minimitzar les zones de voltatges diferents a 0V i 3.3V.

També es pot veure, comparant les figures 3.11 i 3.12, que hi ha uns quants components els quals el seu disseny està preparat per travessar la placa. Aquests components són el transmissor xBee, el sensor de contaminació TGS2600, el connector de de programació, el botó de reset i l'entrada d'alimentació.

Cal notar a la placa final que hi va haver una errada en el disseny del footprint del xBee i el TGS2600. Degut a això, aquests components no caben dins els seus forats. No

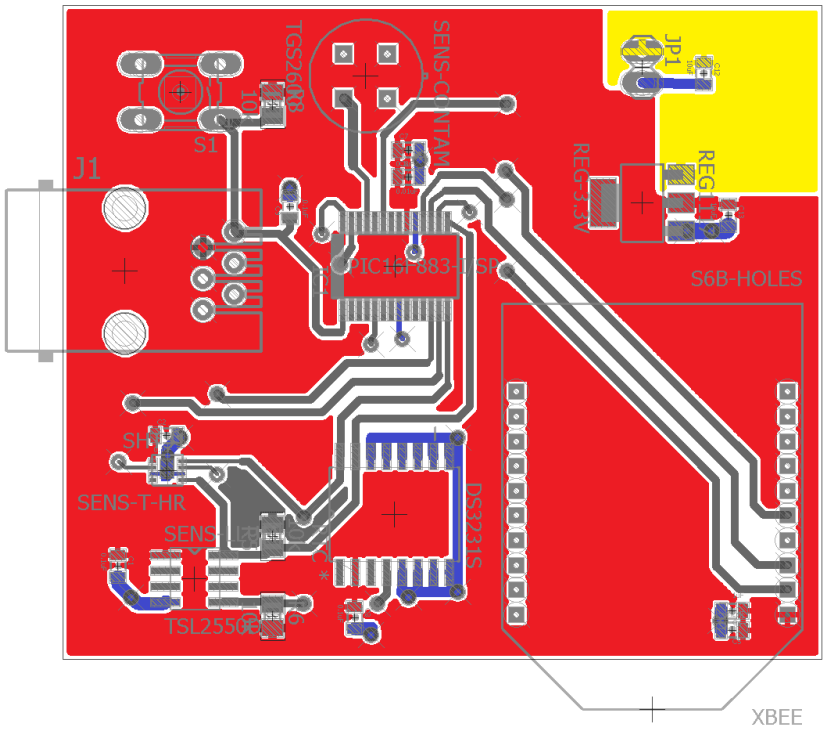

Figura 3.13: Distribució dels voltatges a la part superior de la PCB.

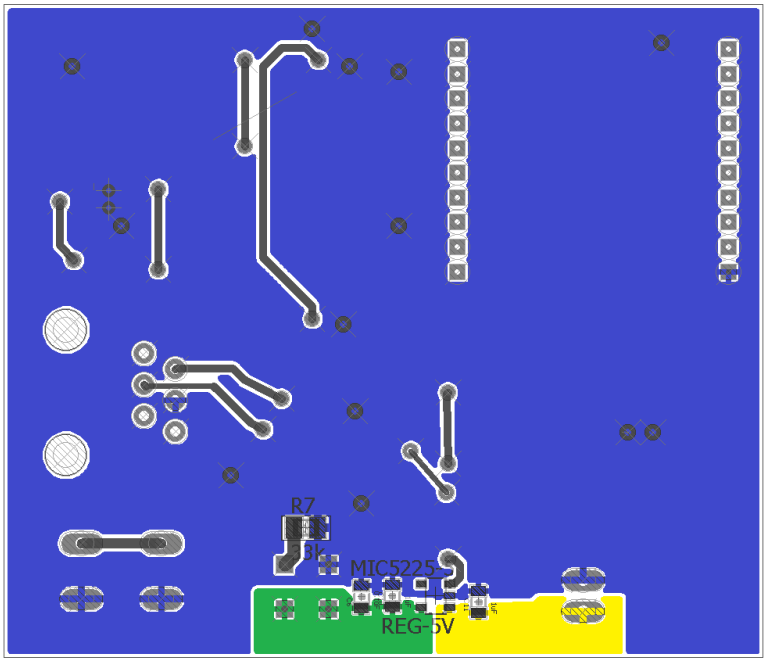

Figura 3.14: Distribució dels voltatges a la part inferior de la PCB.

obstant, el sensor de contaminació i dos sockets per encaixar-hi dins el xBee s'han pogut soldar als footprints corresponents a la part superior de la placa, sense travessar-la.

La distribució de voltatges a la placa s'ha realitzat de mode que la part superior fos a 3.3V i la part inferior a 0V. Ara bé, la zona del sensor de contaminació és a 5V, i la zona d'alimentació i entrada a reguladors és a voltatge d'alimentació (entre 6V i 16V). A les figures 3.13 i 3.14 es poden veure les zones de voltatge amb els següents colors: blau 0V, vermell 3.3V, verd 5V i groc alimentació.

A la part superior de la placa es pot notar que va haver una errada de disseny: Hi ha una zona desconnectada de la resta, que es mostra en gris a la figura 3.13, a l'esquerra del RTC. Això fa que una de les resistències de pull-up de l'I^2C quedàs a en circuit obert. Al següent apartat s'explica com s'ha solucionat això sobre la placa.

3.5 Model definitiu del node

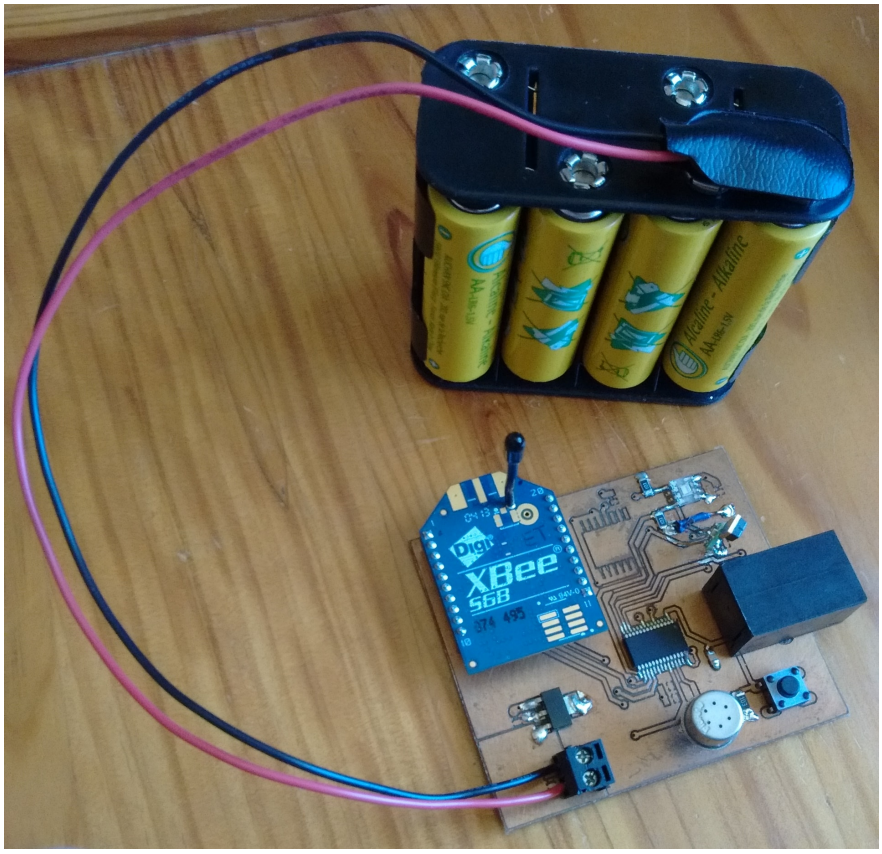

Figura 3.15: Vista general del node amb l'alimentació.

A la figura 3.15 es mostra el node definitiu emprat per a les proves, junt a la seva alimentació. A la dreta del transmissor xBee es veu la zona disponible per al RTC.

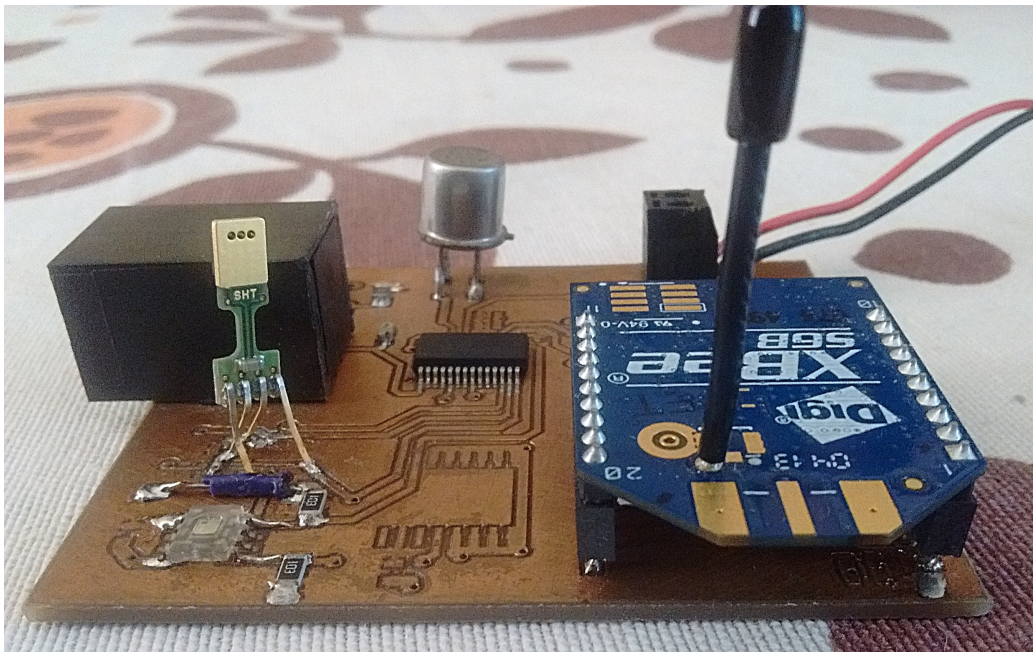

Figura 3.16: Vista lateral del node.

Sobre el muntatge final s'han efectuat dos canvis de disseny, que es troben a la banda esquerra de la figura 3.16. El primer dels canvis és referent a l'error que va deixar la resistència de l'I^2C en circuit obert. Això es va solventar soldant un cable entre la zona desconnectada i la zona de 3.3V.

L'altre canvi que s'ha realitzat és el muntatge del sensor de temperatura i humitat. De nou a la figura 3.16, és notori que la placa no estava pensada per a aquest sensor, que finalment ha estat posicionat en vertical. El footprint a la placa estava pensat per un sensor SHT31 que és Surface-mount Device (SMD); però finalment s'ha emprat el model SHT71, que és through-hole, i s'ha hagut de soldar aprofitant les zones i enrutats disponibles.

CAPÍTOL 4

DISSENY DE SOFTWARE

4.1 PIC16F886

4.1.1 Hardware i software emprats

MPLAB 8 IDE

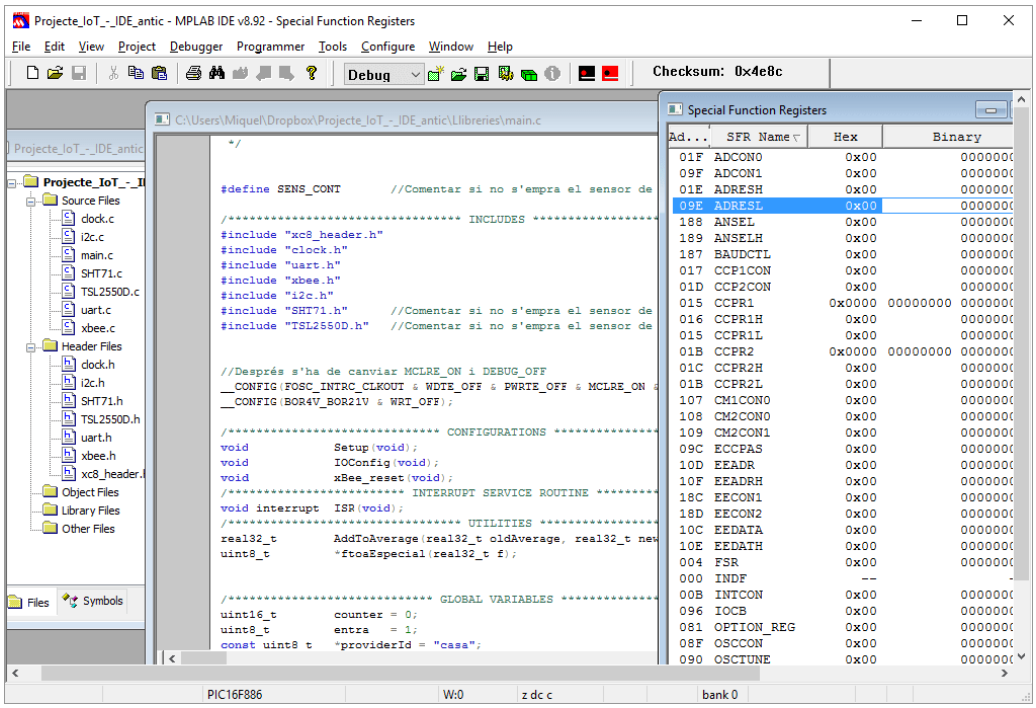

Figura 4.1: Mostra de l'IDE utilitzat.

L'IDE emprat per a la programació del PIC ha estat MPLAB 8 IDE (Figura 4.1) i el compilador HI-TECH. El codi es va començar en el MPLAB X IDE, amb el compilador XC8; no obstant, es va haver de passar a les versions més antigues abans esmentades per raons de compatibilitat amb el debugger/programador utilitzat.

ICD2

Aquest debugger que limita l'IDE és el MPLAB ICD 2. Permet testejar el programa durant el seu desenvolupament més avançat, i finalment carregar-lo dins el PIC16F886. En ser un debugger antic, l'ICD2 té diverses limitacions. Per exemple, no es poden realitzar canvis en el valor de les variables, ni tampoc es poden rebre sortides de text per pantalla. A més, l'ICD2 només permet definir un únic breakpoint, el qual afegeix més complicació al procès de depuració del codi.

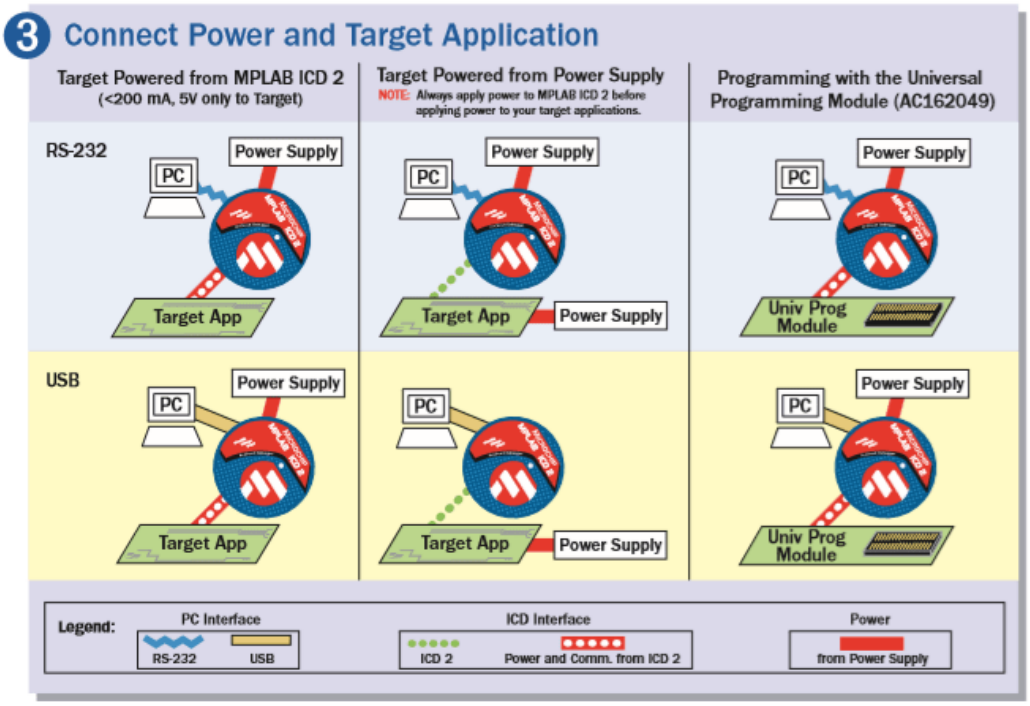

Figura 4.2: Possibilitats per emprar el debugger ICD2.[10]

El debugger no és capaç de fer funcionar el microcontrolador sense una font d'alimentació, ja que ell no la proporciona. Se li ha de proporcionar una font, bé a la placa directament, o bé endollant-li'n una a un entrada d'alimentació que té el propi ICD2. Això es pot veure a un tros d'una breu guia explicativa de l'ICD2[10], que es mostra a la figura 4.2.

Cal recordar que l'ICD2 és el motiu de tenir el connector de programació (Apartat 3.2.9) al node, ja que aquest debugger hi va connectat allà per a realitzar la programació.

4.1.2 Projecte a l'IDE

El llenguatge de programació en el que s'ha dissenyat el programa del PIC16F886 ha estat C, encara que és un C específic per aquests components. Això vol dir que al codi no hi ha classes, no hi ha tipus de variable boolean, i ni tan sols s'hi permeten tipus variable String. Els tipus de dades en coma flotant amb precisió simple (*float* o *real32_t*) s'empren el mínim possible degut a la gran quantitat d'espai de programa i dades que empren. Per aquest mateix motiu, els tipus de dades en coma flotant de precisió doble (*double*) directament ni s'empren al programa.

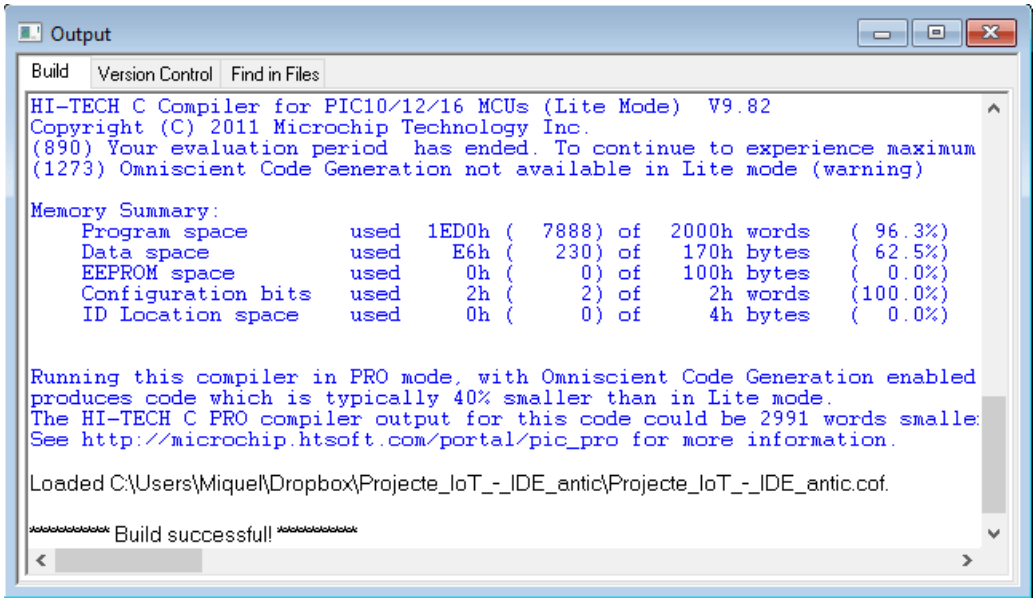

Figura 4.3: Resposta de l'IDE en compilar el programa del PIC.

Per a la creació del programa s'ha anat prenent com a base el diagrama de flux que s'exposarà a l'apartat següent. A mesura que s'ha necessitat el funcionament de les diferents parts s'ha anat muntant el programa, creant noves llibreries, rectificant errors i, finalment, s'ha hagut d'ajustar molt a l'espai de memòria. A la figura 4.3 es mostra com d'ajustat ha quedat el programa a l'espai de memòria, així com el missatge de que amb una llicència el programa es podria haver optimitzat devers un 40%.

Com a altre exemple dels entrebancs trobats en el disseny del software, hi ha el desbordament de la pila (stack overflow) en la part de codi d'enviaments a Sentilo. Per aquest motiu, s'han hagut de dur a terme canvis molt importants en aquesta part del codi evitant en mesura del possible totes les cridades a funcions.

Un cop s'ha tengut el programa llest per a les seves proves o per a carregar-lo s'ha seleccionat el debugger ICD2 a la pestanya de Debugger o Programmer, segons correspongués. Fet això es compila el projecte de l'IDE i seguidament es carrega per a poder depurar-lo o tenir el PIC en funcionament autònom.

4.1.3 Estructura del programa

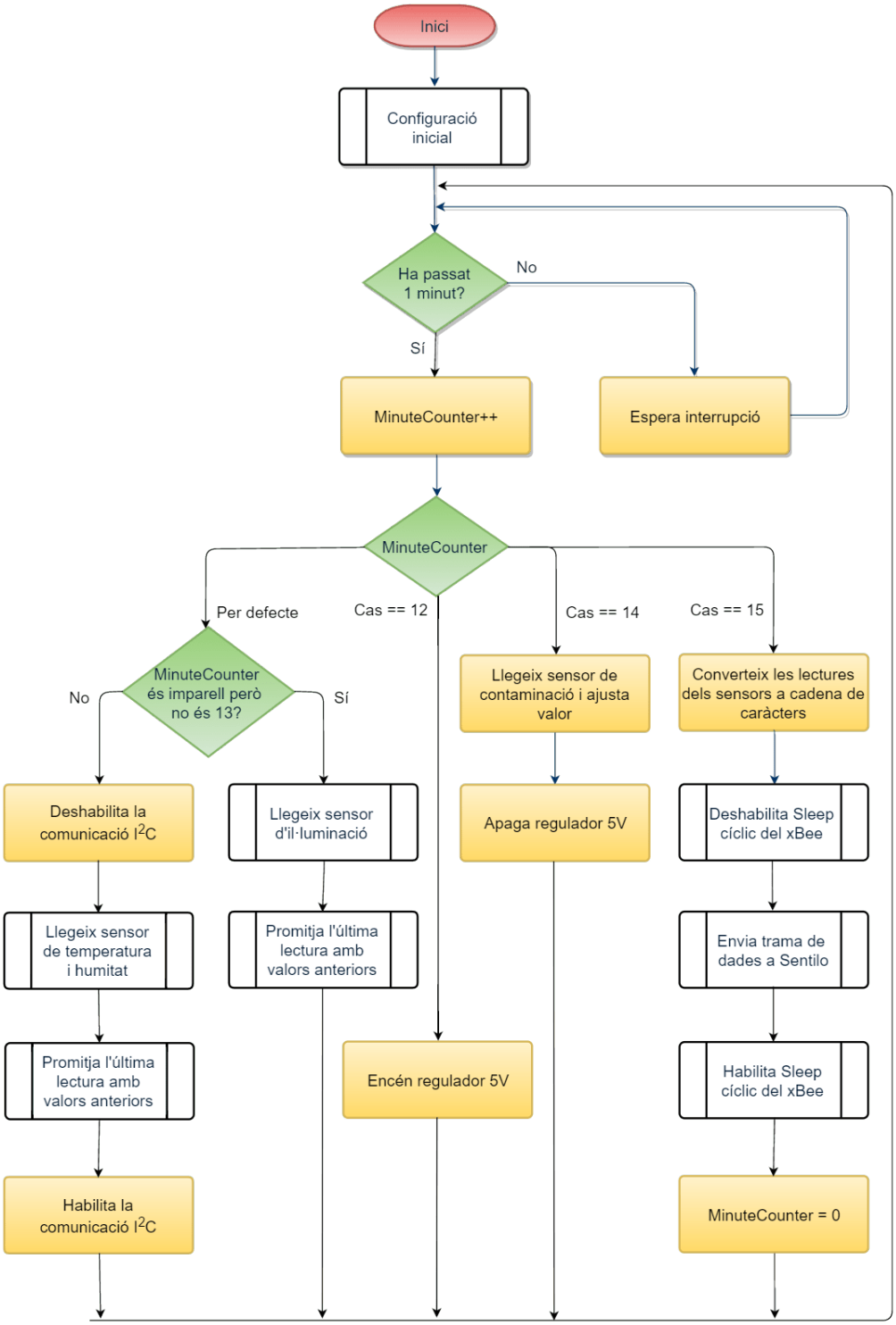

Figura 4.4: Diagrama de flux del programa principal.

Característiques

A la figura 4.4 es pot veure quina és la idea principal del funcionament del programa. Bàsicament, el que fa en inciar-se és un reseteig i una configuració general de tots els paràmetres, dels sensors que ho necessiten, dels perifèrics, etc. Un cop duit a terme això, el programa entra en un bucle sense fi en el qual es fa polling per saber si ha passat un minut. Aquest polling és degut a la impossibilitat del microcontrolador d'entrar en mode Sleep sense un oscil·lador extern, fet que no es va preveure en el disseny inicial del hardware.

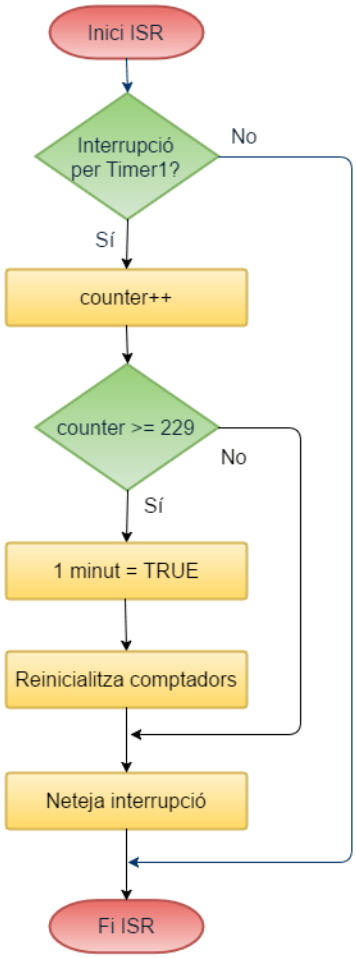

Figura 4.5: Diagrama de flux de l'ISR.

Paral·lelament a aquesta comprovació contínua de que hagi passat un minut, es va activant l'ISR (Figura 4.5). Dins aquesta ISR es comprova que ha succeït una interrupció pel Timer 1 (El timer ha estat activat prèviament a les configuracions incials), i va augmentant un comptador. Quan s'arriba a un minut, el comptador de l'ISR es reinicialitza i es dóna valor a una variable que indica que s'ha complert un minut.

Una vegada que al programa principal es compleix la condició del minut, es realitza una selecció de la tasca a dur a terme. Aquesta tasca depèn del minut en que es troba, amb la variable MinuteCounter. Les possiblitats serien les següents:

- **MinuteCounter val 1, 3, 5, 7, 9 o 11:** Lectura del sensor d'il·luminació i promitjat amb els valors anteriors.
- **MinuteCounter val 2, 4, 6, 8, 10 o 13:** Lectura del sensor d'humitat i temperatura, i promitjat amb els valors anteriors.
- **MinuteCounter val 12:** Encesa del regulador de 5V i, per tant, del sensor de contaminació.
- **MinuteCounter val 14:** Lectura del sensor de contaminació i apagada del regulador de 5V.
- **MinuteCounter val 15:** Conversió dels valors a caràcters, despertar del xBee, enviament a Sentilo, posada en sleep del xBee i reinici de MinuteCounter.

Llibreries

A la taula 4.1 s'explica breument la funció que té cada una de les llibreries que s'han inclòs al projete en MPLAB. Per a tenir-ne una major informació es pot revisar en pronfunditat el codi a l'annex A. Allà s'hi adjunta el codi que està completament documentat en format Doxygen.

Parts remarcables del programa

En aquest apartat es tracten algunes parts del programa que són d'especial interès deguda a la complexitat que puguin tenir o l'estranyesa que puguin causar al que no n'hi estigui familiartizat.

- Per a realitzar el pas de valors a cadena de caràcters s'aplica una conversió. Donada la numeració d'emmagatzematge dels valors dels sensors (0: Il·luminació, 1: Humitat, 2: Temperatura, 3: Contaminació), per a la il·luminació es converteix el nombre com sencer.

  Això és així perquè el valor d'il·luminació és l'únic que no es va considerar que necessitàs més precisió. Per a la resta de valors es conserva un decimal, emprant una funció *ftoa()* modificada amb aquest propòsit. El codi es troba al *case 15* del bucle sense fi de *main.c* i és el següent:

```
1     for(uint8_t i = 0; i < 4; i++){
2         if(i){
3             help = ftoaEspecial(sensorValue[i]);
4             sprintf(SentiloXBee_sensVal[i], "%s", help);
5         }else   //Only for lighting sensor
6             sprintf(SentiloXBee_sensVal[i], "%u", (uint16_t)sensorValue[i])
      ;
7     }
```

Nom	Fitxers	Descripció
Rellotge	clock.h clock.c (Annex A.3)	Correspon al control del rellotge del microcontrolador. És una llibreria mínima ja que només inclou una funció, però la idea seria ampliar-la si es disposa de més memòria.
I^{2}C	i2c.h i2c.c (Annex A.4)	Funcions de control del bus I^{2}C. El codi original és pres d'una pàgina web[11], tot i que ha sigut modificat i ampliat per a cobrir les necessitats del projecte.
UART	uart.h uart.c (Annex A.5)	Aquesta llibreria inclou les funcions per manejar l'UART del microcontrolador. Igual que al punt anterior, aquest codi ha estat pres d'un lloc web[12] i s'ha adaptat a les necessitats del node.
XBee i Sentilo	xbee.h xbee.c (Annex A.6)	Inclou les funcions necessàries per fer funcionar el transmissor xBee i fer trameses al Sentilo. Realment no arriba a ser una llibreria com a tal, ja que per raons d'espai al microcontrolador el codi es troba limitat a funcionar amb projecte que s'exposa. El codi està composat a partir de dues fonts: La primera són funcions per al xBee extretes d'un codi per a Arduino facilitat pel tutor; i la segona font del codi és una funció per a l'enviament a Sentilo, que és una reducció molt gran del tutorial d'Arduino amb Ethernet, disponible a la web de Sentilo[13]. Per a funcionar, aquest codi empra la llibreria de l'UART, tot i que s'han separat les funcions que criden a l'UART per a facilitar els canvis a un altre tipus de comunicació.
SHT71	SHT71.h SHT71.c (Annex A.7)	És la llibreria per al maneig del sensor de temperatura i humitat amb SENSI-Bus (el tipus de comunicació d'aquest sensor, integrable a un bus I^{2}C). Es va obtenir de la pròpia pàgina web de Sensirion (el fabricant)[3], a l'apartat de descàrrega de documents. S'ha adaptat el codi per al PIC, ja que no cabia dins la memòria i, a més, el funcionament amb el sensor depèn de la velocitat d'execució del programa.
TSL2550	TSL2550D.h TSL2550D.c (Annex A.8)	Aquesta llibreria, creada des de zero, serveix per manejar el sensor d'il·luminació. Necessita la lliberia de l'I^{2}C per funcionar.

Taula 4.1: Llibreries del projecte a MPLAB.

- Per aplicar l'equació del sensor TSL2550 (Equació 3.1) es va decidir fer una linealització de la funció exponencial. Això és degut a que cridar a la funció *exp()* de la llibreria *math.h* ocupava aproximadament el 20% de la memòria del PIC, espai inassumible per a l'aplicació.

  Per fer aquesta linealització s'ha calculat l'exponencial a set punts diferents: Variables *exp#* i 1 (e^0). Llavors, amb aqueixos valors s'han creat set funcions lineals. L'error és molt menor que amb una sola lineal i, a més, segueix quedant molt lluny d'ocupar el que ocupa la funció *exp()*. El tros codi és el següent, trobat dins la funció *LightSensor_Read()* a la llibreria del TSL2550:

```
1     temp1 /= temp0; //temp1 = ADC1/ADC0
2     temp1 *= 3.13;  //temp1 = (ADC1/ADC0)*3.13
3
4     //Linealitzam exp(temp1)
5     if(temp1 > 5.5)
6         temp1 = exp6 * (temp1 - 5);
7     else if(temp1 > 4.5)
8         temp1 = exp5 * (temp1 - 4);
9     else if(temp1 > 3.5)
10        temp1 = exp4 * (temp1 - 3);
11    else if(temp1 > 2.5)
12        temp1 = exp3 * (temp1 - 2);
13    else if(temp1 > 1.5)
14        temp1 = exp2 * (temp1 - 1);
15    else if(temp1 > 0.5)
16        temp1 *= exp1;
17    else
18        temp1++;
19
20    return (uint16_t)(temp0 * 0.46 / temp1);
```

- Per als enviaments de dades a Sentilo es crida a una sola funció des del programa principal: *SentiloXBee_publishObservation(providerId, sensors, apiKey)*. Aquesta funció és tot el que queda de la llibreria disponible a Sentilo per a Arduino[13]. El que fa és muntar una petició Hypertext Transfer Protocol (HTTP) amb el mètode PUT, emprant les diferents constants i variables necessàries.

  Aquests trossos de la petició HTTP es van escrivint a l'UART, fent que s'enviïn al transmissor i aquest ho vagi enviant per Wi-Fi cap a l'enrutador a la direcció que està configurada al transmissor. Més endavant, a l'apartat de Sentilo d'aquest capítol es tracta més en profunditat la trama. Com a mostra de com es va realitzant l'enviament tenim el següent codi, que és la part inicial de la petició HTTP:

```
1     SentiloXBee_write("PUT ");
2     SentiloXBee_write(DATA_BASE_PATH);
3     SentiloXBee_write("/");
4     SentiloXBee_write(providerId);
5     SentiloXBee_write(" HTTP/1.1\r\n");
6
```

```
7       SentiloXBee_write(IDENTITY_KEY_HEADER);
8       SentiloXBee_write(": ");
9       SentiloXBee_write(apiKey);
10      SentiloXBee_write("\r\n");
```

Cal notar que la funció *SentiloXBee_write()* l'únic que fa és cridar a *UART_Write_Text()*.

4.2 xBee S6B

4.2.1 Hardware i software emprats

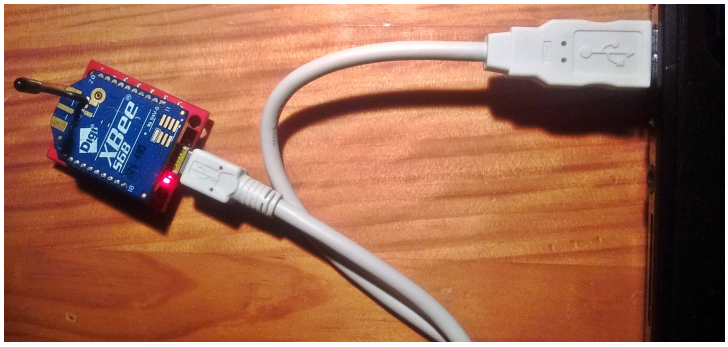

Figura 4.6: Transmissor xBee connectat a l'ordinador.

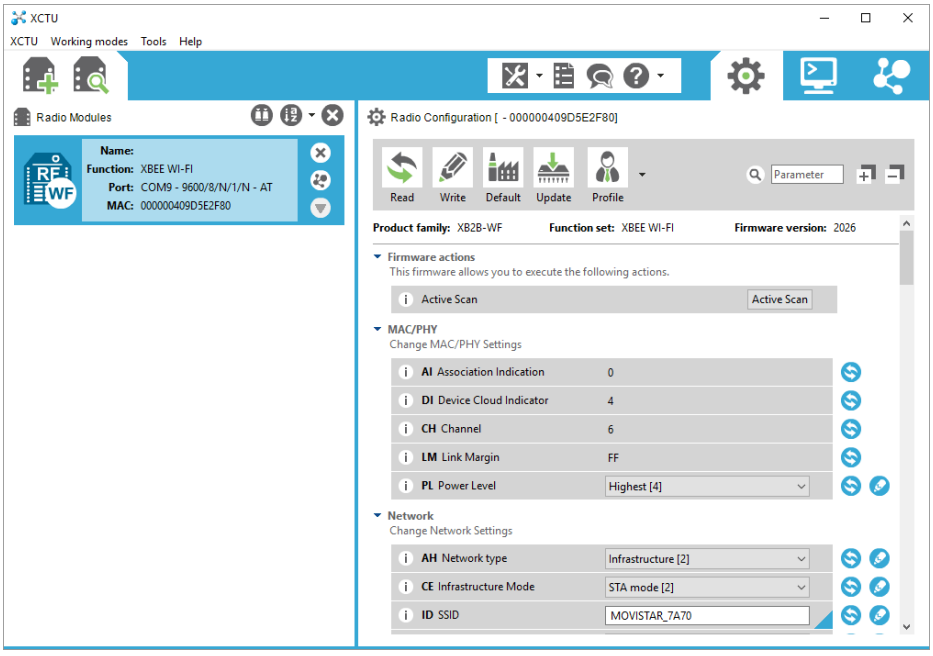

Figura 4.7: Pantalla principal de XCTU amb el xBee del projecte connectat.

Per connectar el transmissor a l'ordinador s'empra un adaptador amb conversor USB-Serial i un cable USB amb connector mini, com es pot veure a la figura 4.6.

En quant a la configuració bàsica del xBee, s'empra el programa XCTU (Figura 4.7) que posa a disposició el seu fabricant, Digi. A la pantalla principal es poden veure els paràmetres actuals del xBee i modificar-los. També es permeten altres accions, com fer un reseteig de fàbrica o actualitzar el firmware del dispositiu si escau.

4.2.2 Configuració

Dins el XCTU es carrega una configuració bàsica per a que el transmissor xBee sigui capaç de connectar amb la xarxa Wi-Fi i realitzar la resta de tasques que li pertoquen correctament: Sleep, enviament a certa direcció IP, etc.

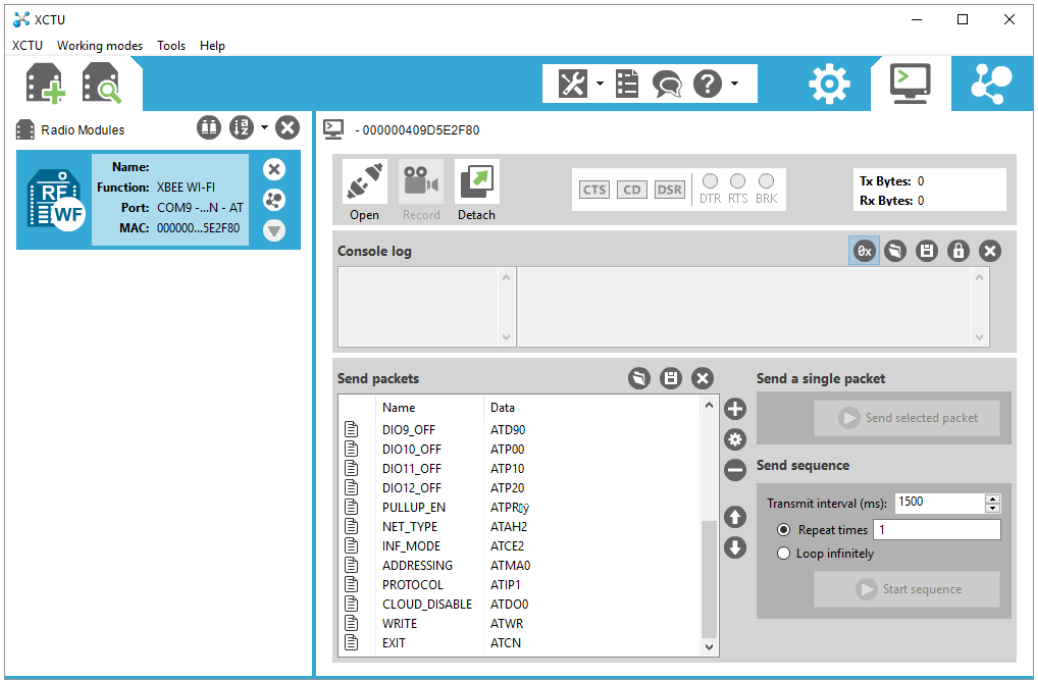

Figura 4.8: Seqüència de comandes preparada per enviar (part inferior).

Des d'una segona pantalla, que fa de terminal, es pot establir comunicació directa amb el transmissor (Figures 4.8 i 4.9). A la figura 4.9 es pot observar que la comunicació comença amb '+++' i acaba amb 'ATCN\r'. Llevat de la comanda d'inici '+++', totes les comandes han d'acabar amb retorn de carro (\r o 0x0D).

També es poden crear seqüències de comandes i enviar-les totes d'una vegada, separades per un cert temps, com es mostra a la figura 4.8. Per a la configuració del xBee del projecte s'han creat dues seqüències:

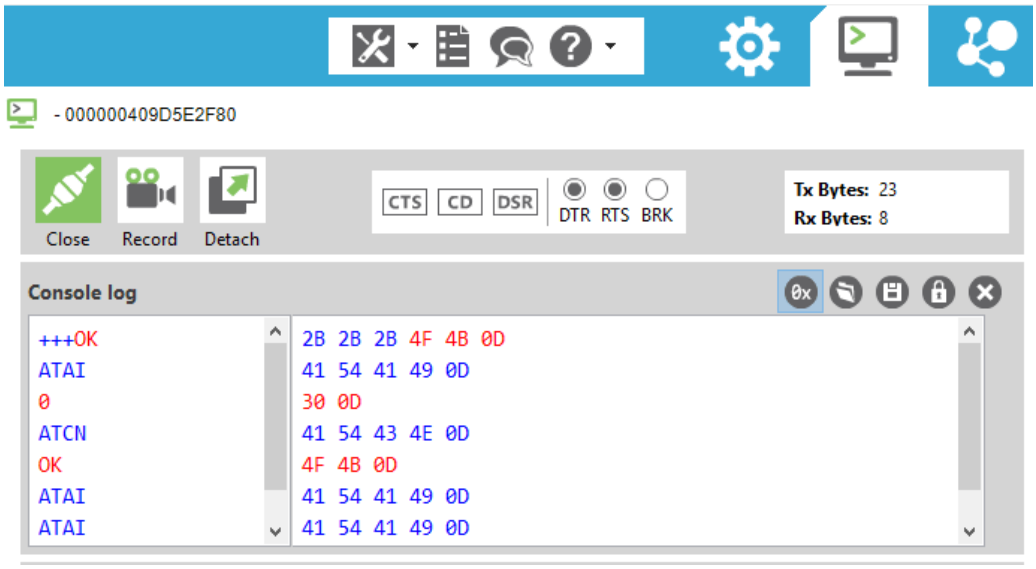

Figura 4.9: Comunicació escrivint comandes manualment.

- La configuració bàsica de qualsevol xBee dins un sistema com el del projecte. Concretament les configuracions que es fan són les mostrades a la taula 4.2.

Acció	**Comanda**
Inicia comandes	+++
UART a 9600 bauds	ATBD3\r
Deshabilita pins DIO0 a DIO12	ATD00\r ... ATD90\r ATP00\r ATP10\r ATP20\r
Habilita tots els pull-ups dels pins	ATPR+(0x7FFF)+\r
Tipus de xarxa és Infraestructura	ATAH2\r
Mode d'infraestructura és STA	ATCE2\r
Direcció IP és amb DHCP	ATMA0\r
Protocol IP és TCP	ATIP1\r
Deshabilita Device Cloud	ATDO0\r
Guarda a la memòria no volàtil	ATWR\r
Finalitza comandes	ATCN\r

Taula 4.2: Paràmetres bàsics per al xBee.

- Configuració dels paràmetres que varien d'una xarxa a altra o entre diferents Sentilos. Es pot veure a la taula 4.3.

Acció	**Comanda**
Inicia comandes	+++
Nom del PA	ATID+[Nom]+\r
Encriptació (x és de 0 a 3)	ATEEx\r
Contrasenya del PA	ATPK+[Contrasenya]+\r
IP de destí	ATDL+[IP del Sentilo]+\r
Port de destí	ATDE+[Port en hexadecimal]+\r
Guarda a la memòria no volàtil	ATWR\r
Finalitza comandes	ATCN\r

Taula 4.3: Paràmetres específics per a cada xarxa i Sentilo.

A més d'aquesta configuració mitjançant seqüències, també s'ha configurat el sleep del transmissor de mode que seguesqui un cicle. En aquest cicle configurat el xBee dorm 13 minuts i 30 segons, seguidament es desperta i cerca xarxa Wi-Fi durant 30 segons.

Tot d'una que troba xarxa comença la fase de funcionament normal, que és d'1 minut, i en acabar la fase normal torna a entrar en sleep. No obstant, si passat el temps de cerca de xarxa no se n'ha trobat, el transmissor entra en fase de dormir directament, botant el mode de funcionament normal.

Aquest sistema que en principi pareix prou pràctic comporta un problema, i és que els cicles oscil·larien entre 14 minuts (13:30 de Sleep + 30 de no trobar xarxa) i 15 minuts (13min30s de Sleep + 30s en trobar xarxa al darrer moment + 1min de funcionament normal). Per evitar això, des del programa del PIC16F886 es força entrar i sortir d'aquest mode cíclic cada vegada que es fa una tramesa de dades. D'aquest mode el cicle no es dessincronitza del microcontrolador, encara que cada cicle del sleep tengui una durada diferent.

Configuració des del microcontrolador

Es mostra el codi que s'aplica abans i després de transmetre les dades a Sentilo. En el cas de no funcionar la comanda per sortir del mode Sleep cíclic es força una neteja d'aquest paràmetre fent un reset al transmissor:

```
1    if(SentiloXBee_sleepEnable(FALSE)){
2        xBee_reset(); //Command has not worked, resetting xBee to clean SM
     parameter
3        SentiloXBee_sleepEnable(FALSE);
4    }
5
6    SentiloXBee_publishObservation(providerId, 4, apiKey);
7    SentiloXBee_sleepEnable(TRUE);
```

4.3 MV Plataforma Sentilo

Per al projecte que s'exposa s'ha emprat la MV que hi ha a disposició a la mateixa pàgina web. Aquesta té limitacions en front a la versió completa per a configurar a un servidor, però és suficient per al que es vol fer: transmetre dades des del node cap a una base de dades. Les configuracions per a aquesta MV són explicades més detalladament a mode de manual a l'Annex C.

4.3.1 Instal·lació de la MV

La virtualització del Sentilo s'ha fet amb el programa VirtualBox, obrint el fitxer descarregat des de la pàgina de Sentilo[14]. Per a la seva configuració bàsica s'ha tengut en compte tota la informació esmentada en dita pàgina. A més a més, s'han anat realitzant back-ups de la MV, a causa de la fragilitat que té; doncs queda inservible si no s'apaga com diu a la plana web.

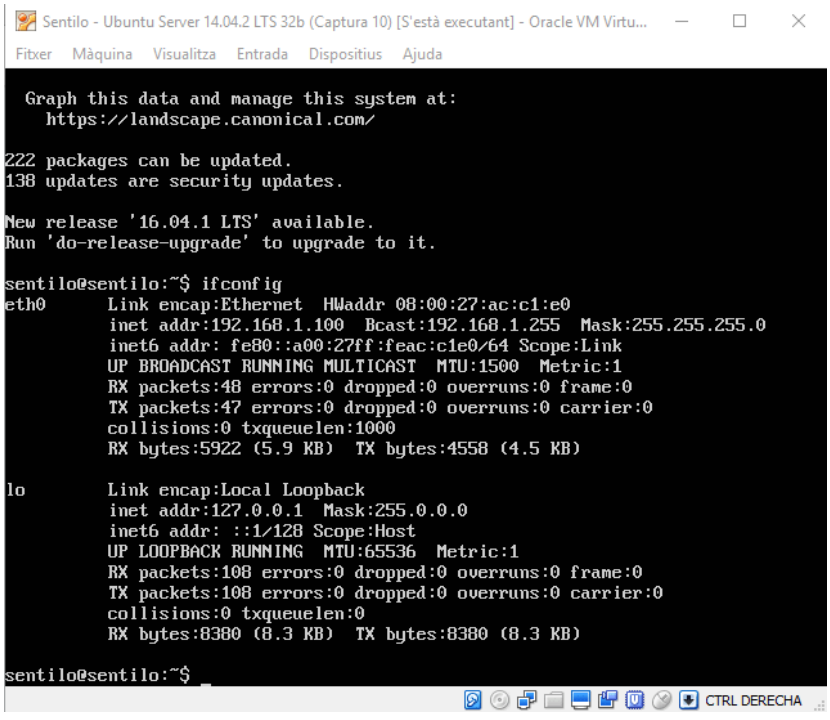

Figura 4.10: Terminal a la MV Sentilo.

La MV s'inicia com un terminal, on s'hi pot demanar la direcció IP que ha pres amb la comanda *ifconfig*. Es veuria com a la figura 4.10. Una vegada coneguda aquesta IP, es pot accedir a la plataforma Sentilo des del navegador i el transmissor xBee. Per a facilitar la feina es va entrar a la configuració DHCP de l'encaminador i es va assignar una IP estàtica a la màquina. D'aquest mode sempre té la mateixa i el xBee no ha de reconfigurar-se cada vegada.

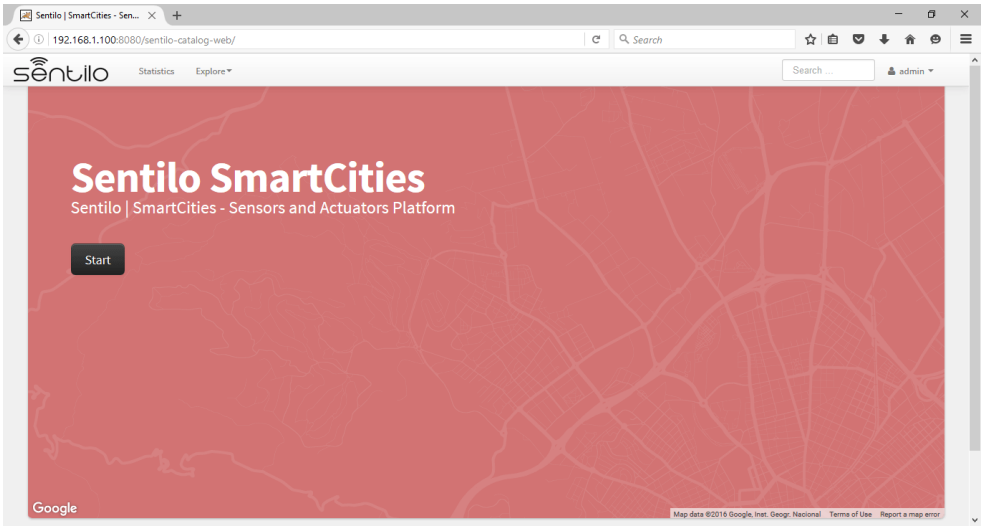

Figura 4.11: Pàgina d'inici de la plataforma.

En haver-se iniciat la MV, s'hi pot accedir pel navegador mitjançant l'IP, especificant el port 8080 i el directori. Per a una IP local seria, per exemple: *http://192.168.1.100:8080/sentilo-catalog-web/*. Un cop oberta aquesta direcció web s'inicia sessió com a *admin* i s'hi poden fer diferents modificacions bàsiques de la plataforma, com per exemple el punt del mapa on s'inicia (per defecte s'obre sobre la ciutat de Barcelona). Es pot veure la pàgina d'inici a la figura 4.11.

4.3.2 Enregistrament de node i sensors

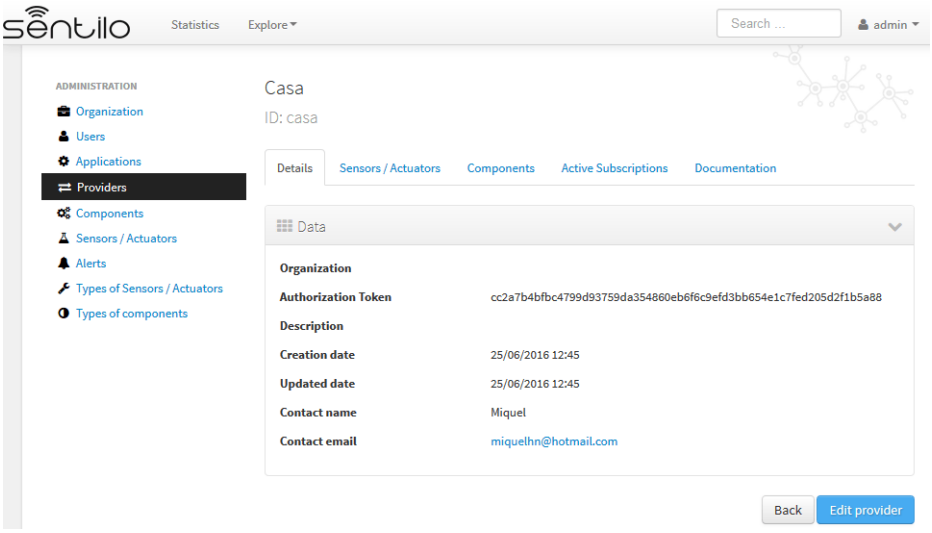

Figura 4.12: Configuració del proveïdor de dades.

Un cop realitzades les configuracions inicials que es vulguin, es passa a crear el necessari per al node. Primerament es va crear un ***Provider***, això és un proveïdor de dades. A la MV del projecte se'n van crear dos: un per al node del projecte i un altre per a proves amb un node d'un altre projecte. En crear-se el proveïdor, la plataforma li assigna un *Authorization Token*, que no és més que l'*apiKey* que s'ha d'emprar des del codi del microcontrolador per a fer els enviaments. Es mostra el proveïdor creat a la figura 4.12.

Seguidament a aquest *Provider* se li crea i assigna un ***Component***, que representaria el node del projecte. La configuració necessària per aquest element és la posició al mapa i l'associació al *Provider*.

La següent passa és la de la creació dels ***sensors***. Per a això es donen d'alta els diferents sensors necessaris a la pestanya de sensors. Allà s'hi selecciona el tipus que és i les unitats que empra entre d'altres paràmetres, i s'associa a un *Provider* i un *Component*. A la figura 4.13 es pot veure el llistat de sensors amb què compta la plataforma.

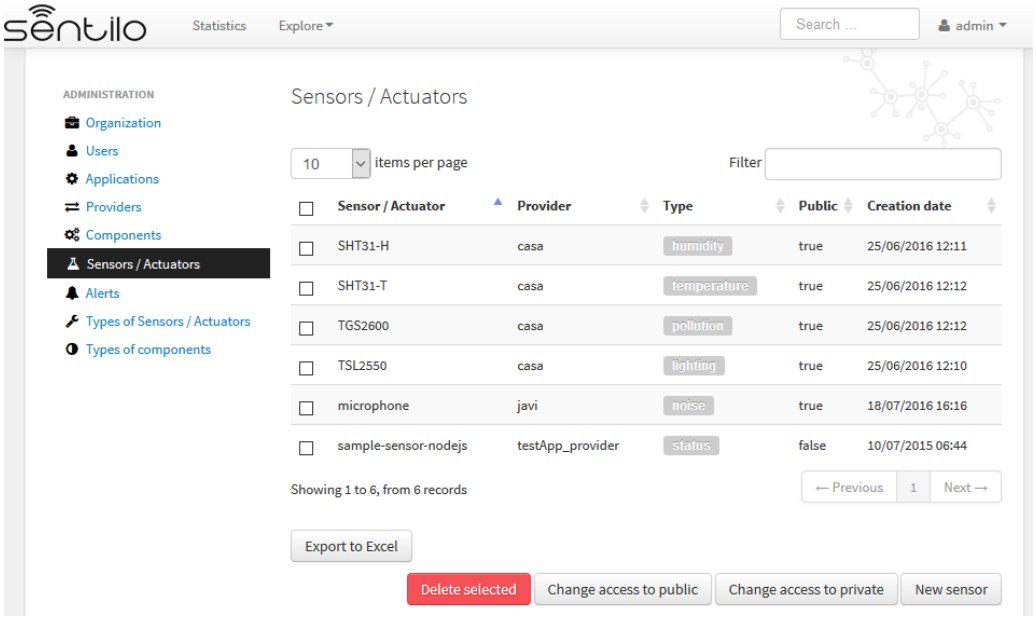

Figura 4.13: Llistat de sensors a la plataforma Sentilo.

Cal notar que el sensor SHT31-T/H mostrat a la figura 4.13 finalment no ha estat aquest, però per tal de poder conservar les dades no s'han creat sensors nous amb nom diferent: S'assumeix que SHT31 representa el SHT71. També es pot veure un sensor micròfon relatiu a un altre projecte i un sensor de prova propi de la MV.

Addicionalment, s'han hagut de crear dos tipus de sensors (*Types of sensor*) a la plataforma, això és degut a que els sensor de contaminació i d'il·luminació no estaven creats per defecte a la màquina. Per a crear-los només cal escriure el nom del tipus i posar-los un identificador de tipus.

4.3.3 Pujada de dades

Per a pujar les dades a la MV és necessari que s'enviï des del transmissor una petició HTTP amb un PUT. Aquesta petició s'ha de fer a l'IP que té la MV, al port 8081; el qual s'ha configurat al transmissor xBee (veure la taula 4.3 de la secció 4.2).

GENERATE CODE SNIPPETS

HTTP

Copy to Clipboard

```
1  PUT /data/casa HTTP/1.1
2  identity_key: cc2a7b4bfbc4799d93759da354860eb6f6c9efd3bb654e1c7fed205d2f1b5a88
3  Host: 192.168.1.100
4  Connection: close
5  Content-Length: 227
6  Content-Type: application/json
7  Cache-Control: no-cache
8  Postman-Token: 52eab1ce-00e0-0e20-64bd-10101e119f37
9
10 {"sensors":[{"sensor":"TSL2550","observations":[{"value":"0"}]},{"sensor":"SHT31-H"
       ,"observations":[{"value":"0.0"}]},{"sensor":"SHT31-T","observations":[{"value"
       :"0.0"}]},{"sensor":"TGS2600","observations":[{"value":"0.0"}]}]}
```

Figura 4.14: Trama de prova enviada des de Postman.

A l'enviament s'ha d'incloure l'IP, l'*apiKey* proporcionada al *Provider* i longitud de trama, entre d'altres paràmetres. Al final també inclou una trama de dades, que pot tenir varis formats com s'explica a l'Annex C. A la figura 4.14 es mostra Postman (una extensió de Chrome que permet enviar diferents peticions HTTP [15]) amb una prova d'enviament de dades.

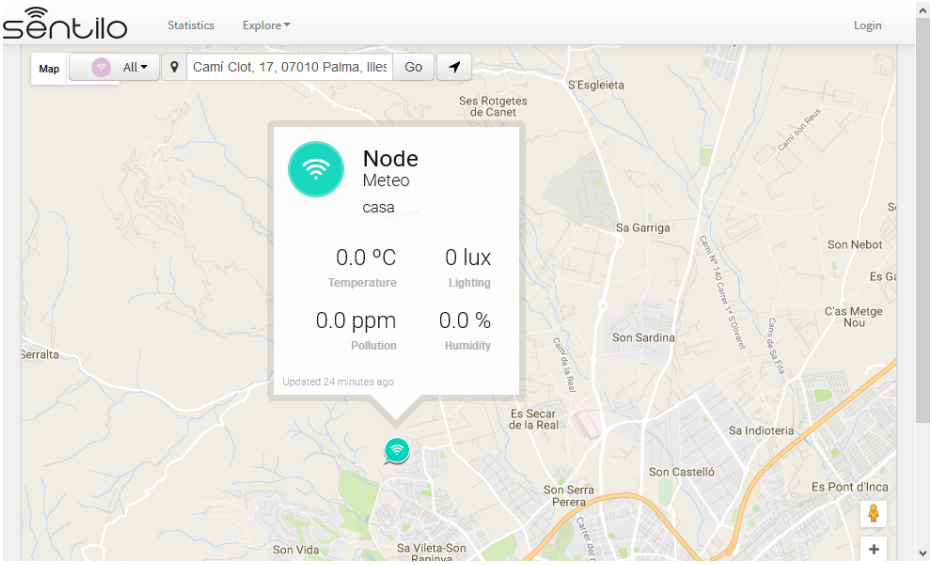

Figura 4.15: Trama de Postman rebuda.

El format de trama que hi apareix és el que s'empra al node, enviant de cop els valors de cada sensor. L'única diferència que hauria entre la trama de la figura 4.14 i la muntada pel programa del microcontrolador són les línies 7 i 8, que són un afegit pel Postman i des del node no s'envien. A la figura 4.15 es pot veure com s'ha rebut aqueixa trama de dades.

4.3.4 Accés a les dades

Per accedir a les dades emmagatzemades a Sentilo hi ha tres formes diferents per fer-ho. La primera forma és clicar al component, com es mostra a la figura 4.15, per veure la darrera lectura.

Una altra manera de veure les lectures és clicant a la pestanya ja oberta de la figura 4.15. En haver-ho fet s'obre una pantalla on es mostren les característiques d'un dels sensors i les seves deu darreres lecures. Es pot veure aquesta pantalla a la figura 4.16. Dins aquesta vista es pot seleccionar la informació d'un altre sensor, però en cap moment es poden veure lectures anteriors a les deu darreres.

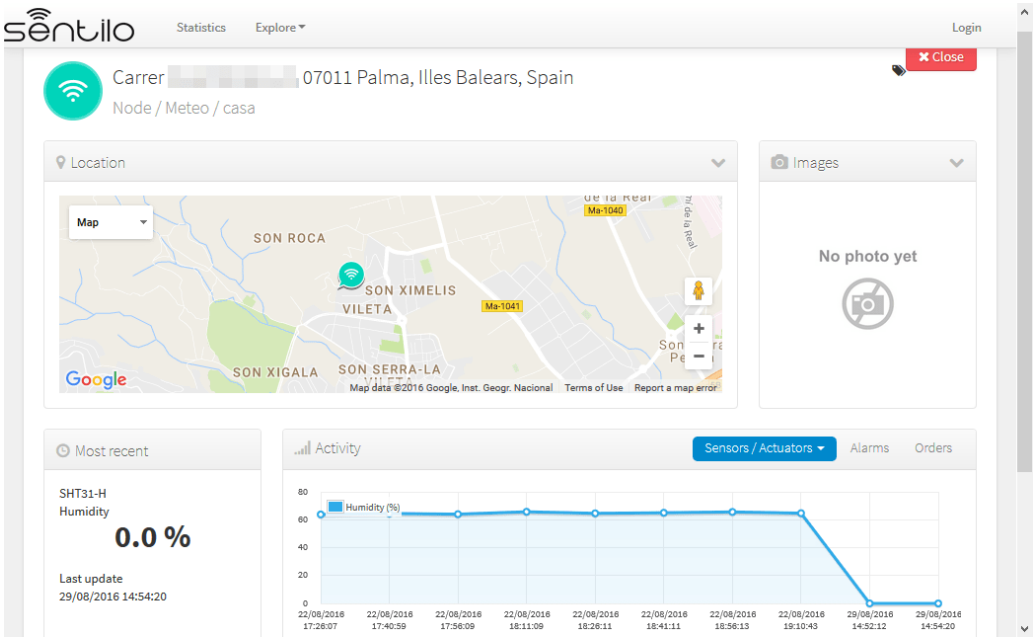

Figura 4.16: Visualització del sensor.

La tercera forma de veure les lectures és més complexa i requereix el Postman o una API específica per extreure'n les dades. S'ha d'enviar una petició HTTP amb un GET, especificant el rang de temps del qual se'n volen dades i el nombre màxim de dades que es volen. Per aquesta acció també es requereix l'*ApiKey* del proveïdor. Tot això s'explica més extensament a la secció corresponent a la web de Sentilo.[16]

A la figura 4.17 es veu la resposta que proporciona el Sentilo en fer aquesta petició de les dades.

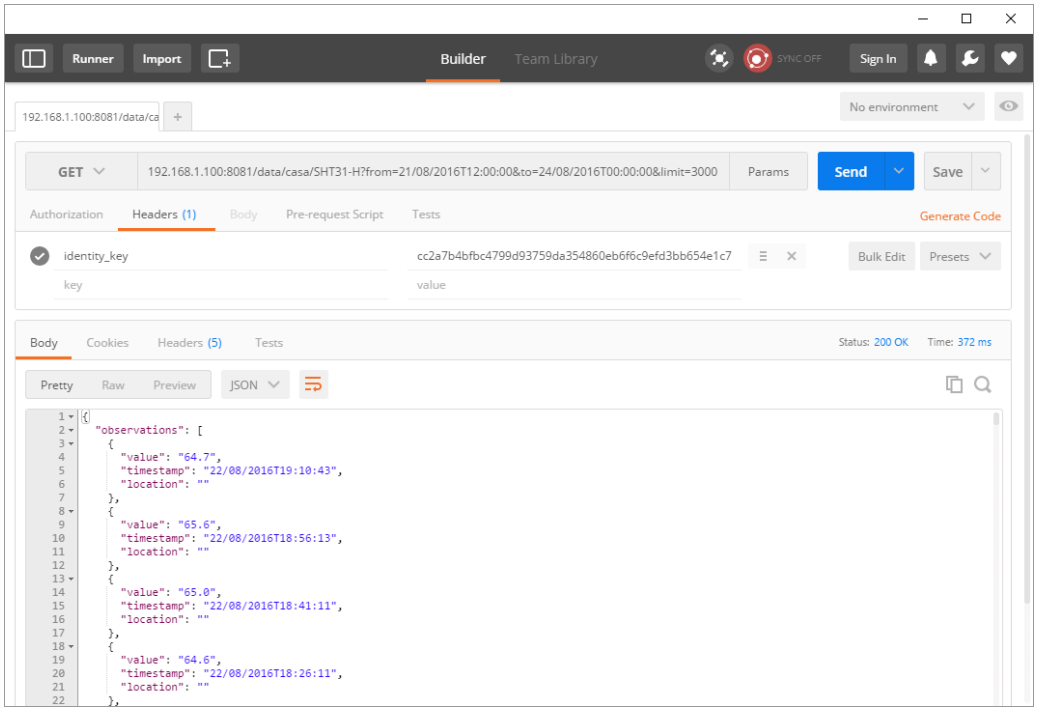

Figura 4.17: Petició de dades mitjançant Postman.

CAPÍTOL 5

RESULTATS OBTENGUTS

5.1 Funcionament general

Per a provar el comportament del node aquest s'ha tengut en funcionament durant 3 dies i 6 hores. Durant aqueix temps s'ha pogut comprovar que el programa no es queda penjat, encara que no pugui enviar alguna trama.

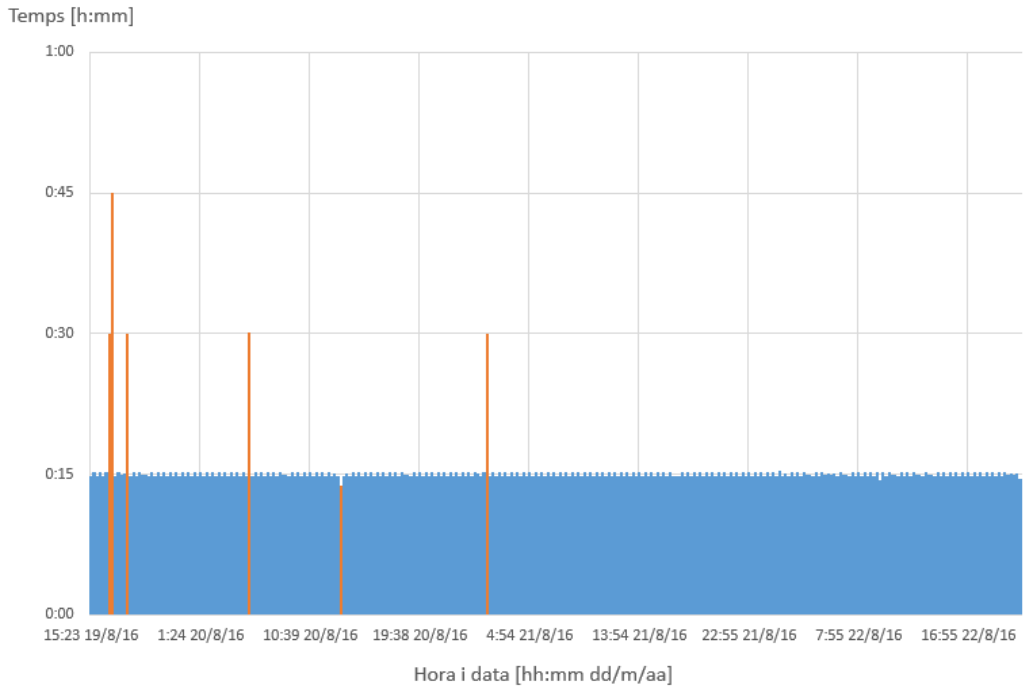

Figura 5.1: Gràfic de l'interval entre lectures.

A la figura 5.1 es mostra el temps entre les lectures rebudes per la Plataforma Sentilo. Són marcats en color vermell els intervals més llargs o curts d'un minut respecte als 15 minuts de cicle. Cal esmentar que al començament de la prova l'ordinador on s'estava executant la MV era amb connexió Wi-Fi, i és probable que algunes de les lectures que no arribaren deixassin de fer-ho per mor de la connexió de l'ordinador.

A més a més, es produeix un desfasament progressiu de les dades respecte al temps real, comptant des de l'inici. Per exemple, l'última dada es va rebre a les 78h 02min 11s després de la primera lectura, quan tocava haver arribat a les 78h exactes. Aquest fet es mostra a la figura 5.2, on els valors negatius simbolitzen que el valor s'ha rebut abans de l'hora prevista i els positius que s'ha rebut més tard de l'hora.

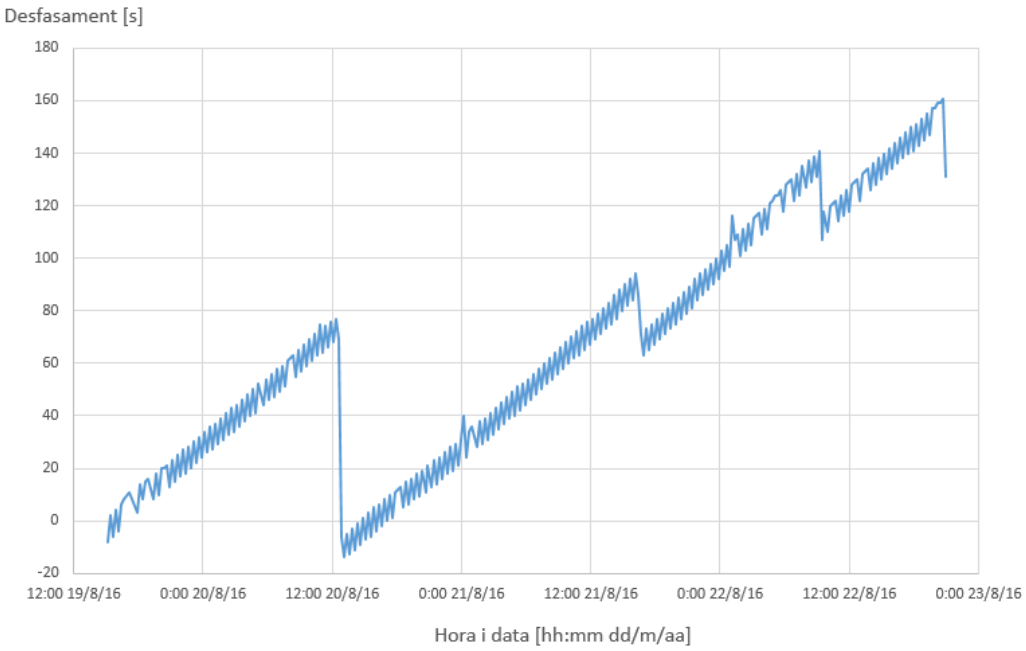

Figura 5.2: Gràfic del desfasament de les lectures.

Un motiu d'aquest efecte podria ser que fos degut a que la MV del Sentilo es desincronitzàs de l'ordinador. Un altre possible motiu seria que els valors del timer del microcontrolador no haguessin estat calculats amb exactitud i, per tant, els cicles no fossin exactes.

5.2 Valors dels sensors

En aquesta secció s'exposen les diferents lectures dels sensors durant la prova de 3 dies i 6 hores. Per a mostrar com responen a les condicions ambientals s'empren gràfiques de l'històric de dades, d'aquest mode és molt més visual que amb valors numèrics.

5.2.1 SHT71 - Humitat i temperatura

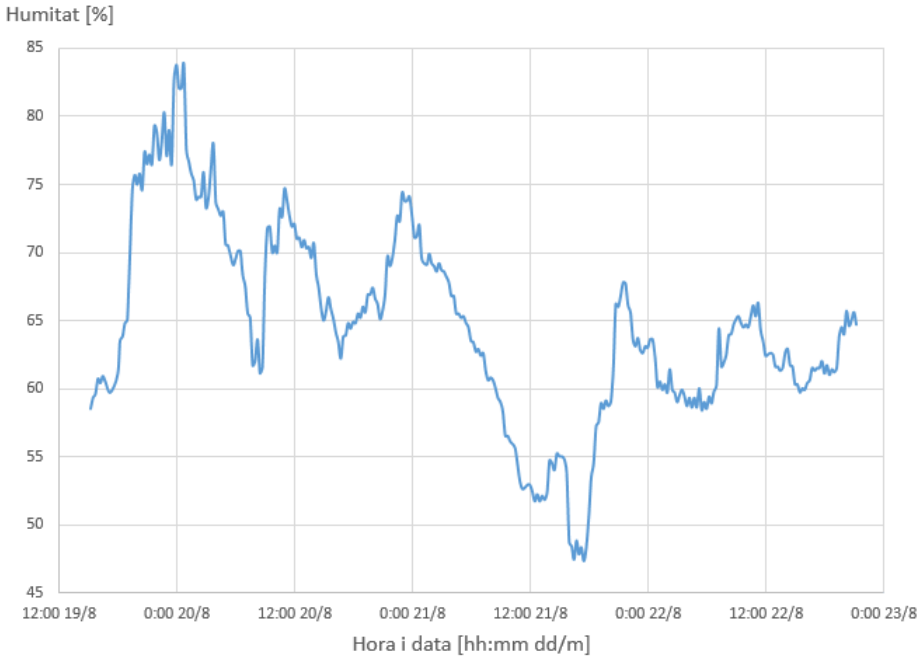

Figura 5.3: Resultats d'humitat relativa obtenguts.

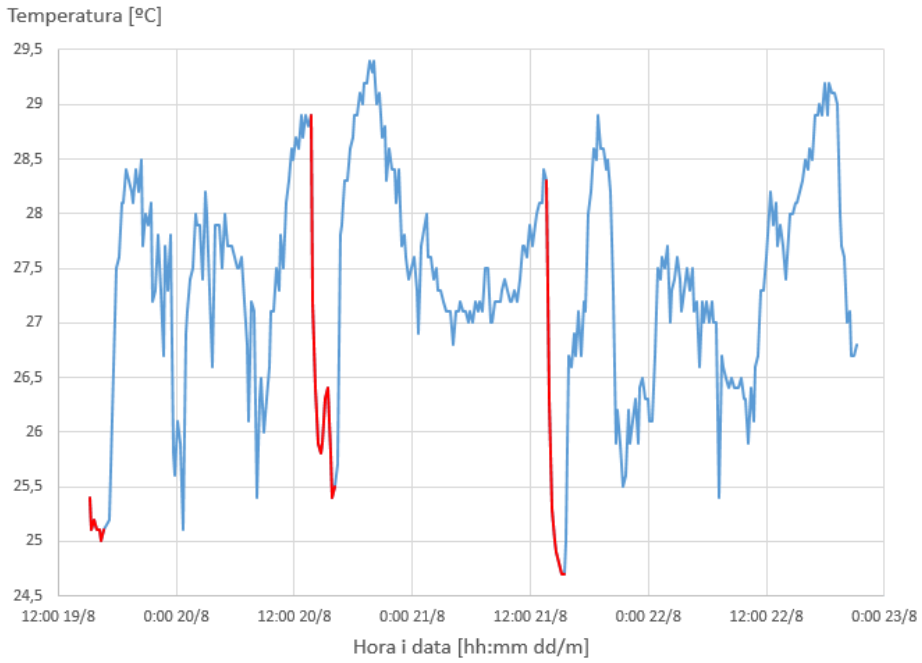

Figura 5.4: Resultats de temperatura obtenguts.

A la figura 5.3 es mostra la resposta d'humitat del sensor durant els 3 dies. Es pot notar que no s'arriba a estabilitzar del tot ni tan sols quan sembla estar a una humitat fixa. De fet, el valor d'humitat enviat al Sentilo és el promig de 6 lectures però, com la precisió típica segons datasheet és de ±3%, podem considerar que aquestes oscil·lacions són normals.

Per altra banda, a la figura 5.4 es poden observar les mesures de temperatura preses durant els 3 dies. Addicionalment es pot veure marcat amb color vermell el temps de funcionament de l'aire condicionat en l'habitació on es trobava el node.

A l'igual que amb el sensor d'humitat, la temperatura tampoc no arriba a estabilitzar-se en cap moment, malgrat el promitjat de 6 lectures. Això és degut als canvis continus entre corrents i aire condicionat. De totes formes, les oscil·lacions dels valors no són grans si, per exemple, es mira la matinada del dia 21. En aqueix moment, durant hores els valors es mouen entre 27°C i 27.5°C, quan la precisió típica segons datasheet és de ±0.4°C.

5.2.2 TSL2550 - Il·luminació

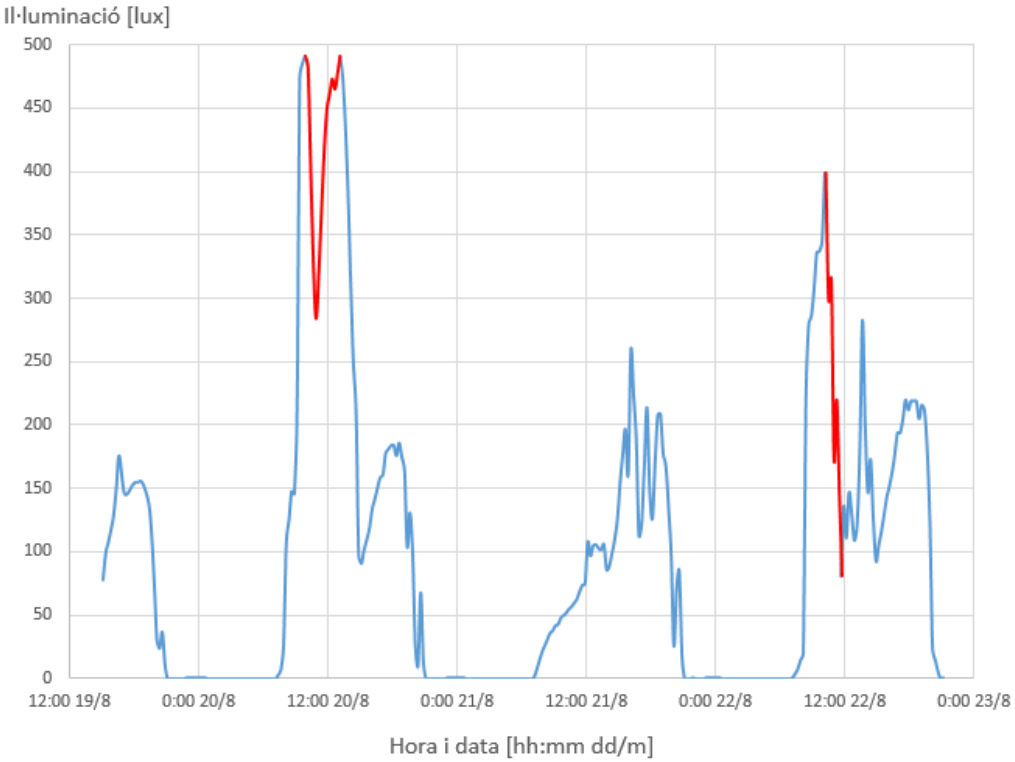

Figura 5.5: Resultats d'il·luminació obtenguts.

La major limitació d'aquest sensor és que entra en saturació amb quantitats altes de llum, com podria ser la llum solar directa. Això és explicat i provat amb l'equació 3.2

a l'apartat 3.4, al capítol de Hardware.

Al gràfic de la figura 5.5 es pot veure l'evolució de la il·luminació al llarg dels tres dies. Es veu que durant les nits el sensor dóna 0, ja que no hi ha llum dins la sala. Durant els matins dels dies 20 i 22, que varen ser dies de sol, es pot veure que el sensor va escalant valors. Tanmateix, arribat a un punt els valors tornen baixar, el qual és per mor de la saturació que pateix el sensor (zones marcades amb color vermell). Pels horabaixes el nivell d'il·luminació és més baix, degut a que les finestres es tanquen per evitar llum directa.

Per altra banda, es pot veure que el dia 21 el comportament és diferent. Això es deu a que el dia era ennigulat i la il·luminació era menor.

5.2.3 TGS2600 - Contaminació

Per a les proves del node, a més de la prova llarga de la que s'han estat mostrant els resultats, es va realitzar també una prova de 12 hores. Aquesta prova es va descartar parcialment per ser massa curta, però serveix per mostrar alguns efectes sobre aquest sensor.

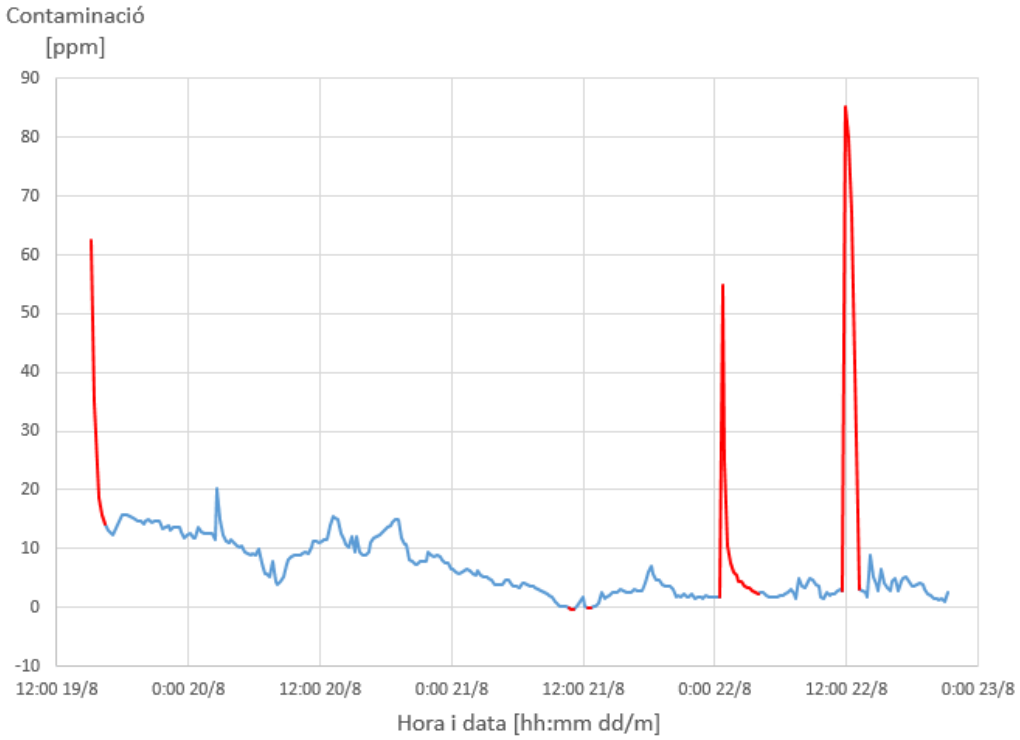

Figura 5.6: Resultats de contaminació obtenguts.

Es pot observar a la figura 5.6 la **prova de 3 dies** per aquest sensor. En iniciar el node el sensor de contaminació necessita un temps d'assentament per començar a donar

valors vàlids. Segons datasheet el temps ideal d'escalfament del sensor ha de ser de 7 dies. No obstant, i degut al baix consum que ha de tenir el dispositiu, aquest només roman encès 2 minuts seguits dins cada cicle de 15 minuts.

Es mostren en color vermell alguns efectes d'aquest sensor:

- Al començament la baixada fins a escalfar-se i estabilitzar-se.
- Pel mig alguns valors que sobrepassen el 0 per baix: El sensor mai queda ben ajustat per a que 0 sigui el mínim.
- El primer pic, amb valors fins 55ppm, va ser produït accidentalment en haver-se flitat un insecticida per la casa. El valor es va esmorteint durant la matinada.
- El segon pic, amb valors fins a 85ppm, va ser produït en moure el node a una habitació petita i provar la contaminació generada per un desodorant d'aerosol (només dues ruixades). El valor s'esmorteix lentament, però després de cinc valors es torna a col·locar al lloc anterior, deixant així de captar el desodorant.

Per altra banda, a la **prova curta de 12 hores**, es va realitzar una prova de la resposta al fum del tabac que es pot observar en el pic que es forma sobre les 21 hores. Es mostra a la figura 5.7 juntament amb la humitat relativa, ja que es creu que la baixa humitat va jugar un paper important en les lectures tan baixes (fins a -5.6 ppm) proporcionades pel sensor.

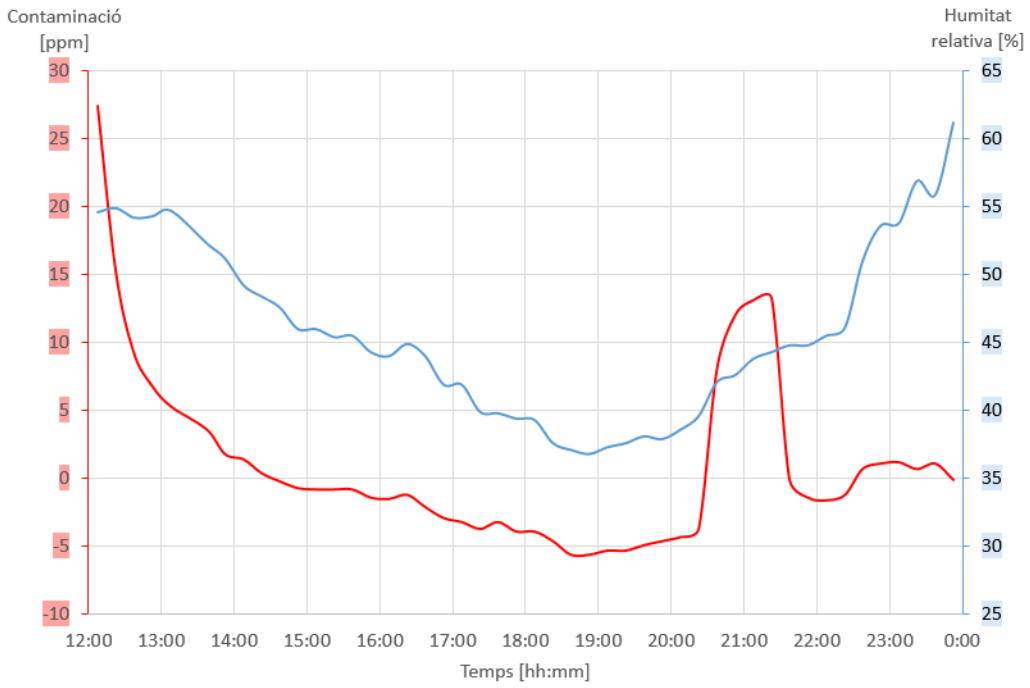

Figura 5.7: Resultats de contaminació i humitat relativa obtenguts a la prova curta.

Segons el datasheet i com es pot comprovar a la figura 5.7, aquest sensor és molt susceptible a canvis en humitat i temperatura. Conseqüentment, no és recomanable en entorns on aquestes condicions ambientals siguin variables.

Finalment, a la figura 5.8 es fa una comparativa de la prova de fum de tabac de la prova curta amb la prova d'aerosol de desodorant de la prova llarga. Cal notar que les dues proves es realitzaren a la mateixa habitació, i que a la prova del desodorant es va tenir la finestra oberta mentre que a la prova del tabac no. Tot i que el pic del tabac partia de valors més baixos es pot veure que el sensor és molt més sensible al desodorant.

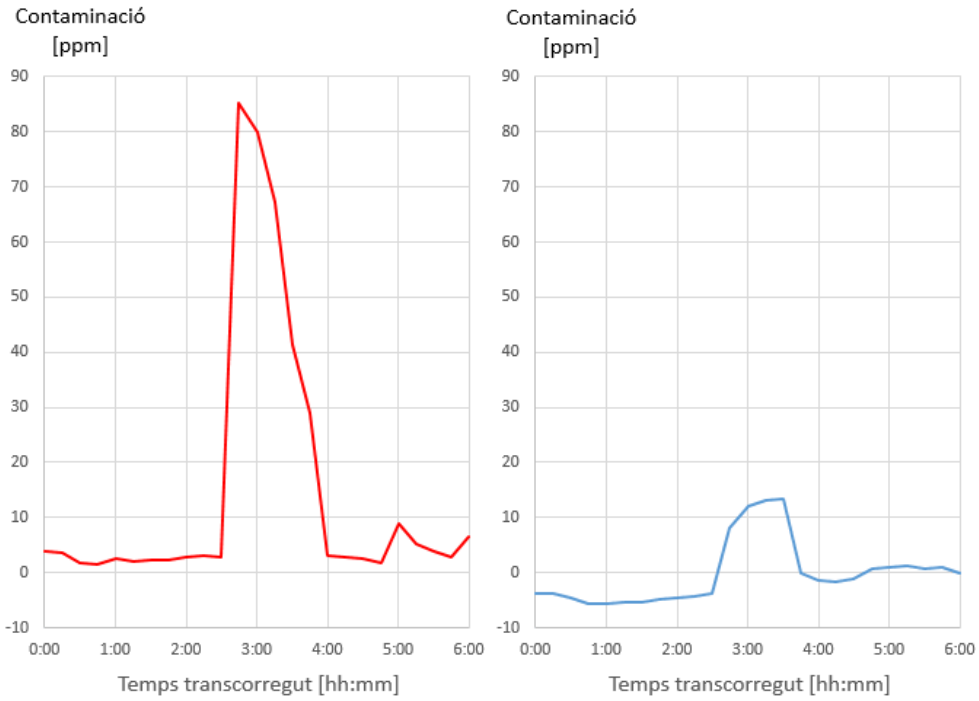

Figura 5.8: Comparativa prova tabac (en blau) i prova desodorant (en vermell).

CAPÍTOL 6

CONCLUSIONS

Amb aquest treball es pretenia el disseny i fabricació d'un node de sensors emmarcat en el camp de l'IoT. Aquest node havia de ser capaç de prendre dades del seu entorn i enviar-les a una base de dades per tenir-ne un registre. Després de la feina realitzada, podem concloure que s'han aconseguit tots aquests objectius satisfactòriament, amb més o menys entrebancs.

6.1 Aplicacions

A partir d'aquest treball s'ha comprovat la possibilitat de crear un node IoT sense necessitat de partir de models ja acabats. Existeix la possibilitat d'emprar material asequible, com seria per exemple el microcontrolador, i poder fer d'aquesta branca de les TIC quelcom més accessible.

6.2 Futur desenvolupament

Per a futurs desenvolupaments seria important fer millores sobre el disseny del node en alguns aspectes.

Millores en components

- Sobre el disseny inicial el sensor de temperatura-humitat era el SHT31, però per qüestions alienes s'ha hagut d'emprar el SHT71. Aquest sensor SHT71 ha desmostrat tenir un bon funcionament, malgrat no funcionar amb protocol I^2C, emprant un codi senzill proporcionat per Sensirion. És per això que seria raonable plantejar un canvi permanent a aquest sensor i realitzar el seu respectiu canvi en el disseny de la PCB.

- El microcontrolador emprat és molt limitat en temes de memòria, fet que ha propiciat la no inclusió del RTC al node, entre d'altres limitacions. Seria interessant passar a un PIC amb major capacitat de memòria o comprar una llicència de compilador que optimitzàs més l'espai.
- Substituir el programador/debugger del microcontrolador per un de més nou, ja que aquest ha quedat obsolet i requereix emprar programari antic.
- Incloure un oscil·lador extern al microcontrolador. Sense ell el PIC no pot entrar en sleep i per tant sempre es té un consum d'energia més elevat.
- Plantejar si els sensors TSL2550 (il·luminació) i TGS2600 (contaminació) són adequats per a l'aplicació, doncs aquests tenen un bon funcionament en espais tancats i ambient poc variable.

Millores sobre el disseny de placa

- Connectar el pin DIO8 del xBee a algun pin digital del microcontrolador. Això ens permetria tenir un control total i senzill del sleep d'aquest transmissor.
- Redissenyar PCB o situar a un altre lloc la resistència pull-up de dades del bus I^2C, ja que no queda ben connectada a 3.3V degut al enrutat de les pistes.

Millores en software

- Modificar bucle switch-case, ja que si MinuteCounter cambiàs eventualment a un valor superior a 15 el funcionament sortiria del cicle de 15 minuts i produiria resultats indesitjables.
- Si es disposa de més espai de memòria, millorar les llibreries de rellotge del sistema i xBee/Sentilo.

APÈNDIX A

CODI DE PROGRAMA PIC16F886

A.1 Header

```
/**
 * @file     xc8_header.h
 * @author   Miquel Hernández Nicolau
 * @version v1.0
 * \~english
 * @date
 * @brief    Header file with macros and basic configuration.
 * @details This header includes several macros and a basic configuration for
 *           the PIC16F886.
 * \~catalan
 * @date
 * @brief    Arxiu header amb macros i configuració.
 * @details Aquest header inclou varis macros i una configuració bàsica per al
 *           PIC16F886.
 */

// PIC16F886 Configuration Bit Settings

// 'C' source line config statements
#ifndef XC8_HEADER_H
#define XC8_HEADER_H
#include <htc.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "pic16f886.h"

typedef signed char         int8_t;
typedef unsigned char       uint8_t;
typedef signed int          int16_t;
typedef unsigned int        uint16_t;
typedef signed long int     int32_t;
```

```
33 typedef unsigned long int    uint32_t;
34 typedef float               real32_t;
35
36 #define TRUE          (uint8_t)1
37 #define FALSE         (uint8_t)0
38 #define _XTAL_FREQ    8000000
39 #define BAUD_RATE     (uint16_t)9600
40 #define I2C_FREQ      50000
41
42
43 #define LIGHTING      0
44 #define TEMPERATURE   1
45 #define HUMIDITY      2
46 #define POLLUTION     3
47
48
49 /** @defgroup System_Frequencies
50   * @{
51   */
52 #define System_Clock_8MHz    ((uint8_t)0x75)
53 #define System_Clock_4MHz    ((uint8_t)0x65)
54 #define System_Clock_2MHz    ((uint8_t)0x55)
55 #define System_Clock_1MHz    ((uint8_t)0x45)
56 #define System_Clock_500kHz  ((uint8_t)0x35)
57 #define System_Clock_250kHz  ((uint8_t)0x25)
58 #define System_Clock_125kHz  ((uint8_t)0x15)
59 #define System_Clock_31kHz   ((uint8_t)0x03)
60 /**
61   * @}
62   */
63
64
65 #endif /* XC8_HEADER_H */
```

A.2 Main

```
1  /**
2   * @file     main.c
3   * @author   Miquel Hernàndez Nicolau
4   * @version  v1.0
5   * \~english
6   * @date     20th January 2016
7   * @brief    Main program to control sensors information, RTC and Wi-Fi module.
8   * @details  This file is the main program to control the sensors, the information
9   *           they provide and the transmission of it through xBee. Furthermore,
10  *           it intercommunicates all these components.
11  * \~catalan
12  * @date     20 de gener de 2016
13  * @brief    Programa principal que controla la informació dels sensors, el RTC
14  *           i el mòdul Wi-Fi.
15  * @details  Aquest arxiu és el programa principal per controlar els sensors, la
16  *           informació que aquests proporcionen i la transmissió amb xBee. A més
17  *           a més, intercomunica tots aquests components entre ells.
18  */
19
```

```
20
21 #define SENS_CONT       //Comentar si no s'empra el sensor de contaminació
22
23 /****************************** INCLUDES *********************************/
24 #include "xc8_header.h"
25 #include "clock.h"
26 #include "uart.h"
27 #include "xbee.h"
28 #include "i2c.h"
29 #include "SHT71.h"      //Comentar si no s'empra el sensor de T i HR
30 #include "TSL2550D.h"   //Comentar si no s'empra el sensor de llum
31
32
33 //Per a carregar el programa s'ha de canviar a MCLRE_ON i DEBUG_OFF
34 __CONFIG(FOSC_INTRC_CLKOUT & WDTE_OFF & PWRTE_OFF & MCLRE_ON & CP_OFF & CPD_OFF &
       BOREN_ON & IESO_OFF & FCMEN_OFF & LVP_OFF & DEBUG_OFF);
35 __CONFIG(BOR4V_BOR21V & WRT_OFF);
36
37 /***************************** CONFIGURATIONS ****************************/
38 void            Setup(void);
39 void            IOConfig(void);
40 void            xBee_reset(void);
41 /*********************** INTERRUPT SERVICE ROUTINE ***********************/
42 void interrupt  ISR(void);
43 /******************************* UTILITIES *******************************/
44 real32_t        AddToAverage(real32_t oldAverage, real32_t newValue, uint8_t
       numberOfValues);
45 uint8_t         *ftoaEspecial(real32_t f);
46
47
48 /***************************** GLOBAL VARIABLES **************************/
49 uint16_t        counter = 0;
50 uint8_t         entra = 1;
51 const uint8_t   *providerId = "casa";
52 const uint8_t   *apiKey = "
       cc2a7b4bfbc4799d93759da354860eb6f6c9efd3bb654e1c7fed205d2f1b5a88";
53
54
55
56
57 int main(int argc, char** argv)
58 {
59     //MinuteCounter inicialitzat per a començar amb una mostra de cada sensor i
       un enviament
60     //sensorValue: 0: Lighting, 1: Humidity, 2: Temperature, 3: Pollution
61     uint8_t     aux = 1, MinuteCounter = 10, *help;
62     real32_t    temp1, temp2, sensorValue[4] = {0.0,0.0,0.0,0.0};
63
64
65 ////////////////////////INITIAL CONFIGURATION//////////////////////////////////
66     Setup();
67
68 ////////////////////////////INFINITE LOOP//////////////////////////////////////
69     while(TRUE)
70     {
71         if(entra)
72         {
```

```
73              entra = 0;
74              MinuteCounter++;
75              switch (MinuteCounter)
76              {
77                  #ifdef SENS_CONT
78                  case 12:
79                      RB2 = 1;          //Enable 5v-regulator
80                      break;
81
82                  case 14:
83                      ADON    = 1;      //ADC enable
84                      __delay_us(10); //10us should be enough
85                      GO_DONE = 1;      //Start conversion
86                      while(GO_DONE); //Polling to conversion
87                      RB2     = 0;      //Disable 5v-regulator
88                      sensorValue[POLLUTION] = (ADRESH << 8) + ADRESL;
89                      ADON    = 0;      //ADC disable
90                      //Es fa zero al mínim i es proporciona per fer coincidir el
    màxim amb 100 ppm
91                      sensorValue[POLLUTION] = (sensorValue[POLLUTION] - 310)/7.13;
92                      break;
93                  #endif
94
95                  case 15:
96                      for(uint8_t i = 0; i < 4; i++)
97                      {
98                          if(i)
99                          {
100                             help = ftoaEspecial(sensorValue[i]);
101                             sprintf(SentiloXBee_sensVal[i], "%s", help);
102                         }
103                         else    //Only for lighting sensor
104                             sprintf(SentiloXBee_sensVal[i], "%u", (uint16_t)
    sensorValue[i]);
105                     }
106
107                     //Due to synchronisation sleep must be disabled here and
    renabled after operating
108                     if(SentiloXBee_sleepEnable(FALSE))
109                     {
110                         xBee_reset();   //Command has not worked, resetting xBee
    to clean SM parameter
111                         SentiloXBee_sleepEnable(FALSE);
112                     }
113
114                     SentiloXBee_publishObservation(providerId, 4, apiKey);
115
116                     SentiloXBee_sleepEnable(TRUE);
117
118                     //Reset variables
119                     aux             = 1;
120                     MinuteCounter   = 0;
121                     break;
122
123                 default: //Average values
124                     if((MinuteCounter % 2)&&(MinuteCounter != 13))
125                     {
```

```
126                         #ifdef TSL2550D_H
127                         LightSensor_Command(LightSensor_PowerUp);
128                         sensorValue[LIGHTING] = AddToAverage(sensorValue[LIGHTING
    ], LightSensor_Read(), aux);
129                         LightSensor_Command(LightSensor_PowerDown);
130                         #endif
131                     }
132                     else
133                     {
134                         #ifdef SHT71_H
135                         SSPEN = 0;       //Disable I2C
136                         TRISC &= 0xE7;   //RC3&RC4 are outputs
137 
138                         //Read raw values
139                         temp1 = (real32_t)SHT71_Read(SHT71_READ_H);
140                         temp2 = (real32_t)SHT71_Read(SHT71_READ_T);
141 
142                         //Adjust these values
143                         temp2 = temp2 * 0.01 - 39.7;     //Adjust T
144                         temp1 = temp2 * 0.01 + temp1 * (0.0347 - 0.0000015955 *
    temp1 + temp2 * 0.00008) - 2.2968; //Adjust RH
145 
146                         sensorValue[HUMIDITY]     = AddToAverage(sensorValue[
    HUMIDITY]    , temp1, aux);
147                         sensorValue[TEMPERATURE] = AddToAverage(sensorValue[
    TEMPERATURE], temp2, aux);
148 
149                         TRISC    |= 0x18;     //RC3&RC4 are inputs
150                         SSPEN    = 1;         //Enable I2C
151                         #endif
152                         aux++;
153                     }
154                 }
155             }
156             NOP();  // El NOP consumeix menys que un GOTO
157         }
158 }
159 
160 
161 
162 /****************************** CONFIGURATIONS ******************************/
163 /**
164  * \~english
165  * @brief   It sets the initial configuration of PIC and peripherals.
166  * @retval  None.
167  * \~catalan
168  * @brief   Estableix la configuració inicial per a PIC i perifèrics.
169  * @retval  Cap.
170  */
171 void Setup(void)
172 {
173 ////////////////////////PIC INITIAL CONFIGURATION////////////////////////////////
174     OSCCON = 0x75;          //System clock: 8MHz
175 
176     IOConfig();             //PIC pins configuration
177 
178     if(I2C_Init(I2C_FREQ))  //Set frequency in Hz. Between 10k and 100kHz
```

```
179         while(1);           //I2C configuration failed. Please, check system's
        code.
180
181     if(UART_Init(BAUD_RATE))
182         while(1);           //UART configuration failed. Please, check system's
        code.
183
184     __delay_ms(100);        //Time for UART to stabilize
185
186 ////////////////////// EXTERNAL PERIPHERALS CONFIGURATION //////////////////////
187     #ifdef TSL2550D_H
188     LightSensor_Command(LightSensor_Reset);
189     LightSensor_Command(LightSensor_PowerDown);
190     #endif
191
192     xBee_reset();
193     SentiloXBee_sleepEnable(TRUE);
194
195 ///////////////////////////// TIMER1 CONFIGURATION ////////////////////////////////
196     T1CON   = 0x34; //T1SYNC = 1, TMR1CS = 0 (Fosc/4 selected), T1CKPS = 11 (
        Prescaler is divide by 8)
197     T0CS    = 0;    //Prescaler gets clock from Fosc (2MHz)
198
199 ////////////////////////// INTERRUPTS CONFIGURATION ////////////////////////////
200     PIE1    |= 0x01; //TMR1IE=1 (TIMER1 int.)
201     INTCON |= 0xC0; //GIE=1, PEIE=1 (Globally INTs and peripheral interrupts)
202
203     TMR1GE  = 0;    //Timer on without conditions
204     TMR1ON  = 1;    //Now start the timer!
205 }
206
207
208 /**
209  * \~english
210  * @brief   It configures the input and output ports of PIC.
211  * @retval  None.
212  * \~catalan
213  * @brief   Configura els ports d'entrada i sortida del PIC.
214  * @retval  Cap.
215  */
216 void IOConfig(void)
217 {
218     //Clear ports
219     PORTA = 0;
220     PORTB = 0;
221     PORTC = 0;
222     PORTE = 0;
223
224     //Digital outputs configuration:
225     TRISA3 = 0;             //RA3 as an output.
226     TRISB2 = 0;             //RB2 as an output.
227     TRISC  = 0b11011011;    //RC5 & RC2 as outputs.
228
229     //Analogic input configuration:
230     ANSEL  = 0;             //All pins digital.
231     ANSELH = 0x08;          //AN11(RB4) as analogic input.
232     ADCON0 = 0b10101100;    //Fosc/32 & AN11 selected. ADC disabled.
```

```
233     ADCON1 = 0x80;          //Right justified. Vdd&Vss as references.
234 }
235
236
237 /**
238  * \~english
239  * @brief   Pulls down RC5 pin for one second, which produces a reset on xBee
        module.
240  * @retval  None.
241  * \~catalan
242  * @brief   Posa a zero el pin RC5 durant un segon, el qual produeix un reset al
        mòdul xBee.
243  * @retval  Cap.
244  */
245 void xBee_reset(void)
246 {
247     RC5 = 0;
248     __delay_ms(1000);
249     RC5 = 1;
250     __delay_ms(2);
251 }
252
253
254
255 /************************ INTERRUPT SERVICE ROUTINE ************************/
256 /**
257  * \~english
258  * @brief   Interrupt service routine.
259  * @warning Only implemented for Timer1, not for UART yet.
260  * @see     http://extremeelectronics.co.in/microchip-pic-tutorials/introduction-
        to-pic-interrupts-and-their-handling-in-c/
261  * @retval  None.
262  * \~catalan
263  * @brief   Rutina de servei d'interrupcions.
264  * @warning Només implementat pel Timer1, no per l'UART encara.
265  * @see     http://extremeelectronics.co.in/microchip-pic-tutorials/introduction-
        to-pic-interrupts-and-their-handling-in-c/
266  * @retval  Cap.
267  */
268 void interrupt ISR(void)
269 {
270     if(TMR1IE && TMR1IF)    //Checks whether it was due to TMR1 interrupt
271     {
272         counter++;
273
274         if(counter >= 229)
275         {
276             entra   = 1;
277             //Inserts 7744 to register. (65536-57792 = 7744)
278             TMR1H   = 0x1E;
279             TMR1L   = 0x44; //TMR1L   = 0x40; i afegits 4 cicles per ajustar
280             //Counter starts over
281             counter = 0;
282         }
283         TMR1IF = 0;
284     }
285 }
```

```
286
287
288
289 /******************************** UTILITIES *********************************/
290 /**
291  * \~english
292  * @brief   It returns the new arithmetic average with the new value included.
293  * @param   oldAverage: Previous average.
294  * @param   newValue: Value to add to average.
295  * @param   numerOfValues: Total number of values in average, including the new
     one.
296  * @retval  Integer: Average.
297  * \~catalan
298  * @brief   Torna el nou valor de la mitjana aritmètica amb el nou valor afegit.
299  * @param   oldAverage: Mitjana previa.
300  * @param   newValue: Valor a afegir a la mitjana.
301  * @param   numerOfValues: Nombre total de valors a la mitjana, incloent el nou.
302  * @retval  Integer: Mitjana.
303  */
304 real32_t AddToAverage(real32_t oldAverage, real32_t newValue, uint8_t
     numberOfValues)
305 {
306     return (oldAverage * (numberOfValues - 1) + newValue) / numberOfValues;
307 }
308
309
310 /**
311  * \~english
312  * @brief   Limited conversor from float values to char array. Based on c
     function ftoa().
313  * @param   f: Float value to convert.
314  * @retval  Pointer to the created char array.
315  * \~catalan
316  * @brief   Conversor limitat de valors float a arrays de caràcters. És basat en
     la funció de c ftoa();
317  * @param   f: Valor float a convertir.
318  * @retval  Punter a l'array de caràcters creat.
319  */
320 uint8_t *ftoaEspecial(real32_t f)
321 {
322     static uint8_t  buf[17];
323     uint8_t*        cp = buf;
324     uint8_t         rem;
325     uint32_t        l;
326
327     if(f < 0) {
328         *cp++ = '-';
329         f = -f;
330     }
331     l = (uint32_t)f;
332     f -= (real32_t)l;
333     rem = (uint8_t)(f * 10);
334     sprintf(cp, "%lu.%.1u", l, rem);
335     return buf;
336 }
```

A.3 System clock

```
/**
 * @file     clock.h
 * @author   Miquel Hernàndez Nicolau
 * @version  v1.0
 * \~english
 * @date     7th May 2016
 * @brief
 * @details
 * \~catalan
 * @date     7 de maig de 2016
 * @brief
 * @details
 */

#include "xc8_header.h"

uint32_t SystemClockFreq(void);
```

```
/**
 * @file     clock.c
 * @author   Miquel Hernàndez Nicolau
 * @version  v1.0
 * \~english
 * @date     7th May 2016
 * @brief
 * @details
 * \~catalan
 * @date     7 de maig de 2016
 * @brief
 * @details
 */

#include "clock.h"


/**
 * \~english
 * @brief   It returns the system clock frequency.
 * @retval  Long: Frequency in hertz.
 * \~catalan
 * @brief   Torna el valor de la freqüència del sistema.
 * @retval  Long: Freqüència en hertzs.
 */
uint32_t SystemClockFreq(void)
{
    uint8_t     i, ValorIRCF;
    uint16_t    Frequency = 8000;

    ValorIRCF = (OSCCON & 0x70) >> 4; //0x70 is the IRCF mask

    for(i = 7; i > ValorIRCF; i--)
    {
        Frequency >>= 1;    //Divided by 2 each time
        if(i == 1)          //The lowest freq. is divided twice
            Frequency >>= 1;
```

```
38      }
39
40      return (uint32_t)Frequency * 1000;
41  }
```

A.4 I²C

```
1   /**
2    * @file     i2c.h
3    * @author   Miquel Hernández Nicolau
4    * @version v1.0
5    * \~english
6    * @date     13th February 2016
7    * @brief    I2C header file with function prototypes.
8    * @details This header includes all the functions prototypes for the i2c.c
9    *           library.
10   * \~catalan
11   * @date     13 de febrer de 2016
12   * @brief    Arxiu header de l'I2C, amb els prototips de funcions.
13   * @details Aquest header inclou tots els prototips de funcions per a la
14   *           lliberia i2c.c.
15   */
16
17  #ifndef I2C_H
18  #define I2C_H
19
20  #include "xc8_header.h"
21  #include "clock.h"
22
23  #ifdef __cplusplus
24  extern "C" {
25  #endif
26
27  #define NotAcknowledge  0
28  #define Acknowledge     1
29
30
31  /** @defgroup Frequency_Range
32    * @{
33    */
34  #define I2C_Maximal_Frequency    ((uint32_t)100000)
35  #define I2C_Minimal_Frequency    ((uint32_t)10000)
36  #define IS_VALID_I2C_FREQUENCY(FREQUENCY)    (((FREQUENCY) <=
        I2C_Maximal_Frequency)&&\
37                                                ((FREQUENCY) >=
        I2C_Minimal_Frequency))
38  /**
39    * @}
40    */
41
42  /** @defgroup Message_Type
43    * @{
44    */
45  #define I2C_Restart          ((uint8_t)0x00)
46  #define I2C_Restart_Stop     ((uint8_t)0x01)
```

```
47 #define I2C_Start           ((uint8_t)0x02)
48 #define I2C_Start_Stop      ((uint8_t)0x03)
49 #define IS_MESSAGE_TYPE(TYPE)   (((TYPE) == I2C_Restart)||\
50                                  ((TYPE) == I2C_Restart_Stop)||\
51                                  ((TYPE) == I2C_Start)||\
52                                  ((TYPE) == I2C_Start_Stop))
53 /**
54   * @}
55   */
56
57 //Functions
58 uint8_t I2C_Init(uint32_t Frequency);
59 void    I2C_StartCondition(void);
60 void    I2C_StopCondition(void);
61 void    I2C_RestartCondition(void);
62 void    I2C_Ack(void);
63 void    I2C_Nak(void);
64 void    I2C_Wait(void);
65 void    I2C_Send(uint8_t dat);
66 uint8_t I2C_SendString(uint8_t *dat, uint8_t num, uint8_t restart);
67 uint8_t I2C_Read(uint8_t ack);
68 uint8_t I2C_ReadChar(uint8_t);
69
70
71
72 #ifdef __cplusplus
73 }
74 #endif
75
76 #endif /* I2C_H */
```

```
1  /**
2   * @file     i2c.c
3   * @author   Miquel Hernàndez Nicolau
4   * @version  v1.0
5   * \~english
6   * @date     13th February 2016
7   * @brief    Library functions to use the PIC16F886's I2C communication.
8   * @details  This file provides some library functions to use the MSSP
9   *           peripheral module of the PIC16F886. This module is configured to
10  *           be Master of an I2C bus.
11  * @see      http://www.8051projects.net/wiki/I2C_Implementation_on_PIC
12  * \~catalan
13  * @date     13 de febrer de 2016
14  * @brief    Llibreria de funcions per emprar la comunicació I2C del PIC16F886.
15  * @details  Aquest arxiu proporciona algunes funcions de llibreria per emprar
16  *           el perifèric MSSP del PIC16F886. Aquest mòdul és configurat per ser
17  *           el mestre d'un bus I2C.
18  * @see      http://www.8051projects.net/wiki/I2C_Implementation_on_PIC
19  */
20
21 #include "i2c.h"
22
23 /**
24  * \~english
25  * @brief    It initializes I2C in master mode and sets the required baudrate.
26  * @param    Frequency: It has to be a number into @ref Frequency_Range.
```

```
27  * @retval  Char: Returns EXIT_SUCCESS or EXIT_FAILURE.
28  * \~catalan
29  * @brief   Inicialitza el I2C en mode mestre i assigna la freqüència demanada.
30  * @param   Frequency: Ha de ser un nombre que es trobi dins @ref Frequency_Range
        .
31  * @retval  Char: Retorna EXIT_SUCCESS o EXIT_FAILURE.
32  */
33 uint8_t I2C_Init(uint32_t Frequency)
34 {
35     int16_t  aux;
36     uint32_t sysclk;
37
38     sysclk = SystemClockFreq();
39
40     //Amb SYS=31kHz mai es podrien tenir els 10kHz mínims al bus
41     if((sysclk == 31000) || (!IS_VALID_I2C_FREQUENCY(Frequency)))
42         return EXIT_FAILURE;
43
44     TRISC   |= 0x18;     //TRISC3 | TRISC4
45         //TRISC3 = 1; -> SDA and SCL as input pin
46         //TRISC4 = 1; -> These pins can be configured either i/p or o/p
47     SSPSTAT |= 0x80;     //Slew rate disabled
48     SSPCON   = 0x28;     //SSPEN = 1, I2C Master mode, clock = FOSC/(4 * (SSPADD +
        1))
49
50     aux = (int16_t)(sysclk / 4 / Frequency - 1);
51     if(aux > 127)
52         aux = 127;
53     else if((aux < 1) || ((sysclk == 1000000) && (aux == 1))) //Problemes per l'
       arrodoniment
54         aux++;
55
56     SSPADD = (SSPADD & 0x7F) | (uint8_t)aux;
57
58     return EXIT_SUCCESS;
59 }
60
61 /**
62  * \~english
63  * @brief   It sends a start condition on I2C Bus.
64  * @retval  None.
65  * \~catalan
66  * @brief   Envia una condició de start al bus I2C.
67  * @retval  Cap.
68  */
69 void I2C_StartCondition(void)
70 {
71     SEN = 1;
72     while(SEN); //Automatically cleared by hardware
73                 //Wait for start condition to finish
74 }
75
76 /**
77  * \~english
78  * @brief   It sends a stop condition on I2C Bus.
79  * @retval  None.
80  * \~catalan
```

```
81  * @brief   Envia una condició de stop al bus I2C.
82  * @retval  Cap.
83  */
84 void I2C_StopCondition(void)
85 {
86     PEN = 1;
87     while(PEN); //Wait for stop condition to finish
88                 //PEN automatically cleared by hardware
89 }
90
91 /**
92  * \~english
93  * @brief   It sends a repeated start condition on I2C Bus.
94  * @retval  None.
95  * \~catalan
96  * @brief   Envia una condició de repeated start al bus I2C.
97  * @retval  Cap.
98  */
99 void I2C_RestartCondition(void)
100 {
101     RSEN = 1;
102     while(RSEN);    //Wait for condition to finish
103 }
104
105 /**
106  * \~english
107  * @brief   It generates acknowledge for a transfer.
108  * @retval  None.
109  * \~catalan
110  * @brief   Genera un acknowledge per transferir.
111  * @retval  Cap.
112  */
113 void I2C_Ack(void)
114 {
115     ACKDT = 0;      //Acknowledge data bit, 0 = ACK
116     ACKEN = 1;      //Ack data enabled
117     while(ACKEN);   //Wait for ack data to send on bus
118 }
119
120 /**
121  * \~english
122  * @brief   It generates Not-acknowledge for a transfer.
123  * @retval  None.
124  * \~catalan
125  * @brief   Genera un Not-acknowledge per transferir.
126  * @retval  Cap.
127  */
128 void I2C_Nak(void)
129 {
130     ACKDT = 1;      //Acknowledge data bit, 1 = NAK
131     ACKEN = 1;      //Ack data enabled
132     while(ACKEN);   //Wait for ack data to send on bus
133 }
134
135 /**
136  * \~english
137  * @brief   It waits for transfer to finish.
```

```
138  * @retval  None.
139  * \~catalan
140  * @brief   Espera a que acabin les transferències.
141  * @retval  Cap.
142  */
143 void I2C_Wait(void)
144 {
145     while((SSPCON2 & 0x1F ) || (READ_WRITE));
146 }
147 
148 /**
149  * \~english
150  * @brief   It sends 8-bit data on I2C bus.
151  * @param   Data: 8-bit data to be sent on bus. It can be either address or
152  *          data byte.
153  * @retval  None.
154  * \~catalan
155  * @brief   Envia una dada de 8 bits al bus I2C.
156  * @param   Data: Dada de 8 bits a enviar al bus. Pot ser un byte d'adreça o de
157  *          dada.
158  * @retval  Cap.
159  */
160 void I2C_Send(uint8_t Data)
161 {
162     SSPBUF = Data;   //Move data to SSPBUF
163     while(BF);       //Wait till complete data is sent from buffer
164     I2C_Wait();      //Wait for any pending transfer
165 
166     if(ACKSTAT)      //Stop if not acknowledged
167         I2C_StopCondition();
168 }
169 
170 /**
171  * \~english
172  * @brief   It sends a data char array on I2C bus.
173  * @param   Data: Pointer to a data char array to be sent on bus.
174  * @param   Length: Number of array's chars.
175  * @param   Type: Kind of frame that will be sent. It has to be a value of
176  *          @ref Message_Type :
177  *              @arg I2C_Restart
178  *              @arg I2C_Restart_Stop
179  *              @arg I2C_Start
180  *              @arg I2C_Start_Stop
181  * @retval  Char: Returns EXIT_SUCCESS or EXIT_FAILURE..
182  * \~catalan
183  * @brief   Envia un array de caràcters de dades al bus I2C.
184  * @param   Data: Punter a un array de caràcters amb dades per enviar al bus.
185  * @param   Length: Nombre de caràcters a l'array.
186  * @param   Type: Tipus de trama que s'enviarà. Ha de ser un valor de
187  *          @ref Message_Type :
188  *              @arg I2C_Restart
189  *              @arg I2C_Restart_Stop
190  *              @arg I2C_Start
191  *              @arg I2C_Start_Stop
192  * @retval  Char: Retorna EXIT_SUCCESS o EXIT_FAILURE.
193  */
194 uint8_t I2C_SendString(uint8_t *Data, uint8_t Length, uint8_t Type)
```

```
195 {
196     uint8_t i;
197 
198     if(!IS_MESSAGE_TYPE(Type))
199         return EXIT_FAILURE;
200     else
201     {
202         if(Type & 0x02) //Vàlid per 2 i 3
203             I2C_StartCondition();
204         else
205             I2C_RestartCondition();
206 
207         for(i = 0; i < Length; i++)
208             I2C_Send(Data[i]);
209 
210         if (Type & 0x01) //Vàlid per 1 i 3
211             I2C_StopCondition();
212 
213         return EXIT_SUCCESS;
214     }
215 }
216 
217 /**
218  * \~english
219  * @brief    It reads 8-bit data from I2C bus.
220  * @param    ack: Makes an acknowledge or a not-acknowledge after reading.
221  *           It can be "Acknowledge" or "NotAcknowledge".
222  * @retval   Char: Returns the data that has been read.
223  * \~catalan
224  * @brief    Llegeix dades de 8 bits del bus I2C.
225  * @param    ack: Fa un acknowledge o un not-acknowledge després de llegir.
226  *           Se'l pot assignar el valor "Acknowledge" o "NotAcknowledge".
227  * @retval   Char: Retorna la dada que s'ha llegit.
228  */
229 uint8_t I2C_Read(uint8_t ack)
230 {
231     uint8_t temp;
232     //Reception works if transfer is initiated in read mode
233     RCEN = 1;        //Enable data reception
234     while(!BF);      //Wait for buffer full
235     temp = SSPBUF;   //Read serial buffer and store in temp register
236     I2C_Wait();      //Wait to check any pending transfer
237 
238     if(ack)
239         I2C_Ack();
240     else
241         I2C_Nak();
242 
243     return temp;     //Return the read data from bus
244 }
245 
246 /**
247  * \~english
248  * @brief    It starts, reads a single 8-bit data from I2C bus and stops it.
249  * @param    address: I2C slave address.
250  * @retval   Char: Returns the data that has been read.
251  * \~catalan
```

```
252  * @brief   Inicia, llegeix una sola dada de 8 bits del bus I2C i l'atura.
253  * @param   address: Adreça de l'esclau I2C
254  * @retval  Char: Retorna la dada que s'ha llegit.
255  */
256 uint8_t I2C_ReadChar(uint8_t address)
257 {
258     uint8_t tmp;
259 
260     I2C_StartCondition();
261     I2C_Send(address);
262     tmp = I2C_Read(Acknowledge);
263     I2C_StopCondition();
264 
265     return tmp;
266 }
```

A.5 UART

```
1  /**
2   * @file     uart.h
3   * @author   Miquel Hernández Nicolau
4   * @version  v1.0
5   * \~english
6   * @date     12th February 2016
7   * @brief    UART function prototypes.
8   * @details  This header includes all the functions prototypes for the uart.c
9   *           library.
10  * \~catalan
11  * @date     12 de febrer de 2016
12  * @brief    Prototips de funció de l'UART.
13  * @details  Aquest header inclou tots els prototips de funció per a la
14  *           llibreria uart.c.
15  */
16 
17 #ifndef UART_H
18 #define UART_H
19 
20 #include "xc8_header.h"
21 #include "clock.h"
22 
23 
24 #ifdef __cplusplus
25 extern "C" {
26 #endif
27 
28 uint8_t UART_Init(const uint32_t BaudRate);
29 void    UART_Write(uint8_t Data);
30 uint8_t UART_TX_Empty();
31 void    UART_Write_Text(uint8_t *Text);
32 uint8_t UART_Data_Ready();
33 uint8_t UART_Read();
34 void    UART_Read_Text(uint8_t *Output, uint16_t Length);
35 
36 
37 #ifdef __cplusplus
```

```
38 }
39 #endif
40
41 #endif /* UART_H */
```

```
1  /**
2   * @file     uart.c
3   * @author   Miquel Hernàndez Nicolau
4   * @version  v1.0
5   * \~english
6   * @date     3rd February 2016
7   * @brief    Library functions to use the PIC16F886's EUSART.
8   * @details  This file provides some library functions to use the EUSART module
9   *           of the PIC16F886. It is configured to be used as UART, i.e. in
10  *           asynchronous mode.
11  * @see      https://electrosome.com/uart-pic-microcontroller-mplab-xc8/
12  * @see      https://github.com/CytronTechnologies/Xbee-WiFi-Mobile-Robot/blob/
      master/WiFiMobileRobot.X/uart.h
13  * \~catalan
14  * @date     3 de febrer de 2016
15  * @brief    Llibreria de funcions per emprar el EUSART del PIC16F886.
16  * @details  Aquest arxiu proporciona algunes funcions de llibreria per emprar
17  *           el mòdul EUSART del PIC16F886. La configuració està feta per ser
18  *           emprat com a UART, és a dir, en mode asíncrone.
19  * @see      https://electrosome.com/uart-pic-microcontroller-mplab-xc8/
20  * @see      https://github.com/CytronTechnologies/Xbee-WiFi-Mobile-Robot/blob/
      master/WiFiMobileRobot.X/uart.h
21  */
22
23 #include "uart.h"
24
25 /**
26  * \~english
27  * @brief    It enables and configures the UART communication of the PIC.
28  * @param    BaudRate: Baud rate at which the UART will transmit.
29  * @retval   Char: Returns EXIT_SUCCESS or EXIT_FAILURE.
30  * \~catalan
31  * @brief    Habilita i configura la comunicació UART del PIC.
32  * @param    BaudRate: Rati de bauds als quals trasmetrà l'UART.
33  * @retval   Char: Retorna EXIT_SUCCESS o EXIT_FAILURE.
34  */
35 uint8_t UART_Init(const uint32_t BaudRate)
36 {
37     uint16_t x = 0;
38     uint32_t sysclk;
39
40     sysclk = SystemClockFreq();
41
42     //SPBRG:
43     x = sysclk / BaudRate;
44     if(x < 64)       //If High Baud Rage Required
45     {
46         x /= 16;     //SPBRG for High Baud Rate: sysclk/(BaudRate*16)-1;
47         BRGH = 1;    //Setting High Baud Rate
48     }
49     else
50     {
```

```
51          x /= 64;     //SPBRG for High Baud Rate: sysclk/(BaudRate*64)-1;
52          BRGH = 0;
53      }
54      x--;
55
56      if((x < 256) && (x > 0))
57      {
58          SPBRG  = x;      //Writing SPBRG Register
59          SYNC   = 0;      //Setting Asynchronous Mode, ie UART
60          TRISC |= 0xC0;   //TRISC6 | TRISC7
61          TXEN   = 1;      //Enables Transmission
62          RCSTA |= 0x90;   //SPEN = 1, CREN = 1: Enables Serial Port & Continuous
    Reception
63
64          return EXIT_SUCCESS; //Returns 0 to indicate Successful Completion
65      }
66      else
67          return EXIT_FAILURE; //Returns 1 to indicate UART initialization failed
68  }
69
70
71  /**
72   * \~english
73   * @brief    It writes a char to UART.
74   * @param    Data: Char to be send to UART.
75   * @retval   None.
76   * \~catalan
77   * @brief    Escriu caràcter a l'UART.
78   * @param    Data: Caràcter a enviar per l'UART.
79   * @retval   Cap.
80   */
81  void UART_Write(uint8_t Data)
82  {
83      while(!TRMT);
84      TXREG = Data;
85  }
86
87  /**
88   * \~english
89   * @brief    Trasmission buffer status.
90   * @retval   Char: Returns 1 if buffer is empty, and 0 if it is not.
91   * \~catalan
92   * @brief    Estat del buffer de transmissió.
93   * @retval   Char: Retorna 1 si el buffer és buit, i 0 si no ho és.
94   */
95  uint8_t UART_TX_Empty()
96  {
97      return TRMT;
98  }
99
100 /**
101  * \~english
102  * @brief    It writes a char array to UART.
103  * @param    Text: Char array to be send to UART.
104  * @retval   None.
105  * \~catalan
106  * @brief    Escriu una cadena de caràcters a l'UART.
```

```
107  * @param   Text: Array de caràcters a enviar per l'UART.
108  * @retval  Cap.
109  */
110 void UART_Write_Text(uint8_t *Text)
111 {
112     while(*Text)
113     {
114         UART_Write(*Text);
115         Text++;
116     }
117 }
118 
119 /**
120  * \~english
121  * @brief   Reception buffer status.
122  * @retval  Char: Returns the interruption flag that is 1 if there is an unread
123  *          character in the buffer. Otherwise, it is 0.
124  * \~catalan
125  * @brief   Estat del buffer de recepció.
126  * @retval  Char: Retorna el flag d'interrupció que val 1 si hi ha un caràcter
127  *          al buffer. En altre cas val 0.
128  */
129 uint8_t UART_Data_Ready()
130 {
131     return RCIF;
132 }
133 
134 /**
135  * \~english
136  * @brief   It reads a char from UART.
137  * @retval  Char: Read data.
138  * \~catalan
139  * @brief   Llegeix un caràcter des de l'UART.
140  * @retval  Char: Dada llegida.
141  */
142 uint8_t UART_Read()
143 {
144     if(OERR)
145     {
146         CREN = 0;
147         CREN = 1;
148     }
149     while(!RCIF);
150     return RCREG;
151 }
152 
153 /**
154  * \~english
155  * @brief   It reads a char array from UART.
156  * @param   Output: Pointer to char array to write the read values.
157  * @param   Length: Number of chars to be read.
158  * @retval  None.
159  * \~catalan
160  * @brief   Llegeix un array de caràcters des de l'UART.
161  * @param   Output: Punter a un array de caràcters per escriure les lectures.
162  * @param   Length: Nombre de caràcters a llegir.
163  * @retval  Cap.
```

```
 */
void UART_Read_Text(uint8_t *Output, uint16_t Length)
{
    uint8_t i;

    for(i = 0; i < Length; i++)
        Output[i] = UART_Read();
}
```

A.6 xBee - Sentilo

```
/**
 * @file     xbee.h
 * @author   Miquel Hernández Nicolau
 * @version v1.0
 * \~english
 * @date     12th February 2016
 * @brief    xBee function prototypes and definitions.
 * @details This header includes all the functions prototypes for the xbee.c
 *           library.
 * \~catalan
 * @date     12 de febrer de 2016
 * @brief    Prototips de funcions i definicions per xBee.
 * @details Aquest header inclou tots els prototips de funcions per a la
 *           llibreria xbee.c.
 */

#ifndef XBEE_H
#define XBEE_H

#include "uart.h" //uart.h inclou xc8_header.h


#ifdef __cplusplus
extern "C" {
#endif

#define COMMAND_TIMEOUT 2000 //ms

const uint8_t*  CONTENT_TYPE = "application/json";
const uint8_t*  IDENTITY_KEY_HEADER = "identity_key";
const uint8_t*  DATA_BASE_PATH = "/data";

//SentiloXBee_client
const uint8_t*  SentiloXBee_host = "192.168.1.100";

const uint8_t   (*SentiloXBee_sensorId[4])[8] = {"TSL2550","SHT31-T","SHT31-H","
    TGS2600"};
uint8_t         SentiloXBee_sensVal[4][7];


//UART functions:
static void     SentiloXBee_emptySerial(void);
static uint8_t  SentiloXBee_read(void);
static uint16_t SentiloXBee_waitForAvailable(void);
```

```
44 static void     SentiloXBee_write(const uint8_t* string);
45
46 //xBee functions:
47 static uint8_t  SentiloXBee_commandMode(uint8_t enter);
48 static uint8_t  SentiloXBee_responseOK(void);
49 uint8_t         SentiloXBee_checkConnection(void);
50 uint8_t         SentiloXBee_sleepEnable(uint8_t mode);
51
52 //Sentilo functions:
53 uint8_t         SentiloXBee_publishObservation(const uint8_t* providerId, uint8_t
       sensors, const uint8_t* apiKey);
54
55
56 #ifdef __cplusplus
57 }
58 #endif
59
60 #endif /* XBEE_H */
```

```
1  /**
2   * @file     xbee.c
3   * @author   Miquel Hernàndez Nicolau
4   * @version  v1.0
5   * \~english
6   * @date     12th February 2016
7   * @brief    Library functions to use xBee.
8   * @details  This file provides some library functions to make easier the use
9   *           of the xBee S6B component with Sentilo Platform.
10  * \~catalan
11  * @date     12 de febrer de 2016
12  * @brief    Llibreria de funcions per emprar el xBee.
13  * @details  Aquest arxiu proporciona algunes funcions de llibreria per
14  *           facilitar l'ús del component xBee S6B amb la plataforma Sentilo.
15  */
16
17 #include "xbee.h"
18
19
20 ////////////////////////////////////// UART ///////////////////////////////////////
21 /**
22  * \~english
23  * @brief    Waits for a byte at UART buffer.
24  * @retval   Integer: If 0 there is no more available data.
25  * \~catalan
26  * @brief    Espera a que el buffer de l'UART hagi rebut un byte.
27  * @retval   Integer: Si és 0 no hi ha més dades disponibles.
28  */
29 static uint16_t SentiloXBee_waitForAvailable(void)
30 {
31     uint16_t timeout = COMMAND_TIMEOUT;
32
33     while((timeout > 0) && (!UART_Data_Ready()))
34     {
35         __delay_ms(1);
36         timeout--;
37     }
38
```

```
39      return timeout;
40  }
41
42  /**
43   * \~english
44   * @brief   Read data until buffer is empty.
45   * @retval  None.
46   * \~catalan
47   * @brief   Llegeix dades fins buidar el buffer.
48   * @retval  Cap.
49   */
50  static void SentiloXBee_emptySerial(void)
51  {
52      while(SentiloXBee_waitForAvailable())
53          UART_Read();
54  }
55
56  /**
57   * \~english
58   * @brief   Writes string to the UART.
59   * @param   string: Pointer to string to write.
60   * @retval  None.
61   * \~catalan
62   * @brief   Escriu cadena de caràcters a l'UART.
63   * @param   string: Punter a la cadena de caràcters a escriure.
64   * @retval  Cap.
65   */
66  static void SentiloXBee_write(const uint8_t* string)
67  {
68      UART_Write_Text(string);
69  }
70
71  /**
72   * \~english
73   * @brief   Reads a byte from UART.
74   * @retval  Char: Byte read.
75   * \~catalan
76   * @brief   Llegeix byte de l'UART.
77   * @retval  Char: Byte llegit.
78   */
79  static uint8_t SentiloXBee_read(void)
80  {
81      return UART_Read();
82  }
83
84
85  ////////////////////////////////// XBEE ////////////////////////////////////
86  /**
87   * \~english
88   * @brief   Starts or stops command mode on xBee.
89   * @param   enter: If it is 0 exits command mode, otherwise command mode is set.
90   * @retval  Char: EXIT_SUCCESS or EXIT_FAILURE.
91   * \~catalan
92   * @brief   Inicia o atura el mode comanda al xBee.
93   * @param   enter: Si és 0 surt del mode comanda, si no s'activa el mode comanda.
94   * @retval  Char: EXIT_SUCCESS o EXIT_FAILURE.
95   */
```

```
96  static uint8_t SentiloXBee_commandMode(uint8_t enter)
97  {
98      if (enter)
99      {
100         uint8_t c;
101         //Send CMD mode string
102         __delay_ms(1000);
103         SentiloXBee_write("+++");
104         __delay_ms(1000);
105
106         if(SentiloXBee_responseOK())
107             return EXIT_SUCCESS;
108
109         return EXIT_FAILURE; //If no (or incorrect) receive, return fail
110     }
111     else
112     {
113         SentiloXBee_write("ATCN\r");
114         SentiloXBee_emptySerial();
115         return EXIT_SUCCESS;
116     }
117 }
118
119 /**
120  * \~english
121  * @brief   Checks if response to a command is "OK".
122  * @retval  Char: If response is "OK" it returns TRUE, if not it returns FALSE.
123  * \~catalan
124  * @brief   Comprova si la respota a l'ordre és "OK".
125  * @retval  Char: Si la resposta és "OK" torna TRUE, si no torna FALSE.
126  */
127 static uint8_t SentiloXBee_responseOK(void)
128 {
129     uint8_t isOK = FALSE;
130
131     if(SentiloXBee_waitForAvailable())
132     {
133         if(SentiloXBee_read() == 'O') //That's the letter 'O', assume 'K' is next
134             isOK = TRUE;
135
136         SentiloXBee_emptySerial();
137     }
138
139     return isOK;
140 }
141
142 /**
143  * \~english
144  * @brief   Check if the module is connected to an access point (AP).
145  * @retval  Char: EXIT_SUCCESS if it is connected to an AP, and EXIT_FAILURE
      otherwise.
146  * \~catalan
147  * @brief   Comprova si el mòdul és connectat a un punt d'accés (PA).
148  * @retval  Char: EXIT_SUCCESS si està connectat a un PA, i EXIT_FAILURE en
      altres casos.
149  */
150 uint8_t SentiloXBee_checkConnection(void)
```

```
151 {
152     uint8_t connectionType = 0xFF;
153 
154     SentiloXBee_commandMode(TRUE);
155     SentiloXBee_write("ATAI\r");
156 
157     if(SentiloXBee_waitForAvailable())
158     {
159         connectionType = SentiloXBee_read();
160         SentiloXBee_emptySerial();
161         SentiloXBee_commandMode(FALSE);
162 
163         if(connectionType == 0x30)
164             return EXIT_SUCCESS; //Successfully connected
165         else
166             return EXIT_FAILURE;
167     }
168     else
169         return EXIT_FAILURE;
170 }
171 
172 /**
173  * \~english
174  * @brief   Enables or disables the sleep mode on xBee.
175  * @param   mode: If 0 sleep mode is disabled, otherwise is enabled.
176  * @retval  Char: EXIT_SUCCESS or EXIT_FAILURE.
177  * \~catalan
178  * @brief   Habilita o deshabilita el mode sleep del xBee.
179  * @param   mode: Si és 0 es deshabilita el mode sleep, en altre cas s'habilita.
180  * @retval  Char: EXIT_SUCCESS o EXIT_FAILURE.
181  */
182 uint8_t SentiloXBee_sleepEnable(uint8_t mode)
183 {
184     uint8_t msg[7];
185 
186     if(mode == 0)
187         sprintf(msg,"ATSM0\r");
188     else
189         sprintf(msg,"ATSM4\r");
190 
191     SentiloXBee_commandMode(TRUE);
192     SentiloXBee_write(msg);
193 
194     msg[0] = SentiloXBee_responseOK();
195     SentiloXBee_commandMode(FALSE);
196 
197     if(msg[0])
198         return EXIT_SUCCESS;
199 
200     else
201         return EXIT_FAILURE;
202 
203 }
204 
205 
206 ////////////////////////////////// SENTILO ////////////////////////////////////
207 /**
```

```
208  * \~english
209  * @brief    Sends observations to a component of a Sentilo platform.
210  * @param    providerId: String with provider identifier name.
211  * @param    sensors: Number of sensors at the component.
212  * @param    apiKey: Key provided by Sentilo to the provider.
213  * @retval   Char: EXIT_SUCCESS or EXIT_FAILURE.
214  * \~catalan
215  * @brief    Envia observacions a un component d'una plataforma Sentilo.
216  * @param    providerId: String amb el nom identificador del proveïdor.
217  * @param    sensors: Nombre de sensors al component.
218  * @param    apiKey: Contrasenya donada per Sentilo al proveïdor.
219  * @retval   Char: EXIT_SUCCESS o EXIT_FAILURE.
220  */
221 uint8_t SentiloXBee_publishObservation(const uint8_t* providerId, uint8_t sensors
         , const uint8_t* apiKey)
222 {
223     uint8_t timeoutConnection = 10; //Cada unitat són uns 3-4 segons
224 
225     while(SentiloXBee_checkConnection() && timeoutConnection)
226     {
227         timeoutConnection--;
228         __delay_ms(500);
229     }
230 
231     if(timeoutConnection)
232     {
233         SentiloXBee_write("PUT ");
234         SentiloXBee_write(DATA_BASE_PATH);
235         SentiloXBee_write("/");
236         SentiloXBee_write(providerId);
237         SentiloXBee_write(" HTTP/1.1\r\n");
238 
239         SentiloXBee_write(IDENTITY_KEY_HEADER);
240         SentiloXBee_write(": ");
241         SentiloXBee_write(apiKey);
242         SentiloXBee_write("\r\n");
243 
244         SentiloXBee_write("Host: ");
245         SentiloXBee_write(SentiloXBee_host);
246 
247         SentiloXBee_write("\r\nConnection: close\r\n");
248 
249 
250         if(sensors)
251         {
252             uint8_t contentLength[6];               //5 xifres i 1 NULL
253             uint16_t bodylength;
254 
255             bodylength = sensors + 13;              //14 de trama, sensors - 1 de
          comes
256             for(uint8_t i = 0; i < sensors; i++)    //Llarg del valor, llarg del
         nom i 43 de trama
257                 bodylength += strlen(SentiloXBee_sensVal[i]) + strlen(
         SentiloXBee_sensorId[i]) + 43;
258 
259             SentiloXBee_write("Content-Length: ");
260             sprintf(contentLength, "%u", bodylength);
```

```
261             SentiloXBee_write(contentLength);
262 
263             SentiloXBee_write("\r\nContent-Type: ");
264             SentiloXBee_write(CONTENT_TYPE);
265             SentiloXBee_write("\r\n");
266 
267             SentiloXBee_write("\r\n");
268 
269             SentiloXBee_write("{\"sensors\":[");
270             for(uint8_t i = 0; i < sensors; i++)
271             {
272                 if(i)
273                     SentiloXBee_write(","); //El primer valor és sense coma
    davant
274                 SentiloXBee_write("{\"sensor\":\"");
275                 SentiloXBee_write(SentiloXBee_sensorId[i]);
276                 SentiloXBee_write("\",\"observations\":[{\"value\":\"");
277                 SentiloXBee_write(SentiloXBee_sensVal[i]);
278                 SentiloXBee_write("\"}]}");
279             }
280             SentiloXBee_write("]}\r\n");
281         }
282 
283         SentiloXBee_write("\r\n");
284 
285         //Make sure you write all those bytes.
286         __delay_ms(100);
287 
288         return EXIT_SUCCESS;
289     }
290     else
291         return EXIT_FAILURE;
292 }
```

A.7 SHT71

```
1  /**
2   * @file    SHT71.h
3   * @author  Miquel Hernàndez Nicolau
4   * @version v1.0
5   * \~english
6   * @date    7th July 2016
7   * @brief   Header file for SHT71.c
8   * \~catalan
9   * @date    7 de juliol de 2016
10  * @brief   Fitxer header per a SHT71.c
11  */
12 
13 #ifndef SHT71_H
14 #define SHT71_H
15 
16 #include "xc8_header.h"
17 
18 #ifdef __cplusplus
19 extern "C" {
```

```
#endif

#define SDA      RC4
#define SCL      RC3

#define NAK      0
#define ACK      1


/** @defgroup SHT71_Basic_Commands
  * @{
  */                                //adr  command  r/w
#define SHT71_READ_T     0x03      //000   0001     1
#define SHT71_READ_H     0x05      //000   0010     1
 /**
  * @}
  */


uint16_t         SHT71_Read(uint8_t Parameter);
static uint8_t   SHT71_write_byte(uint8_t value);
static uint8_t   SHT71_read_byte(uint8_t ack);
void             SHT71_transstart(void);
void             SHT71_connectionreset(void);
uint8_t          SHT71_measure(uint8_t *p_value, uint8_t mode);


#ifdef __cplusplus
}
#endif

#endif /* SHT31_H */
```

```
/**
 * @file     SHT71.c
 * @author   Miquel Hernàndez Nicolau
 * @version v1.0
 * \~english
 * @date     7th July 2016
 * @brief    Library functions to use the SHT71 sensor through SENSI-BUS.
 * @details
 * @see
 * \~catalan
 * @date     7 de juliol de 2016
 * @brief    Llibreria de funcions per emprar el sensor SHT71 amb SENSI-BUS.
 * @details
 * @see
 */

#include "SHT71.h"


/**
 * \~english
 * @brief    Reads temperature and humidity from SHT71 sensor.
 * @param    Parameter: It is the measurement required. It can be a value of
 *           @ref SHT71_Basic_Commands :
```

```
25  *               @arg SHT71_READ_H
26  *               @arg SHT71_READ_T
27  * @retval  Float: Returns temperature or humidity value. If an error
28  *          has occurred the returning value will be 0xFFFF.
29  * \~catalan
30  * @brief   Lectura de temperatura i humitat al sensor SHT71.
31  * @param   Parameter: És el paràmetre que es demana saber. Pot ser
32  *          un valor de @ref SHT71_Basic_Commands :
33  *               @arg SHT71_READ_H
34  *               @arg SHT71_READ_T
35  * @retval  Float: Retorna el valor de temperatura o humitat. Si hi ha
36  *          hagut un error el valor serà 0xFFFF.
37  */
38 uint16_t SHT71_Read(uint8_t Parameter)
39 {
40     uint16_t Message;
41
42     SHT71_connectionreset();
43
44     if(SHT71_measure((uint8_t*) &Message, Parameter)) //Send command
45         return 0xFFFF;
46
47     return Message;
48 }
49
50
51
52 /**
53  * \~english
54  * @brief   Writes a byte on the bus and checks the acknowledge.
55  * @param   value: Byte to be write on the sensor.
56  * @retval  Char: Returns an error if data has not been acknowledged.
57  * \~catalan
58  * @brief   Escriu un byte al bus del sensor i comprova el acknowledge.
59  * @param   value: Byte a escriure al sensor.
60  * @retval  Char: Retorna un error si després d'enviar la dada no s'ha
61  *          rebut un acknowledge.
62  */
63 static uint8_t SHT71_write_byte(uint8_t value)
64 {
65     uint8_t i, error = 0;
66
67     //Shift bit for masking
68     for(i = 0x80; i > 0; i /= 2)
69     {
70         if(i & value)   //Masking value with i, write to bus
71             SDA = 1;
72         else
73             SDA = 0;
74         NOP();          //Observe setup time
75         NOP();
76         SCL = 1;        //Clk for SENSI-BUS
77         __delay_us(5);  //Pulswith approx. 5 us
78         SCL = 0;
79         NOP();          //Observe hold time
80         NOP();
81     }
```

```
82
83      TRISC4 = 1;             //SDA is input, release SDA-line
84      NOP();                  //Observe setup time
85      NOP();
86      SCL = 1;                //Clk #9 for ack
87      error = SDA;            //Check ack (SDA will be pulled down by SHT71)
88      NOP();
89      NOP();
90      SCL = 0;
91      NOP();
92      NOP();
93      TRISC4 = 0;             //SDA is output
94
95      return error;           //error=1 in case of no acknowledge
96  }
97
98  /**
99   * \~english
100  * @brief   Reads a byte form the Sensibus and gives an acknowledge.
101  * @param   ack: Used to acknowledge (if 1) or not-acknowledge after reading.
102  * @retval  Char: Read byte.
103  * \~catalan
104  * @brief   Llegeix un byte del Sensibus i respon amb un acknowledge.
105  * @param   ack: Emprat per fer acknowledge (si és 1) o not-acknowledge
106  *          després de llegir.
107  * @retval  Char: Byte llegit.
108  */
109 static uint8_t SHT71_read_byte(uint8_t ack)
110 {
111     uint8_t i, val = 0;
112
113     TRISC4 = 1;     //SDA is input, release SDA-line
114     //Shift bit for masking
115     for(i = 0x80; i > 0; i /= 2)
116     {
117         SCL = 1;    //Clk for SENSI-BUS
118         NOP();
119         NOP();
120         if(SDA)     //Read bit
121             val = (val | i);
122         SCL = 0;
123         NOP();
124         NOP();
125     }
126     TRISC4 = 0;     //SDA is output
127     SDA = !ack;     //In case of "ack==1" pull down SDA-Line
128     NOP();          //Observe setup time
129     NOP();
130     SCL = 1;        //Clk #9 for ack
131     __delay_us(5);  //Pulswith approx. 5 us
132     SCL = 0;
133     NOP();          //Observe hold time
134     NOP();
135     SDA = 1;        //Release SDA-line
136
137     return val;
138 }
```

```
139
140 /**
141  * \~english
142  * @brief   Generates a transmission start.
143  * @retval  None.
144  * \~catalan
145  * @brief   Genera una condició de començament.
146  * @retval  Cap.
147  */
148 //-------------------------------
149 //       _____          _____
150 // SDA:       |________|
151 //           ___      ___
152 // SCL: ___|   |___|   |___
153 //-------------------------------
154 void SHT71_transstart(void)
155 {
156     SDA = 1;
157     SCL = 0; //Initial state
158     NOP();
159     NOP();
160     SCL = 1;
161     NOP();
162     NOP();
163     SDA = 0;
164     NOP();
165     NOP();
166     SCL = 0;
167     NOP();
168     NOP();
169     NOP();
170     NOP();
171     NOP();
172     SCL = 1;
173     NOP();
174     NOP();
175     SDA = 1;
176     NOP();
177     NOP();
178     SCL = 0;
179 }
180
181 /**
182  * \~english
183  * @brief   Communication reset:SDA-line=1 and at least 9 SCL cycles followed
184  *          by transstart.
185  * @retval  None.
186  * \~catalan
187  * @brief   Reset de comunicació: Línia-SDA=1 i al manco 9 cicles de SCL seguits
188  *          de transstart.
189  * @retval  Cap.
190  */
191 //---------------------------------------------------------------------------
192 //       _____________________________________________________          _____
193 // SDA:                                                       |________|
194 //          _    _    _    _    _    _    _    _    _         ___      ___
195 // SCL: __| |__| |__| |__| |__| |__| |__| |__| |__| |_______|   |___|   |___
```

```
196 //-------------------------------------------------------------------------------------
197 void SHT71_connectionreset(void)
198 {
199     uint8_t i;
200     SDA = 1;
201     SCL = 0;                //Initial state
202     for(i = 0; i < 9; i++)  //9 SCL cycles
203     {
204         SCL = 1;
205         SCL = 0;
206     }
207     SHT71_transstart();     //Transmission start
208 }
209
210
211 /**
212  * \~english
213  * @brief   Makes a measurement (humidity/temperature)
214  * @param   p_value: Pointer to save the measurement.
215  * @param   mode: It is the measurement required. It can be a value of
216  *          @ref SHT71_Basic_Commands :
217  *              @arg SHT71_READ_H
218  *              @arg SHT71_READ_T
219  * @retval  Char: Number of errors.
220  * \~catalan
221  * @brief   Realitza la mesura del paràmetre demanat.
222  * @param   p_value: Punter per a guardar la mesura.
223  * @param   mode: És el paràmetre que es demana saber. Pot ser
224  *          un valor de @ref SHT71_Basic_Commands :
225  *              @arg SHT71_READ_H
226  *              @arg SHT71_READ_T
227  * @retval  Char: Nombre d'errors.
228  */
229 uint8_t SHT71_measure(uint8_t *p_value, uint8_t mode)
230 {
231     uint8_t     error = 0;
232     uint16_t    i;
233
234     SHT71_transstart();                     //Transmission start
235     error += SHT71_write_byte(mode);
236
237
238     TRISC4 = 1;                             //SDA is input
239     for(i = 0; (i < 65535) && SDA; i++);    //Wait until sensor has finished the
        measurement
240     if(SDA)                                 //or timeout (~2 sec.) is reached
241         error++;
242
243     TRISC4 = 0;                             //SDA is output
244
245     p_value[1]  = SHT71_read_byte(ACK);     //Read the 1st byte (MSB)
246     p_value[0]  = SHT71_read_byte(NAK);     //Read the 2nd byte (LSB)
247
248     return error;
249 }
```

A.8 TSL2550

```
/**
 * @file     TSL2550D.h
 * @author   Miquel Hernàndez Nicolau
 * @version  v1.0
 * \~english
 * @date     13th April 2016
 * @brief    Header file for TSL2550D.c
 * \~catalan
 * @date     13 d'abril de 2016
 * @brief    Fitxer header per TSL2550D.c
 */

#ifndef TSL2550D_H
#define TSL2550D_H

#include "i2c.h"     //Ja inclou "xc8_header.h"
//#include <math.h> //Funció exp()

#ifdef __cplusplus
extern "C" {
#endif

//I2C
#define LIGHT_WRITE ((uint8_t)0x72)
#define LIGHT_READ  ((uint8_t)0x73)

/** @defgroup TSL2550_aux_values_linealization
  * @{
  */
#define exp1     2.71828182845905
#define exp2     7.38905609893065
#define exp3     20.0855369231877
#define exp4     54.5981500331442
#define exp5     148.413159102577
#define exp6     403.428793492735
/**
  * @}
  */

/** @defgroup TSL2550_Commands
  * @{
  */
#define LightSensor_Reset        ((uint8_t)0x18)
#define LightSensor_PowerUp      ((uint8_t)0x03)
#define LightSensor_PowerDown    ((uint8_t)0x00)
/**
  * @}
  */

/** @defgroup TSL2550_ADCs
  * @{
  */
#define Light_ADC0  ((uint8_t)0x43)
#define Light_ADC1  ((uint8_t)0x83)
/**
```

```
56   * @}
57   */
58
59
60 static uint16_t LightSensor_ReadADC(uint8_t ADC);
61 uint16_t        LightSensor_Read(void);
62 void            LightSensor_Command(uint8_t Command);
63
64
65 #ifdef __cplusplus
66 }
67 #endif
68
69 #endif /* TSL2550D_H */
```

```
1  /**
2   * @file     TSL2550D.c
3   * @author   Miquel Hernàndez Nicolau
4   * @version  v1.0
5   * \~english
6   * @date     13th April 2016
7   * @brief    Library functions to use the TSL2550 sensor through I2C.
8   * @details
9   * \~catalan
10  * @date     13 d'abril de 2016
11  * @brief    Llibreria de funcions per emprar el sensor TSL2550 amb I2C.
12  * @details
13  */
14
15 #include "TSL2550D.h"
16
17
18 /**
19  * \~english
20  * @brief    It reads the selected ADC value of the sensor TSL2550D.
21  * @param    ADC: Selects the ADC that you want to read.
22  * @retval   Integer: Returns ADC's value. If an error has occurred, it returns
23  *           0xFFFF.
24  * \~catalan
25  * @brief    Llegeix el valor de llum des del sensor TSL2550D.
26  * @param    ADC: Selecciona ADC que es vol llegir.
27  * @retval   Integer: Retorna el valor de l'ADC. Si ha succeït un error
28  *           retorna 0xFFFF.
29  */
30 static uint16_t LightSensor_ReadADC(uint8_t ADC)
31 {
32     uint8_t tmp = 0, StepValue = 1, conv = 0;
33
34     LightSensor_Command(ADC);
35     __delay_ms(500);
36
37     conv = I2C_ReadChar(LIGHT_READ);
38
39     if(conv & 0x80) //Check that value is valid
40     {
41         tmp       = (conv & 0x70) >> 4;    //c: Chord number
42         StepValue = StepValue << tmp;      //2^c: Step value
```

```
43          tmp         = conv & 0x0F;          //Step number
44
45          //Return ADC value
46          return (((uint16_t)(16.5 * (StepValue - 1))) + tmp * StepValue);
47      }
48      else
49          return 0xFFFF;
50  }
51
52  /**
53   * \~english
54   * @brief   Reads the ADC values and use them to get the lighting level.
55   * @retval  Integer: Lighting in luxes. If an error has occurred it returns 0
         xFFFF.
56   * \~catalan
57   * @brief   Llegeix els ADCs i els empra per obtenir el nivell d'il·luminació.
58   * @retval  Integer: Il·luminació en luxes. Si ha hagut un error retorna 0xFFFF.
59   */
60  uint16_t LightSensor_Read(void)
61  {
62      uint16_t temp0;
63      real32_t temp1;
64
65      temp0 = LightSensor_ReadADC(Light_ADC0);
66      temp1 = LightSensor_ReadADC(Light_ADC1);
67
68      if((temp0 == 0xFFFF)||(temp1 == 0xFFFF)) //Check that value is valid
69          return 0xFFFF;
70
71      temp1 /= temp0; //temp1 = ADC1/ADC0
72
73      temp1 *= 3.13;  //temp1 = (ADC1/ADC0)*3.13
74
75      //Linealitzam exp(temp1)
76      //Aquesta instrucció ocuparia el 20% de la memòria de programa del PIC!
77      if(temp1 > 5.5)
78          temp1 = exp6 * (temp1 - 5);
79
80      else if(temp1 > 4.5)
81          temp1 = exp5 * (temp1 - 4);
82
83      else if(temp1 > 3.5)
84          temp1 = exp4 * (temp1 - 3);
85
86      else if(temp1 > 2.5)
87          temp1 = exp3 * (temp1 - 2);
88
89      else if(temp1 > 1.5)
90          temp1 = exp2 * (temp1 - 1);
91
92      else if(temp1 > 0.5)
93          temp1 *= exp1;
94
95      else
96          temp1++;
97
98      return (uint16_t)(temp0 * 0.46 / temp1);
```

```
99  }
100
101 /**
102  * \~english
103  * @brief   Sends the specified command to sensor through I2C bus.
104  * @param   command: Command to send to sensor. It can be a value of @ref
       TSL2550_Commands
105  *          or @ref TSL2550_ADCs :
106  *              @arg LightSensor_Reset
107  *              @arg LightSensor_PowerUp
108  *              @arg LightSensor_PowerDown
109  *              @arg LightSensor_ADC0
110  *              @arg LightSensor_ADC1
111  * @retval  None.
112  * \~catalan
113  * @brief   Envia l'ordre específic al sensor a través del bus I2C.
114  * @param   command: Ordre a enviar al sensor. Pot ser un valor de @ref
       TSL2550_Commands
115  *          o @ref TSL2550_ADCs :
116  *              @arg LightSensor_Reset
117  *              @arg LightSensor_PowerUp
118  *              @arg LightSensor_PowerDown
119  *              @arg LightSensor_ADC0
120  *              @arg LightSensor_ADC1
121  * @retval  Cap.
122  */
123 void LightSensor_Command(uint8_t command)
124 {
125     uint8_t Message[2];
126
127     Message[0] = LIGHT_WRITE;
128     Message[1] = command;
129     I2C_SendString(Message, 2, I2C_Start_Stop);
130 }
```

APÈNDIX B

DADES COMPLETES DE LES PROVES DELS SENSORS

B.1 Prova curta: 12 hores

Totes les dades d'aquesta taula són amb data de 07/08/2016.

Hora	TGS2600	TSL2550	SHT71	
			Humitat	Temperatura
12:07:09	27.4	268	54.6	27.6
12:22:09	15.2	170	54.9	27.6
12:37:09	9.2	167	54.2	28
12:52:09	6.8	138	54.3	27.9
13:07:36	5.3	229	54.7	28.3
13:37:11	3.6	309	52.3	28.2
13:52:01	1.8	134	51.2	28.1
14:07:37	1.4	173	49.2	28.8
14:22:11	0.4	121	48.4	28.5
14:37:02	−0.2	116	47.6	28.4
14:52:11	−0.7	101	46	28.7
15:07:02	−0.8	65	46	28.6
15:22:12	−0.8	55	45.4	28.6
15:37:04	−0.8	58	45.5	28.3
15:52:14	−1.4	54	44.3	28.6
16:07:06	−1.5	60	44	28.4
16:22:16	−1.2	62	44.9	28.6
16:37:06	−2.1	67	44	28.4
16:52:16	−2.9	63	41.9	28.8
17:07:15	−3.2	47	41.9	28.7
17:22:16	−3.7	61	39.9	28.7

Hora	TGS2600	TSL2550	SHT71	
			Humitat	Temperatura
17:37:08	−3.2	63	39.8	28.5
17:52:17	−3.9	64	39.4	28.9
18:07:08	−3.9	66	39.3	28.7
18:22:18	−4.6	64	37.6	28.8
18:37:09	−5.6	63	37.1	28.7
18:52:18	−5.6	60	36.8	28.9
19:07:10	−5.3	58	37.3	28.7
19:22:20	−5.3	49	37.6	28.7
19:37:19	−4.9	47	38.1	28.5
19:52:23	−4.6	38	37.9	28.9
20:07:29	−4.3	32	38.6	28.7
20:22:22	−3.7	39	39.6	28.4
20:37:30	8.1	50	42.1	28.1
20:52:22	12	5	42.6	28.8
21:07:13	13.1	1	43.8	28.5
21:22:23	13.3	0	44.3	28.8
21:37:15	−0.1	0	44.8	28
21:52:24	−1.4	0	44.8	27.5
22:07:16	−1.6	0	45.5	27.2
22:22:50	−1.2	0	46.1	27.2
22:37:24	0.7	0	51	27.1
22:52:15	1.1	0	53.6	27
23:07:24	1.2	0	53.8	27.1
23:22:15	0.7	0	56.9	26.1
23:37:24	1.1	0	55.9	26.3
23:52:15	−0.1	0	61.2	24

B.2 Prova llarga: 78 hores

Data	Hora	TGS2600	TSL2550	SHT71	
				Humitat	Temperatura
19/08/2016	15:08:32	62.6	78	58.5	25.4
19/08/2016	15:23:24	35	99	59.3	25.1
19/08/2016	15:38:34	24.2	108	59.6	25.2
19/08/2016	15:53:26	18.7	118	60.7	25.1
19/08/2016	16:08:36	15.7	130	60.4	25.1
19/08/2016	16:23:28	14.1	152	60.9	25
19/08/2016	16:38:38	13.1	176	60.5	25.1
19/08/2016	17:08:40	12.4	146	59.7	25.2
19/08/2016	17:53:43	15.7	153	61	27.5
19/08/2016	18:08:35	15.8	155	63.5	27.6
19/08/2016	18:23:46	15.8	155	63.8	28.1

Data	Hora	TGS2600	TSL2550	SHT71	
				Humitat	Temperatura
19/08/2016	18:38:40	15.5	156	64.8	28.1
19/08/2016	18:53:47	15.2	153	65.1	28.4
19/08/2016	19:23:48	14.7	138	74.6	28.2
19/08/2016	19:38:40	14.7	115	75.7	28.1
19/08/2016	19:53:50	14.3	76	75	28.4
19/08/2016	20:08:42	14.7	31	75.8	28.2
19/08/2016	20:23:52	15	24	74.6	28.5
19/08/2016	20:38:52	14.5	37	77.4	27.7
19/08/2016	20:53:53	14.8	11	76.5	28
19/08/2016	21:08:45	14.8	0	77.2	27.9
19/08/2016	21:23:55	14.7	0	76.5	28.1
19/08/2016	21:38:47	13.4	0	79.3	27.2
19/08/2016	21:53:57	13.7	0	78.8	27.3
19/08/2016	22:08:49	13.8	0	76.8	27.8
19/08/2016	22:23:59	13.1	0	78.2	27.3
19/08/2016	22:38:50	13.6	0	80.3	26.7
19/08/2016	22:54:00	13.7	1	77.1	27.7
19/08/2016	23:08:52	13.7	1	79	27.3
19/08/2016	23:24:02	12.9	1	76.5	27.8
19/08/2016	23:38:54	11.7	1	82.7	25.8
19/08/2016	23:54:04	12.3	1	83.8	25.6
20/08/2016	0:08:56	12.7	1	82.1	26.1
20/08/2016	0:24:06	11.9	1	82.1	25.9
20/08/2016	0:38:58	11.9	1	83.8	25.1
20/08/2016	0:54:08	13.7	0	77.6	26.9
20/08/2016	1:08:59	12.9	0	76.7	27.1
20/08/2016	1:24:09	12.6	0	75.8	27.4
20/08/2016	1:39:01	12.7	0	75.3	27.5
20/08/2016	1:54:11	12.7	0	73.9	28
20/08/2016	2:09:03	12.6	0	74.1	27.9
20/08/2016	2:24:13	11.5	0	74.1	27.9
20/08/2016	2:39:05	20.3	0	75.9	27.4
20/08/2016	2:54:15	15	0	73.3	28.2
20/08/2016	3:09:06	12.3	0	74	28
20/08/2016	3:24:16	11.3	0	76.1	27.3
20/08/2016	3:39:08	11	0	78	26.6
20/08/2016	3:54:18	11.5	0	73.7	27.9
20/08/2016	4:09:10	11	0	73.2	27.9
20/08/2016	4:24:20	10.6	0	72.7	27.9
20/08/2016	4:39:12	10.2	0	73	27.5
20/08/2016	4:54:22	10.5	0	70.6	28
20/08/2016	5:09:13	9.3	0	70.5	27.7
20/08/2016	5:39:24	8.8	0	69.1	27.7
20/08/2016	5:54:16	9.2	0	69.5	27.6

Data	Hora	TGS2600	TSL2550	SHT71	
				Humitat	Temperatura
20/08/2016	6:09:26	8.9	0	70.1	27.5
20/08/2016	6:24:18	9.9	0	70.1	27.5
20/08/2016	6:39:28	7.5	0	68.3	27.6
20/08/2016	6:54:19	5.8	0	67.5	27.2
20/08/2016	7:09:30	5.7	0	65.5	26.8
20/08/2016	7:24:21	5.1	3	65.2	26.1
20/08/2016	7:39:31	7.9	7	61.7	27.2
20/08/2016	7:54:23	4.3	26	62	27.1
20/08/2016	8:09:33	4	107	63.6	25.4
20/08/2016	8:24:34	4.3	125	61.1	26.3
20/08/2016	8:39:35	5.1	148	61.6	26.5
20/08/2016	8:54:27	7.2	146	68.3	26
20/08/2016	9:09:37	8.1	210	71.8	26.3
20/08/2016	9:24:29	8.6	474	71.9	26.6
20/08/2016	9:39:39	8.9	485	70	27.1
20/08/2016	9:54:31	8.8	491	70.5	27.1
20/08/2016	10:09:41	8.8	482	70	27.5
20/08/2016	10:24:33	9.2	402	73.2	27.3
20/08/2016	10:39:43	9.5	326	72.6	27.8
20/08/2016	10:54:35	9.1	284	74.7	27.5
20/08/2016	11:09:47	10.3	322	73.8	28.1
20/08/2016	11:24:36	11.2	369	72.7	28.3
20/08/2016	11:39:46	11.2	417	71.9	28.6
20/08/2016	11:54:38	11	449	72.1	28.5
20/08/2016	12:09:48	11.2	461	71	28.7
20/08/2016	12:24:40	11.5	473	71.1	28.6
20/08/2016	12:39:49	11.5	465	70.4	28.9
20/08/2016	12:54:41	13.8	478	70.9	28.7
20/08/2016	13:08:26	15.4	491	70.3	28.9
20/08/2016	13:23:18	15.2	476	70.4	28.8
20/08/2016	13:38:27	15	438	69.6	28.9
20/08/2016	13:53:19	12.7	385	70.7	27.2
20/08/2016	14:08:29	11.6	313	68.4	26.4
20/08/2016	14:23:21	10.7	250	67.4	25.9
20/08/2016	14:38:31	10.2	211	65.9	25.8
20/08/2016	14:53:23	12	97	65	25.9
20/08/2016	15:08:33	9.5	91	65.7	26.3
20/08/2016	15:23:25	12.2	102	66.7	26.4
20/08/2016	15:38:35	9.5	110	65.8	25.8
20/08/2016	15:53:26	8.9	119	65.1	25.4
20/08/2016	16:08:37	8.8	134	64.1	25.5
20/08/2016	16:23:28	9.5	142	63.4	25.7
20/08/2016	16:38:38	11	151	62.2	27.8
20/08/2016	16:53:30	11.7	159	63.8	27.9

Data	Hora	TGS2600	TSL2550	SHT71	
				Humitat	Temperatura
20/08/2016	17:08:40	12.2	161	63.9	28.3
20/08/2016	17:23:32	12.4	178	64.8	28.3
20/08/2016	17:38:42	12.7	181	64.4	28.6
20/08/2016	17:53:33	13.1	184	64.9	28.7
20/08/2016	18:08:43	13.7	184	64.8	28.9
20/08/2016	18:23:44	14	176	65.5	28.9
20/08/2016	18:38:45	14.5	186	65.2	29.1
20/08/2016	18:53:37	15	175	66	29
20/08/2016	19:08:47	15.1	164	65.6	29.2
20/08/2016	19:23:38	11.9	104	66.9	29.2
20/08/2016	19:38:48	10.7	131	66.9	29.4
20/08/2016	19:53:40	10.7	100	67.4	29.3
20/08/2016	20:08:50	8.2	26	66.6	29.4
20/08/2016	20:23:41	7.9	10	66.2	29
20/08/2016	20:38:51	7.4	68	65.1	29.1
20/08/2016	20:53:43	7.2	13	65.7	28.7
20/08/2016	21:08:53	7.9	0	66.9	28.8
20/08/2016	21:23:45	7.9	0	69.7	28.3
20/08/2016	21:38:55	7.9	0	69	28.6
20/08/2016	21:53:46	9.5	0	69.6	28.4
20/08/2016	22:08:56	8.9	0	70.9	28.4
20/08/2016	22:23:48	8.6	0	72.7	28.1
20/08/2016	22:38:58	8.8	0	72.3	28.4
20/08/2016	22:53:50	8.6	0	74.4	27.7
20/08/2016	23:09:00	8.1	1	73.8	27.8
20/08/2016	23:23:51	7.7	1	73.8	27.6
20/08/2016	23:39:01	7.5	1	74.1	27.4
20/08/2016	23:53:53	6.5	1	72.7	27.5
21/08/2016	0:09:03	6.5	1	71.1	27.6
21/08/2016	0:24:12	6	1	71.2	27.3
21/08/2016	0:38:56	5.7	1	72	26.9
21/08/2016	0:54:06	6	0	69.5	27.7
21/08/2016	1:24:08	6.5	0	69.1	28
21/08/2016	1:39:00	6.3	0	69.9	27.6
21/08/2016	1:54:10	5.8	0	69.2	27.6
21/08/2016	2:09:01	5.6	0	69	27.4
21/08/2016	2:24:11	6.3	0	68.6	27.5
21/08/2016	2:39:03	5.6	0	69.2	27.3
21/08/2016	2:54:13	5.1	0	68.7	27.3
21/08/2016	3:09:05	5.1	0	68.6	27.2
21/08/2016	3:24:15	4.9	0	68.2	27.1
21/08/2016	3:39:07	4.7	0	67.8	27.1
21/08/2016	3:54:17	4	0	66.8	27.1
21/08/2016	4:09:09	3.9	0	66.8	26.8

Data	Hora	TGS2600	TSL2550	SHT71	
				Humitat	Temperatura
21/08/2016	4:24:19	4	0	65.5	27.1
21/08/2016	4:39:11	4	0	65.5	27.1
21/08/2016	4:54:21	4.6	0	65.2	27.2
21/08/2016	5:09:12	4.6	0	65.3	27.1
21/08/2016	5:24:23	3.9	0	64.8	27.1
21/08/2016	5:39:14	3.7	0	64.5	27
21/08/2016	5:54:24	3.5	0	63.5	27.1
21/08/2016	6:09:16	3.3	0	63.4	27
21/08/2016	6:24:26	4.2	0	62.7	27.2
21/08/2016	6:39:18	4.2	0	62.9	27.1
21/08/2016	6:54:28	4	0	62.4	27.2
21/08/2016	7:09:20	3.7	1	62.6	27.1
21/08/2016	7:24:30	3.6	7	61.3	27.5
21/08/2016	7:39:22	3.3	14	60.6	27.5
21/08/2016	7:54:32	3	21	60.8	27
21/08/2016	8:09:24	2.9	26	60.6	27
21/08/2016	8:24:34	2.5	31	60	27.2
21/08/2016	8:39:26	2.3	36	59.3	27.2
21/08/2016	8:54:36	2.1	38	59	27.2
21/08/2016	9:09:28	1.8	42	58.3	27.3
21/08/2016	9:24:38	1.1	43	56.5	27.4
21/08/2016	9:39:30	0.5	48	56.5	27.3
21/08/2016	9:54:40	0.2	50	56.1	27.2
21/08/2016	10:09:32	0.2	52	55.9	27.2
21/08/2016	10:24:42	0.1	55	55.6	27.3
21/08/2016	10:39:34	−0.2	57	54.4	27.2
21/08/2016	10:54:44	−0.4	60	53.1	27.4
21/08/2016	11:09:35	−0.4	63	52.6	27.7
21/08/2016	11:24:46	0.1	69	52.7	27.7
21/08/2016	11:39:37	1.1	74	52.9	27.6
21/08/2016	11:54:48	1.9	75	52.9	27.9
21/08/2016	12:09:39	0.2	108	52.4	27.7
21/08/2016	12:24:49	−0.1	97	51.7	27.8
21/08/2016	12:39:41	−0.1	105	52.2	28
21/08/2016	12:54:51	0.1	106	51.7	28.1
21/08/2016	13:09:43	0.1	103	52.1	28.1
21/08/2016	13:24:53	0.8	102	51.8	28.4
21/08/2016	13:39:45	2.6	106	52.3	28.3
21/08/2016	13:54:55	1.6	86	54.7	26.3
21/08/2016	14:09:47	1.8	89	54.5	25.3
21/08/2016	14:24:58	2.1	99	54	25
21/08/2016	14:39:49	2.6	110	55.2	24.9
21/08/2016	14:55:00	2.5	126	55	24.8
21/08/2016	15:09:52	2.5	154	55	24.7

Data	Hora	TGS2600	TSL2550	SHT71	
				Humitat	Temperatura
21/08/2016	15:25:02	3	177	54.8	24.7
21/08/2016	15:39:54	2.8	197	53.8	25
21/08/2016	15:55:04	2.6	161	48.7	26.7
21/08/2016	16:09:56	2.5	259	48.4	26.6
21/08/2016	16:25:06	2.6	224	47.4	26.9
21/08/2016	16:39:58	3.2	189	48.8	26.7
21/08/2016	16:54:43	2.9	113	47.8	27.1
21/08/2016	17:09:35	2.9	121	48.3	26.7
21/08/2016	17:24:45	2.8	167	47.3	27.2
21/08/2016	17:39:37	4.2	214	48.3	27.1
21/08/2016	17:54:47	6.1	149	50.5	28
21/08/2016	18:09:39	7	126	53.4	28.2
21/08/2016	18:24:49	5.7	170	54.5	28.6
21/08/2016	18:39:41	4.6	208	57.2	28.5
21/08/2016	18:54:51	4.7	209	57.5	28.9
21/08/2016	19:09:43	4	178	58.9	28.6
21/08/2016	19:24:53	3.7	168	58.5	28.6
21/08/2016	19:39:45	3.5	136	59.1	28.4
21/08/2016	19:54:55	3.5	97	58.7	28.5
21/08/2016	20:09:47	3.2	26	59	28.2
21/08/2016	20:24:57	1.9	71	61.4	27.3
21/08/2016	20:39:49	2.1	85	66.2	25.9
21/08/2016	20:54:59	1.9	18	66	26.2
21/08/2016	21:09:51	2.3	0	66.8	25.9
21/08/2016	21:25:01	1.9	0	67.8	25.5
21/08/2016	21:39:53	1.9	0	67.7	25.6
21/08/2016	21:55:04	2.3	1	66.1	26.2
21/08/2016	22:09:56	1.6	0	65.6	25.9
21/08/2016	22:25:06	1.8	0	63.6	26.1
21/08/2016	22:39:58	1.8	0	63.1	26.3
21/08/2016	22:55:08	1.6	0	63.7	25.9
21/08/2016	23:10:00	2.1	1	62.8	26.4
21/08/2016	23:25:10	1.9	1	62.6	26.5
21/08/2016	23:40:02	1.8	1	63.1	26.3
21/08/2016	23:55:12	1.9	1	63	26.3
22/08/2016	0:10:04	1.8	1	63.6	26.1
22/08/2016	0:25:15	1.8	1	63.6	26.1
22/08/2016	0:40:07	54.9	0	62.3	26.7
22/08/2016	0:55:17	25.1	0	60.1	27.5
22/08/2016	1:10:09	10.6	0	60.5	27.4
22/08/2016	1:25:28	7.5	0	59.9	27.6
22/08/2016	1:40:19	6.1	0	60.3	27.5
22/08/2016	1:55:21	5.4	0	59.7	27.7
22/08/2016	2:10:13	4.3	0	61.4	27

Data	Hora	TGS2600	TSL2550	SHT71	
				Humitat	Temperatura
22/08/2016	2:25:23	4.3	0	59.9	27.3
22/08/2016	2:40:15	3.7	0	59.7	27.4
22/08/2016	2:55:25	3.3	0	59	27.6
22/08/2016	3:10:17	3.3	0	59.5	27.4
22/08/2016	3:25:27	2.8	0	59.9	27.1
22/08/2016	3:40:28	2.5	0	59.4	27.3
22/08/2016	3:55:29	2.3	0	58.7	27.5
22/08/2016	4:10:21	2.5	0	59.3	27.3
22/08/2016	4:25:31	2.5	0	58.6	27.5
22/08/2016	4:40:23	2.1	0	59.3	27.1
22/08/2016	4:55:33	1.8	0	58.6	27.2
22/08/2016	5:10:34	1.9	0	60	26.6
22/08/2016	5:25:36	1.9	0	58.4	27.2
22/08/2016	5:40:36	1.9	0	59	27
22/08/2016	5:55:38	2.1	0	58.5	27.2
22/08/2016	6:10:30	2.1	0	59.4	27
22/08/2016	6:25:40	2.2	0	58.9	27.2
22/08/2016	6:40:41	2.6	0	59.8	27
22/08/2016	6:55:42	3	0	60.2	27
22/08/2016	7:10:34	2.2	1	64.4	25.4
22/08/2016	7:25:44	1.6	4	61.6	26.7
22/08/2016	7:40:36	4.9	8	61.9	26.6
22/08/2016	7:55:47	3.7	15	62.5	26.5
22/08/2016	8:10:39	3.3	20	63.9	26.4
22/08/2016	8:25:49	4.4	223	64	26.5
22/08/2016	8:40:41	4.9	279	64.7	26.4
22/08/2016	8:55:51	4.7	286	65.1	26.4
22/08/2016	9:10:43	3.9	307	65.3	26.4
22/08/2016	9:25:53	3.7	336	64.8	26.5
22/08/2016	9:40:19	1.8	337	64.5	26.3
22/08/2016	9:55:30	1.5	345	64.7	26.3
22/08/2016	10:10:22	2.5	399	64.5	25.9
22/08/2016	10:25:32	2.1	298	65.3	26.4
22/08/2016	10:40:33	2.2	315	66.1	26.1
22/08/2016	10:55:34	2.3	173	65.3	26.6
22/08/2016	11:10:26	2.9	220	66.3	26.7
22/08/2016	11:25:36	3.2	145	64.3	27.3
22/08/2016	11:40:28	2.8	81	63.5	27.3
22/08/2016	11:55:38	85.2	136	62.4	27.6
22/08/2016	12:10:30	79.9	111	62.5	27.8
22/08/2016	12:25:40	67.3	147	62.6	28.2
22/08/2016	12:40:41	41.3	130	62.5	27.9
22/08/2016	12:55:42	29	109	61.6	28.1
22/08/2016	13:10:34	3.2	121	61.6	27.7

Data	Hora	TGS2600	TSL2550	SHT71	
				Humitat	Temperatura
22/08/2016	13:25:44	2.9	179	61.3	27.9
22/08/2016	13:40:45	2.6	283	61.5	27.7
22/08/2016	13:55:46	1.9	200	62.6	27.4
22/08/2016	14:10:38	8.8	147	62.9	27.6
22/08/2016	14:25:48	5.3	173	61.7	28
22/08/2016	14:40:40	4	127	61.6	28
22/08/2016	14:55:50	2.8	93	60.3	28.1
22/08/2016	15:10:42	6.4	105	60.3	28.1
22/08/2016	15:25:52	4.2	116	59.7	28.2
22/08/2016	15:40:44	3.3	129	60	28.3
22/08/2016	15:55:54	2.8	143	59.9	28.5
22/08/2016	16:10:46	4.3	152	60.4	28.4
22/08/2016	16:25:56	4.9	163	60.6	28.6
22/08/2016	16:40:48	2.9	177	61.5	28.5
22/08/2016	16:55:58	4.3	194	61.3	28.9
22/08/2016	17:10:50	4.9	194	61.5	28.9
22/08/2016	17:26:00	5.1	205	61.5	29
22/08/2016	17:40:52	4.4	220	62	28.9
22/08/2016	17:56:02	3.7	212	61.1	29.2
22/08/2016	18:10:53	3.5	219	61.7	28.9
22/08/2016	18:26:03	4	219	61	29.2
22/08/2016	18:40:55	4.2	219	61.4	29.1
22/08/2016	18:56:05	4	205	61.2	29.1
22/08/2016	19:10:57	3	216	61.5	29
22/08/2016	19:26:07	2.3	212	63.8	28
22/08/2016	19:40:59	2.1	182	64.5	27.7
22/08/2016	19:56:09	1.6	125	64	27.6
22/08/2016	20:11:09	1.4	24	65.7	27
22/08/2016	20:26:11	1.2	15	64.6	27.1
22/08/2016	20:41:11	1.6	6	65	26.7
22/08/2016	20:56:13	1.1	0	65.6	26.7
22/08/2016	21:10:43	2.6	1	64.7	26.8

APÈNDIX C

GUIA D'UTILITZACIÓ DEL TRANSMISSOR XBEE AMB SENTILO

Aquesta guia pretèn ser un suport per a iniciar-se a Sentilo. Però no s'ha de prendre com quelcom estàtic, ja que cal tenir en compte que el funcionament final depèn de cada ordinador, versió de MV, seguritat de la xarxa, tipus de transmissor, etc. Els següents paràmetres han funcionat correctament per a Sentilo 1.5.0 virtual machine, VirtualBox 5.0.22, xBee Wi-Fi i dins una xarxa local amb accés a la configuració de l'encaminador.

C.1 MV de Sentilo

- Emprar-la amb la versió de VirtualBox recomanada a la web.
- Dins VirtualBox configurar xarxa del Sentilo com adaptador pont, mode promiscu denegat. Hauria de venir així per defecte.
- És important apagar la màquina virtual correctament i anar guardant punts de restauració. Si l'ordinador es penja la MV quedarà inservible i s'haurà de reinstal·lar al VirtualBox.
- Arrencar la MV i iniciar sessió al Linux amb usuari i contrasenya 'sentilo'.
- Es pot conèixer la IP de Sentilo escrivint 'ifconfig' a la consola de la MV.
- Entrar a l'interfície gràfica de Sentilo escrivint al navegador http://[IP de la MV]:8080 (p.e. http://192.168.1.100:8080)
- Ja dins Sentilo, entrar a l'usuari 'admin' amb la contrasenya '1234'.
- Editar l'usuari 'admin' i deixar-lo només amb privilegis d'administrador.

- (Opcional) Configurar el punt del mapa on s'obre Sentilo (Si tenim un sol node interessa que el mapa ja s'obre directament a damunt d'ell).
- Si no existeixen, crear al sistema els tipus de sensor que es necessitin. (A aquest projecte es crearen il·luminació i contaminació).
- Crear un provider, al qual la MV li associarà una apiKey.
- Crear un component, que pertanyerà a un provider. S'ha de situar al mapa.
- Crear diferents sensors i associar-los al component.

C.2 Encaminador/Router

- (Opcional) Crear IP estàtiques per a les MAC de la MV i del xBee, així a cada sessió es tendran les mateixes IP i facilitarà el treball.

C.3 XBee

Els següents paràmetres s'han de configurar al xBee des de XCTU o des de microcontrolador:

- AH: Infrastructure [2]
- CE: STA mode [2]
- IP: TCP [1]
- DO: 0
- DL: [IP de la MV] (Si s'ha configurat estàtica només s'haurà de modificar una vegada aquest paràmetre)
- DE: 1F91 (Nombre del port 8081 en hexadecimal.)
- AP: Transparent Mode [0]

C.4 Tallafocs

Obrir port 8081 al tallafocs i realitzar un enviament de prova. Si no ha funcionat, desactivar el tallafocs i tornar-ho a provar. Si segueix sense funcionar és possible que el problema sigui a un altre lloc.

C.5 Trames

Per a dur a terme els enviaments es poden trobar alguns tutorials i llibreries que faciliten la feina. S'hi pot trobar codi per Arduino, Java i Raspberry Pi.[17]
La introducció de dades s'ha de fer mitjançant una petició PUT a la direcció, on s'ha d'especificar l'apiKey, la IP de la MV, la llargària de la trama de dades i, a continuació,

la pròpia trama de dades. Per al present projecte, en cas d'un sol sensor, el que s'hauria d'enviar pel xBee seria el següent:

PUT /data/casa/sensor HTTP/1.1
identity_key: cc2a7b4bfbc4799d93759da354860eb6[...]
Host: 192.168.1.100
Connection: close
Content-Length: 33
Content-Type: application/json

{"observations":[{"value":"20"}]}

La transmissió de les dades es pot fer sensor a sensor o amb varis sensors a la vegada. Per a un sol sensor s'ha d'enviar especificant a la direcció quin és (p.e. /data/casa/TSL2550), i amb una trama amb el següent format:

{"observations":[{"value":"9.6","timestamp":"17/02/2016T11:43:45CET","location":"41.3888 2.15899"}]}

Cal tenir en compte que 'timestamp' i 'location' són paràmetres opcionals. I de fet, 'location' és bastant innecessari si el node no s'ha de moure.

Per a transmetre vàries dades d'un sensor:

{"observations":[{"value":"10.1"},{"value":"11.2","timestamp":"17/09/2012T12:34:45"}, {"value":"12.3","timestamp":"17/09/2012T10:34:45"}]}

Per transmetre dades de varis sensors. No caldria especificar cap sensor a la direcció de la petició PUT (p.e. /data/casa), i el format de trama seria:

{"sensors":[{"sensor":"RE0012","observations":[{"value":"1.1"},{"value":"1.2","timestamp":"17/09/2012T12:34:45CET"}]},{"sensor":"RE0013","location":"41.12345 2.12354","observations":[{"value":"2.1"}]}]}

BIBLIOGRAFIA

[1] Microchip. Pic16f886. [Online]. Available: http://www.microchip.com/wwwproducts/en/PIC16F886 (document), 3.2.1

[2] DIGI. Xbee wi-fi. [Online]. Available: http://www.digi.com/products/xbee-rf-solutions/modules/xbee-wi-fi (document), 3.2.2

[3] Sensirion. Digital humidity sensor sht7x (rh/t). [Online]. Available: https://www.sensirion.com/en/products/humidity-sensors/pintype-digital-humidity-sensors/ (document), 3.2.3, 4.1.3

[4] TAOS/AMS. Tsl2550 ambient light sensor. [Online]. Available: http://ams.com/eng/Products/Light-Sensors/Ambient-Light-Sensors/TSL2550 (document), 3.2.4

[5] Figaro. Tgs2600. [Online]. Available: http://www.figaro.co.jp/en/product/entry/tgs2600.html (document), 3.2.5

[6] ONSemiconductor. Ncp1117. (Model exacte: NCP1117ST33T3G). [Online]. Available: http://www.onsemi.com/PowerSolutions/product.do?id=NCP1117 (document), 3.2.6

[7] Micrel/Microchip. Mic5225. (Model exacte: MIC5225-5.0YM5). [Online]. Available: http://www.microchip.com/wwwproducts/en/MIC5225 (document), 3.2.7

[8] Sentilo. What is. [Online]. Available: http://www.sentilo.io/xwiki/bin/view/Sentilo.About.Product/Whatis 2.3.2

[9] Eagle pcb design. [Online]. Available: https://cadsoft.io/ 3.3

[10] Microchip. Usign mplab icd 2. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/51265e.pdf 4.2, 4.1.1

[11] Desconegut. I2c implementation on pic 16f877 microcontroller. [Online]. Available: http://www.8051projects.net/wiki/I2C_Implementation_on_PIC 4.1.3

[12] L. G. (ElectroSome). Using uart of pic microcontroller – mplab xc8. [Online]. Available: https://electrosome.com/uart-pic-microcontroller-mplab-xc8/ 4.1.3

[13] Sentilo. Arduino client. [Online]. Available: http://www.sentilo.io/xwiki/bin/view/Sentilo.Community.Tutorials/Arduino+Client 4.1.3, 4.1.3

[14] ——. Use a virtual machine. [Online]. Available: http://www.sentilo.io/xwiki/bin/view/Sentilo.Community.Documentation/Use+a+Virtual+Machine 4.3.1

[15] Postman. Postman. [Online]. Available: https://www.getpostman.com/ 4.3.3

[16] Sentilo. Retrieve sensor observations. [Online]. Available: http://www.sentilo.io/xwiki/bin/view/ApiDocs.Services.Data/RetrieveSensorData 4.3.4

[17] ——. Tutorials. [Online]. Available: http://www.sentilo.io/xwiki/bin/view/Sentilo.Community.Tutorials/Tutorials C.5