Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE DE FI DE CARRERA

TÍTOL DEL PFC: Disseny d'una xarxa de sensors acústics i ambientals pel projecte SIAC

TITULACIÓ: Enginyeria de Telecomunicació (segon cicle)

AUTOR: Josep Cardona Fernández

DIRECTOR: Adán Amor Garriga Torres

SUPERVISOR: Rafael Vidal Ferré

DATA: 18 d'abril de 2012

Títol: Disseny d'una xarxa de sensors acústics i ambientals pel projecte SIAC

Autor: Josep Cardona Fernández

Director: Adán Amor Garriga Torres

Supervisor: Rafael Vidal Ferré

Data: 18 d'abril de 2012

Resum

En aquest projecte s'han estudiat les capacitats de la plataforma Arduino pel que fa a processat i comunicació de dades entre nodes. Això s'ha fet per poder dissenyar xarxes de sensor sense fils que siguin capaces d'adquirir dades acústiques, processar-les i enviar-les a un servidor central.

En primer lloc s'han comparat dues plataformes hardware per crear la WSN. Hem escollit la plataforma Arduino per ser de desenvolupament lliure i per tot el suport que es pot trobar tant a la xarxa com en publicacions en paper.

Com a plataforma per a comunicacions, al escollir Arduino hem hagut de treballar amb Xbee, que permet crear xarxes mesh, de forma que les dades poden viatjar per múltiples rutes.

En segon lloc s'han estudiat les característiques dels micròfons com a sensors per a la captació acústica.

Seguidament, i com a part del sistema d'acondicionament del senyal, s'han analitzat les capacitats de diferents amplificadors operacionals per adaptar la sortida del micròfon a l'entrada del conversor A/D del Arduino, escollint un específic per àudio.

En tercer lloc, s'han pogut implementar rutines per calcular la FFT. Aquesta FFT és de 128 punts, amb dades en coma flotant. Amb aquesta resolució som capaços de realitzar la FFT i demés tasques pel processat en un sol node.

També s'han implementat les corbes del A Weighting pel que fa a mesures de soroll ambient, ja que són les més emprades pels aparells comercials de captació de dades acústiques.

Seguidament, s'ha verificat que amb els mòduls Xbee podem enviar/rebre les dades de la FFT amb total normalitat. En aquest projecte també s'han provat diferents configuracions en quan a la manera que es pot fer la interacció entre Arduino i Xbee. Es poden identificar paquets i per tant nodes, i considerar el càlcul distribuït.

Finalment, s'ha conclòs que podem emular les prestacions dels sonòmetres comercials, mitjançant una xarxa de sensors sense fils basada en Arduino i Xbee.

Title: Design of an acoustic and environmental sensor network for SIAC

Author: Josep Cardona Fernández

Director: Adán Amor Garriga Torres

Supervisor: Rafael Vidal Ferré

Date: April, 18th 2012

Overview

In this project the capabilities regarding signal processing and communications between nodes of the Arduino platform have been studied. This has been done in order to design wireless sensor networks able to acquire acoustic data, process them and send them to a central server.

Firstly, two hardware platforms have been compared to create the wireless sensor network. Arduino platform has been chosen because it is open source and because of the amount of support, either on the Internet or in paper, it can be found.

Due to the Arduino, the communications platform that it has been chosen is Xbee, which allows mesh networking so the data can travel through multiple routes.

Secondly, electrec microphones specifications have been studied as a way of acquiring acoustic data.

Furthermore, and as part of the signal conditioning system, the specifications of different operational amplifiers have been analyzed to adapt the output of the microphone to the Arduino A/D input.

In third place, FFT routines have been implemented on the Arduino. This is a 128-point FFT, with floating data types. With this FFT resolution the system is able to process the data within a single node.

A Weighting curves have been also implemented, because they are the standard when measuring ambient or industrial noise, and they are widely implemented in many sound pressure level meters.

Moreover, it has been checked that with Xbee modules FFT data can be correctly sent or received by other nodes. In this project different ways as far as interaction between Arduino and Xbee is concerned have also been studied. Nodes, and therefore packets, can be identified making possible distributed calculations.

Finally, it has been concluded that the system designed can simulate the performance of current sound pressure level meters, with a wireless sensor network formed by Arduino and Xbee.

ÍNDEX

ÍNDEX DE FIGURES

ÍNDEX DE TAULES

CAPÍTOL 1. INTRODUCCIÓ

En els nostres temps adquirir informació s'ha convertit en una tasca fonamental i bàsica per multitud de processos i activitats a la nostra vida quotidiana. La informació adquirida pot servir per a controlar la temperatura en un recinte, pel control de diferents processos en una fàbrica, crear mapes de soroll o mesurar la concentració de gasos en un espai.

La interfície bàsica per interactuar amb les magnituds físiques són els sensors, que dotats del sistemes pertinents en quant a processat del senyal i comunicacions, poden formar xarxes molt diverses.

Un grup d'especial interès dins aquestes xarxes són les que no fan ús de fils per a transmetre o rebre informació. Això permet reduir els costos d'instal·lació i en facilita el seu desplegament.

Amb aquesta filosofia es planteja el present Projecte Final de Carrera, amb col·laboració amb el Centre Internacional de Mètodes Numèrics per a Enginyeria (CIMNE), que pretén estudiar les possibilitats en quant a processat i comunicació de la informació de les xarxes sense fils.

Això es farà tenint en ment una aplicació concreta plantejada des de CIMNE: crear mapes de soroll d'una zona determinada mitjançant la captació de dades acústiques. Aquesta captació de dades es farà de dos formes: amb sonòmetres i amb una xarxa sense fils composta per sensors acústics (micròfons).

Així doncs, en el Capítol 2 fem la proposta del projecte, introduïm els seus objectius i els situem en el context del Sistema d'Informació Acústica. També expliquem breument el per què de la motivació a l'hora d'adquirir informació amb xarxes sense fils.

El Capítol 3 explica amb cert grau de detall la justificació tècnica del projecte. Aquest capítol està dividit en tres parts: una primera dedicada a la plataforma emprada per a la creació de la xarxa de sensors sense fils; una segona a on s'expliquen les particularitats dels elements de la xarxa en quant a processat de dades; i una tercera que explica aspectes relacionats amb la comunicació de les dades i els dispositius emprats per tal tasca.

Finalment, el Capítol 4 és un recull de resultats experimentals per validar el funcionament de la xarxa de sensors sense fils, i per validar que pot servir com a sistema de captació de dades acústiques. Aquest capítol primer presenta els resultats pel sistema de processat de dades i després les proves fetes per entendre les capacitats en quant a comunicació que ofereixen els dispositius escollits.

Amb això, es tindrà una referència per a futurs desplegaments de xarxes de sensors sense fils en l'àmbit en que s'emmarca aquest projecte. Per exemple, si una plataforma hardware determinada pot emular les capacitats dels sonòmetres comercials.

CAPÍTOL 2. PROPOSTA

Des de CIMNE es va plantejar la necessitat de crear un sistema d'informació acústica per tenir controlats els nivells de soroll en una zona determinada. Aquest sistema hauria d'estar composat per una xarxa de sonòmetres i una de sensors de captació acústica (micròfons). La xarxa de sonòmetres és reduïda pel seu cost elevat amb lo que no es cobreixen grans extensions de territori. Per això s'afegeix una altra xarxa sense fils de suport formada per micròfons.

El primer apartat 2.1 anomena els objectius concrets del Projecte. El segon apartat 2.2 explica la motivació pel *Sistema d'Informació Acústica*, SIAC. S'expliquen les parts que el formen i la problemàtica que vol resoldre. En el mateix apartat, s'introdueix la motivació per l'ús de xarxes de sensors sense fils.

2.1. Objectiu del Projecte

L'objectiu específic d'aquest Projecte Final de Carrera és dissenyar la xarxa d'adquisició de dades acústiques, seguint el model d'una xarxa sense fils i com a suport a la xarxa de sonòmetres del SIAC. No s'integren altres sensors per captar altres variables ambientals, només nivells de soroll.

El disseny s'ha centrat en la realització de les següents fites:

- Captació de dades acústiques amb micròfons electret. Condicionament del senyal.
- Processat digital del senyal. Rutines per calcular la Transformada de Fourier del senyal. Altres rutines per realitzar filtrats digitals.
- Consideració del càlcul distribuït entre diversos nodes.
- Enviament de les dades capturades entre nodes o fins un destí final.
- Avaluació bàsica de les capacitats en quant a comunicació sense fils de la plataforma hardware emprada.

En aquest PFC s'ha fet especial esment en l'estudi de les capacitats del hardware emprat per emular les prestacions dels sonòmetres comercials.

Donat que la WSN es dissenya pel SIAC, a continuació explicarem les generalitats del mateix per entendre quin lloc ocupa la WSN dissenyada.

2.2. Àmbit SIAC

Com el seu propi nom indica, SIAC vol permetre la monitorització acústica de determinades zones geogràfiques, en concret, les zones turístiques i residencials de les Illes Pitiüses.

SIAC està format per tres parts: sistema d'adquisició de dades acústiques (juntament amb altres dades de caràcter ambiental i que completen les acústiques: força i direcció del vent, humitat i temperatura); base de dades acústica i sistema d'informació geogràfica (GIS en anglès).

El sistema d'adquisició de dades està format per sonòmetres i un conjunt de sensors, repartits per un territori, per captar diferents aspectes mediambientals: soroll, temperatura, humitat i força i direcció del vent. Aquestes dades són processades i enviades sense fils a un servidor central. Cal dir, que el processat de les mateixes (analògic, filtrat digital, etc.) es realitza al sensor – plataforma de processat.

Les bases de dades recullen la informació enviada pel sistema d'adquisició de dades, i serveixen de suport per els següents elements del sistema. En una primera versió del projecte no es planteja que al servidor de bases de dades es realitzi cap càlcul o processat.

Finalment, aquesta informació es serveix per crear motors de simulació per a mapes acústics i estudiar el impacte de les activitats humanes, en quant a contaminació acústica es refereix, en una zona determinada. La informació de les bases de dades permetrà que s'integrin en altres sistemes de representació de la informació obtinguda (mapes en 3D per exemple).

El conjunt dels tres sistemes permet obtenir informació acústica detallada d'una zona geogràfica en concret. Aquesta informació pot ser d'utilitat per a les Administracions Públiques ja que han de controlar els nivells de soroll[1]. I no només per controlar els nivells de soroll, SIAC també permetrà:

- Estudis sobre el impacte del turisme.
- Creació d'una base de dades acústica, amb un futur suport altres bases de dades ambientals més extenses.
- Creació de mapes en 3D sobre els que analitzar els impactes acústics d'activitats com el turisme, construcció o altres indústries.

Amb tot, SIAC pretén monitoritzar i mostrar informació acústica d'interès per a la ciutadania. També pretén estalviar costos a les AAPP a l'hora de contractar estudis acústics.

2.2.1. Xarxes sense fils com a plataforma per a la captació de dades.

Les WSNs no només són fàcils de desplegar. Al emprar comunicacions sense fils per transmetre informació entre nodes, o entre un servidor central, també es poden configurar de forma remota.

En qualsevol WSN, el primer estadi és la captació de dades a través de sensors (un o varis). Com que aquestes dades es solen digitalitzar, s'han desenvolupat plataformes pel processat de les dades dels sensors i la seva transmissió. Aquestes plataformes permeten una complexitat major o menor de càlcul, fent que les dades entregades per la WSN no siguin simples bits, si no que tinguin algun significat [1].

Després dels estadis de captació i processat, les dades s'envien a un altre node per a fer el càlcul de manera distribuïda, o bé a un servidor central. Els elements de comunicació solen operar en bandes lliures (ISM), i depenen de l'aplicació, les plataformes de processat es poden 'expandir' per incorporar comunicacions en bandes llicenciades.

[1] European Noise Directive 2002/49/EC i el Real Decreto 1367/2007, de 19 de octubre.

Com que el SIAC requereix d'informació acústica (i ambiental) de diversos punts, l'ús de les WSNs està més que justificat, en el sentit que abarateixen els costos de desplegament i manteniment, i ofereixen capacitats de càlcul que fan que enviïn dades llestes per ser interpretades pels altres estadis del SIAC, com el GIS.

2.2.2. Introducció a les WSNs

Segons [1] i [2], una WSN és un conjunt de nodes capaços d'interactuar amb el seu entorn, mesurant magnituds físiques, i que col·laboren entre ells per aconseguir-ho. L'activador d'aquesta col·laboració són els elements de transmissió d'informació sense fils.

En aquesta definició, el terme node fa referència al conjunt d'elements que permeten captar magnituds físiques, transformar-les en senyals digitals o analògiques (tot i que avui en dia és sol fer una associació automàtica entre WSN i senyal digital) i enviar sense fils aquestes senyals, cap un altre node o cap un destí final.

La figura 2-1 mostra un diagrama de blocs d'un node en general, a on es suposa que la senyal a enviar és digital.

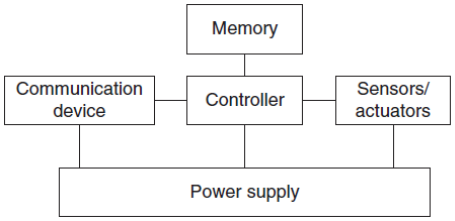


Figura 2-1 Diagrama de blocs general d'un node [1]

Segons [1], la descripció de cada un dels elements és:

- Controlador: processa les dades i és capaç d'executar codis.
- Memòria: emmagatzema, de forma temporal o no, informació rellevant ja siguin dades captades pels sensors o altres dades necessàries pel processat de la informació.
- Sensors/actuadors: són la interfície entre el node i la magnitud física a mesurar. Transformen aquesta magnitud física en senyal que és tractable pels elements posteriors en el processat de les mateixes (controladors, amplificadors de senyal, etc.)
- Element de comunicació: és el que fa que una xarxa sigui inalàmbrica. És un transmissor/receptor capaç d'enviar i rebre dades.
- Font d'alimentació: és l'element que permet que el sistema obtingui l'energia per realitzar les operacions necessàries.

En general, les capacitats de processat dels nodes són limitades degut a les mides reduïdes dels mateixos [1]. No obstant això, la potència de càlcul de les WSNs no la dóna un sol node, si no la suma de tots els que la formen.
Els requeriments que governen el disseny de qualsevol WSN depenen del objectiu que es pretén aconseguir amb la mateixa. De forma general, i seguint la lògica que trobem en [1], els requeriments per qualsevol WSN són:

- Tipus de servei: aquí fa referència a que les WSNs han d'aportar informació amb sentit pel qui la rep, no només bits.
- Qualitat de Servei: pot ser molt diferent, ja que per exemple si hem de controlar en temps real un sensor necessitem una latència baixa. Per contra, si hem de fer mesures cada quinze minuts perquè és el temps de variació de la variable, no cal ser molt restrictiu amb els temps de latència o velocitat de la xarxa.
- Tolerable a errades: pot ser que un node quedi inutilitzat, destruint una ruta de comunicació entre nodes. Això s'ha de preveure i actuar en conseqüència (com tenir la suficient densitat de nodes per disposar de rutes alternatives).
- Temps de vida: s'ha d'intentar maximitzar el temps de vida de la xarxa, mitjançant la captació d'energia pel funcionament del sistema per exemple.
- Escalabilitat: hem de preveure si l'aplicació per a la qual es dissenya la xarxa requerirà d'un augment en nombre de nodes de la mateixa. Hem de preveure cert espai per al creixement de la xarxa sense que afecti al rendiment de la mateixa.
- Densitat de nodes: pot variar segons l'aplicació, i dins la mateixa aplicació segons la posició dels nodes.
- Reprogramació: els nodes han de ser capaços d'acceptar una reprogramació de les seves tasques, modificant el codi que utilitzen si és necessari.
- Manteniment: la xarxa ha de mantenir-se a ella mateixa, i ha de monitoritzar la seva pròpia salut adaptant-se als canvis (rutes caigudes per exemple).

Com es pot entendre, els tipus de WSNs poden ser molt variats. Podem tenir xarxes molt simples amb un nombre reduït de nodes, que únicament es dediquin a recollir dades i enviar-les *en brut*.

Afegir nodes augmenta la complexitat de la xarxa, però el motiu del seu augment pot ser divers: des d'incloure nodes per transmetre dades a distàncies llargues (o més llargues que la cobertura d'un sol node); fins a crear xarxes amb una quantitat de nodes més nombrosa, per captar la major quantitat de successos possibles dins un entorn molt variant en el temps.

També podem augmentar el nombre de nodes per distribuir el processat del senyal. Anteriorment hem comentat que les capacitats de càlcul dels nodes són limitades, sovint per raons de consum energètic. Distribuint el càlcul entre diferents nodes augmentem la capacitat de càlcul de la xarxa. Per exemple, si hem de fer una *Fast Fourier Transform* i aplicar més endavant dues etapes de filtrat digital; pot ser que un sol node no tingui memòria per realitzar totes les operacions. En canvi, si un node fa la FFT i altres dos nodes fan el filtrat, augmentem la capacitat de càlcul de la xarxa.

Dins del disseny de la xarxa sense fils, el consum energètic és un factor limitant a tenir en compte. Els nodes han de tenir alimentació autònoma, d'altre banda no estaríem desplegant una WSN, el que implica emprar bateries o altres sistemes d'alimentació.

L'ús de bateries condiciona el disseny de la WSN, ja que per exemple, si havíem pensat en una xarxa per captar dades en temps real en un entorn remot, ens trobarem amb que les bateries es poden esgotar aviat. Això farà que pensem en introduir plaques solars, però pot ser que no donin suficient energia el sistema.

En [1] es tenen més exemples amb diferents casuístiques, així com explicacions molt més detallades sobre els requisits de disseny de les WSNs. Tot i així, i com a petit resum d'aquesta secció, les WSN poden ser molt diferents i hi ha que negociar els requeriments de la mateixa en base a les capacitats dels nodes i de les fonts d'energia.

CAPÍTOL 3. JUSTIFICACIÓ TÈCNICA

En aquest capítol es descriu la solució tècnica plantejada per CIMNE. Com ja s'ha mencionat anteriorment, l'objectiu de la WSN és fer de xarxa de suport als sonòmetres, emulant les seves prestacions.

El capítol s'organitza de la següent manera: en primer lloc es descriu la plataforma WSN, explicant les generalitats bàsiques de la mateixa així com comparant les característiques dels diferents elements que la poden formar. També es justifica l'elecció del hardware emprat pel disseny de la WSN.

En segon lloc es descriu el processat de dades, tant analògic com digital. S'expliquen les característiques del mateix en base al hardware seleccionat en l'apartat anterior. També es presentaran els resultats teòrics més rellevants per a cada una de les parts.

Finalment, el darrer apartat és el dedicat a les comunicacions entre nodes. Aquí fem una breu explicació del protocols de comunicació i altres característiques de la xarxa a tenir en compte.

3.1. Plataforma WSN

En aquest apartat es descriu la plataforma WSN que s'ha dissenyat i fet servir pel Projecte. La primera secció d'aquest apartat la dediquem a introduir el concepte de la WSN. A la segona secció fem una comparació del elements hardware que la poden formar, a nivell de processat de dades i comunicacions. Finalment, a la tercera secció justifiquem l'elecció de la plataforma hardware i de comunicacions escollida.

La figura 3-1 mostra el diagrama de blocs de qualsevol node dins la WSN de SIAC. Els nodes estan formats per un sistema de captació de les dades acústiques (sensor i circuit d'adaptació del senyal), un processador que fa la conversió A/D del senyal; i el xip ràdio que permet la comunicació entre nodes. Aquests elements s'explicaran al llarg del capítol.

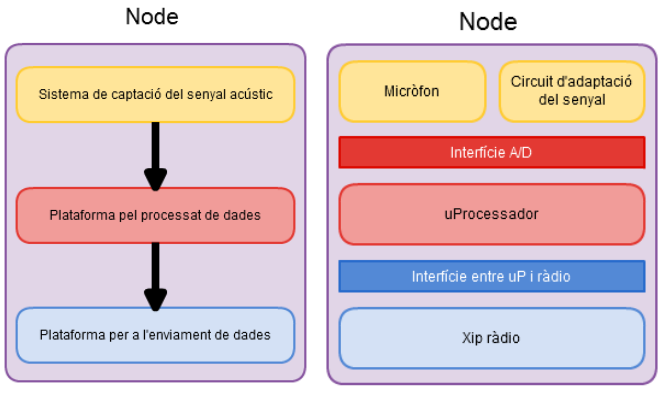


Figura 3-1 Esquema de blocs per un node de la WSN

La figura 3-2 mostra una representació de la WSN i la seva posició dins SIAC, com a xarxa de suport per a la xarxa de sonòmetres. En la figura veiem com els nodes

transmeten la informació acústica a un *gateway*, i aquest ho fa cap una base de dades (BBDD), a on es serviran per ser emprats pels altres estadis de SIAC.

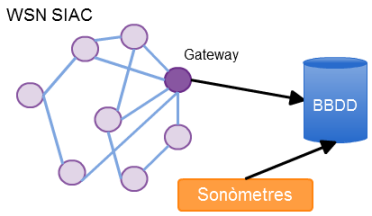


Figura 3-2 Esquema de la WSN i posició dins SIAC

El *gateway* és un node amb característiques diferents a la resta, que fa servir una interfície que permet passar dels estàndards de comunicació per WSNs a estàndards de xarxa mòbil (3G o GPRS).

Aquest element és necessari perquè el protocol emprat per aquest PFC no permet comunicacions a llargues distàncies, i no és viable col·locar el servidor a la intempèrie o físicament a prop de la WSN.

Així aconseguim transmetre la informació de la WSN cap a la BBDD. A la BBDD també es troba aquesta interfície que permet recollir les dades i després lliurar-les al servidor.

Ni la interfície del *gateway* ni la comunicació de les dades a la BBDD, ni els sonòmetres formen part dels objectius d'aquest PFC, per la qual cosa no s'expliquen.

Els sonòmetres també compten amb una interfície de comunicacions per transmetre les dades cap a la BBDD.

3.1.1. Comparació d'elements per a les WSN

En aquesta secció compararem les capacitats dels nodes en quant a processat i comunicacions. Comparem nodes de dos fabricants: Xbow i Arduino. Aquests fabricants s'han triat perquè són els dispositius dels quals disposa CIMNE per implementar la WSN.

3.1.1.1. Xbow: Mica2

La família de MICAx fa referència al conjunt format pel hardware de processat i comunicacions i les plaques de sensors. Aquest grup de sensors (per aquesta subsecció sobre els MICAx anomenem sensor al conjunt anteriorment descrit) han set creats pel fabricant Xbow [3].

A [7] podem trobar agrupats els diferents sensors pels models de MICAx, així com les característiques dels mateixos. Del que s'extreu de les referències és que la filosofia que hi ha al darrera de les plaques Xbow és oferir solucions completes per a les WSNs. Ofereixen les plaques de sensors adaptades juntament amb el hardware per processar les dades, sent una plataforma pensada per fer servir els seus productes i no per emprar d'altres (altres sensors per exemple).

A continuació passem a exposar les característiques de cada un dels models (característiques del processador, memòria i xip ràdio), extretes de [4, 5 i 6].

Taula 3-1 Característiques del processador i del xip ràdio pel MICA2 MPR400CB [5]

Component	Característica	Valor
Processador (ATmega 128L)	Freqüència rellotge	4 MHz
	Memòria Flash	128 KB
	Memòria per mesures (Flash)	512 KB
	EEPROM	4 KB
	Protocol comunicació sèrie	UART
	Conversor A/D	10 bits
	Consum de corrent del processador (típic)	8 mA
	Consum de corrent en *sleep mode*	< 15 uA
Xip ràdio	Freqüència ràdio (ISM)	868 / 916 MHz
	Nombre de canals (depèn de país)	4 / 50
	Baud rate (màxim, Manchester coding)	38.4 Kbauds/s
	Potència transmissió ràdio	-20 a 5 dBm
	Sensibilitat del receptor	- 98 dBm
	Consum de corrent del xip ràdio (típic, Tx/Rx)	27 / 10 mA
	Consum de corrent en *sleep mode*	< 1 uA
	Distància de transmissió (exterior)	152 m
	Alimentació	2x piles AA
	Alimentació externa	2.7 a 3.3 V

3.1.1.2. Arduino Duemilanove

L'altre plataforma de la que disposem per crear la WSN són les plaques Arduino. Al igual que el hardware de Xbow, incorporen un processador i memòria flash i EEPROM. Com afegit, les plaques Arduino incorporen un seguit d'entrades i sortides tant analògiques com digitals, a banda d'altres pins per voltatges de referència i massa.

Al igual que amb Xbow, existeixen nombroses plaques amb diferents característiques, totes elles baix el segell Arduino. En el nostre cas, el model emprat és el *Duemilanove* (2009), amb el PIC ATmega 328.

La següent taula mostra les característiques d'aquest model [9].

Taula 3-2 Característiques del hardware Arduino Duemilanove [9]

Component	Valor
Micro controlador	ATmega328
Voltatge de referència	5V
Voltatge d'entrada (recomanat)	7-12V
Voltatge d'entrada (límits)	6-20V
Pins digitals Entrada/Sortida	14 (6 dels quals proporcionen sortida PWN)
Pins analògics d'entrada	6

Corrent contínua per pin d'entrada	40 mA
Corrent pel pin de 3.3V	50 mA
Memòria flash	32 KB, 2 KB dels quals s'empren pel gestor d'arrancada
SRAM	2 KB
EEPROM	1 KB
Freqüència de rellotge	16 MHz

A [9] podem trobar més informació sobre la plataforma, així com el datasheet del processador ATmega 328.

3.1.1.3. Xbow vs. Arduino

La documentació analitzada [3 a 9] permet fer-se una idea de les filosofies de cadascuna de les plataformes hardware per crear WSN.

Xbow proveeix de solucions directament relacionades amb les WSNs, perquè les seves plaques estan específicament dissenyades per ser desplegades com a nodes per una WSN. Les plaques que contenen el processador i el xip ràdio es poden expandir amb diferents plaques de sensors que Xbow proporciona. El software que governa la programació dels nodes ha sigut creat a mida de les característiques de les plaques de Xbow.

Per contra, el producte Arduino no té una orientació tan clara cap a les WSNs. La seva plataforma no és més que un micro controlador amb entrades A/D i sortides D/A (entre d'altres), pensat com a interfície per a sistemes de control, monitorització, robots, etc. Si es volen crear solucions WSN amb Arduino, cal adquirir plaques d'expansió pels xips ràdio i pels sensors.

En quant a les característiques tècniques d'ambdues plataformes, no n'hi ha cap que destaqui especialment per sobre de l'altre. De fet, tant Xbow com Arduino tenen un processador de la família ATmega, memòries similars i el mateix nombre de bits en quant a resolució del ADC.

Finalment s'ha optat per Arduino. Els motius per escollir aquesta plataforma han estat que és oberta, amb multitud de projectes disponibles, i molt personalitzable. La interfície software integrada dins Arduino fa molt senzilla la interacció amb el mateix, i també és oberta.

Gràcies a les contribucions de la comunitat, Arduino s'ha convertit en una plataforma molt potent per realitzar projectes que impliquin processar dades o controlar sistemes. És una plataforma amb un disseny modular, que permet anar afegint mòduls de forma ràpida i senzilla per augmentar les seves prestacions.

Com a conclusió d'aquesta secció, dir que s'ha escollit Arduino pel disseny de la WSN pel motiu de facilitat d'ús, opcions en quant a personalització – integració de mòduls – i per ser una plataforma oberta.

En la següent secció és descriu amb detall el hardware Arduino emprat pel disseny de la WSN. També es descriu el mòdul Xbee utilitzada per a la comunicació entre nodes.

L'elecció del Xbee ve marcada pel fet de fer servir Arduino. Existeixen altre fabricants, però en la següent secció explicarem la motivació per treballar amb Xbee.

3.1.2. Descripció plataforma Arduino i Xbee

En aquesta secció explicarem les característiques del Arduino com a sistema per processar dades; i de Xbee com a interfície de comunicacions. És a dir, recollirem en un sol punt les especificacions del dos components, tot i que descripcions més detallades poden trobar-se a [9] pel que fa a Arduino i a [10] pel que fa a Xbee.

En pròximes seccions, dedicades a diferents aspectes del disseny de la WSN, s'entrarà en més detall en certs aspectes tant d'Arduino com de Xbee.

3.1.2.1. Arduino Duemilanove

La il·lustració 3-3 mostra els components i els pins de la placa Arduino Duemilanove utilitzada en aquest projecte [9].

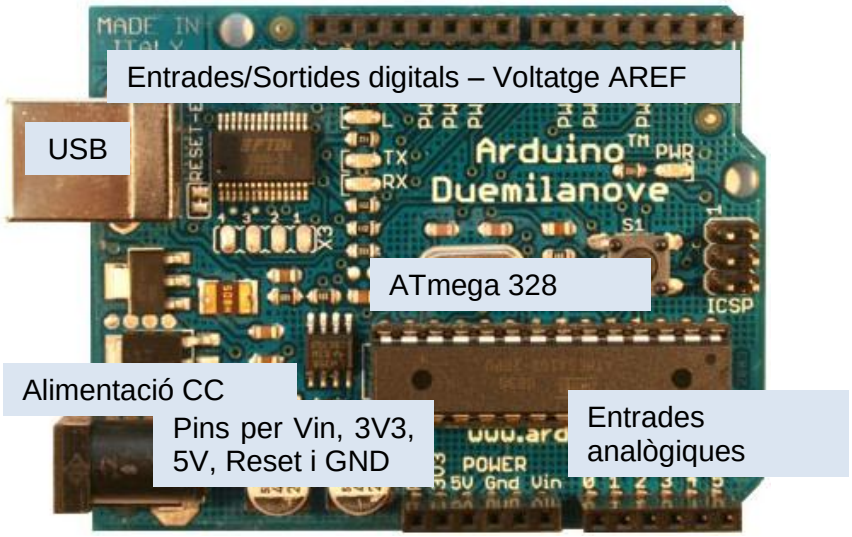


Figura 3-3 Placa Arduino Duemilanove [9]

Segons [9], les opcions per alimentar la placa són diverses:

- Connector USB.
- Connector per alimentació contínua.
- Vin entre 7 i 12 V (bateries per exemple).

Si hem d'alimentar la placa amb bateries, podem fer-ho directament connectant-les a través de Vin i GND. Aquestes fonts de tensió no és necessari que estiguin regulades, ja que la placa compta amb reguladors interns per proporcionar 5 V estables.

En quant a les entrades/sortides digitals, poden ser emprades indistintament tant com entrada o sortida de dades digitals. Les funcions de les mateixes són variades: sortides PWM, transmetre o rebre dades (TTL) o altres formes de comunicació digital amb les corresponents llibreries (més detalls a [9]).

La placa Arduino té 6 entrades analògiques de 10 bits de resolució. Per defecte el voltatge de referència és de 5 V, tot i que pot ser canviat amb AREF.

A banda de pins per entrar o extreure dades, la placa també permet comunicació amb el PC o altres plaques Arduino. Això ho pot fer a través del port USB, o dels pins per transmetre o rebre dades sèrie (TTL).

Altres característiques importants són que podem carregar codi des del PC a través del software específic i via USB. També la placa assegura que, a l'hora de carregar un nou codi, elimina el que prèviament existia.

3.1.2.2. Xbee

Xbee fa referència al conjunt de mòduls ràdio de *Digi International*. Com es pot veure a [11], a on la varietat d'opcions per xips ràdio és gran: des de ràdios per aplicacions punt-multipunt fins *mesh*, passant per protocols propietaris (*Mesh, Multipunt i Combinat*) o lliures com Wi-Fi o 802.15.4.

Els xips escollits per aquest Projecte són els que incorporen el protocol per xarxes mesh i tenen les següents característiques tècniques, resumides a la taula:

Taula 3-3 Resum característiques tècniques Xbee Mesh [14].

Característica	**Valor**
Tecnologia ràdio	IEEE 802.15.4
Bit rate (màxim)	250 kbps
Cobertura en interiors	40 m
Cobertura en exteriors (LOS)	120 m
Potència de transmissió	1 dBm (3 dBm màx.)
Sensibilitat de receptor (1% PER)	-97 dBm
Freqüència (ISM)	2.4 GHz
Velocitat dades port sèrie	1.2 kbps – 1 Mbps
Configuració (per software)	API (llibreries), comandes AT, de forma local o remota (over the air)

A [10] podem trobar la fulla de característiques del mòdul, així com explicacions detallades del seu funcionament. En aquesta subsecció resumirem les característiques que podem trobar a les referències. Al igual que hem comentat amb la plataforma Arduino, a l'apartat 3.3 s'entrarà en més detall en el funcionament d'algunes de les característiques del mòdul Xbee.

El mòdul Xbee també té entrades analògiques (conversor AD de 10 bits) i digitals, a més d'estar capacitat per transmetre dades pel seu port sèrie. Tot i així, en aquest projecte no hem fet servir cap de les seves entrades analògiques o digitals per capturar dades. Únicament ens hem centrat en explotar les seves capacitats de comunicació sense fils.

Els mòduls emprats tenen dos modes de funcionament:

- *Transparent Operation*: en aquest mode, el xip ràdio es comporta com un substitut al port sèrie, transmetent les dades tal qual arriben. Amb aquest mode

no podem interactuar amb les capacitats de xarxa del mòdul. Aquest mode és útil si el que volem fer es transmetre les dades sense preocupar-nos per res més.

- *API (Application Programming Interface) Mode*: en aquest mode, les dades també són enviades tal i com arriben, però podem interactuar amb les capacitats de xarxa del mòdul. Per permetre això, les dades són paquetitzades el que permet:
    - Transmetre dades a múltiples destinacions, sense emprar comandes AT.
    - Rebre estats d'enviament correcte o incorrecte per a cada paquet enviat.
    - Identificar adreces.

Pel que fa a comunicació amb altres nodes, aquesta pot ser punt a punt, o punt a multipunt (*broadcast* inclòs). El mòdul també pot "apagar-se" quan no ha de transmetre durant un determinat període, el que es coneix com *sleep mode*. A [10] s'expliquen les particularitats del mode *sleep*, així com les característiques i particularitats de les entrades analògiques o digitals en cas de voler-les fer servir.

3.2. Processat de dades

En aquest apartat expliquem el sistema d'adquisició de dades format pel micròfon i el processat, tant analògic pel que fa a l'adaptació del senyal provinent del micròfon; com el digital pel que fa a la implementació de les rutines per calcular la FFT.

Primer explicarem el circuit d'adaptació pel senyal provinent del micròfon. Descriurem les característiques bàsiques d'aquest procés, així com certes consideracions a tenir en compte.

A continuació parlarem sobre la conversió del senyal analògic a digital, explicant també certes particularitats del conversor A/D emprat pel processador ATmega 328.

Finalment, descriurem la rutina per calcular la FFT i el filtre aplicat per processar les dades acústiques.

La següent figura mostra el diagrama de blocs de tota la part de processat de dades, formada pel processat analògic (amplificador) i digital (conversor).

Figura 3-4 Esquema de blocs pel sistema d'adaptació del senyal

3.2.1. Circuit d'adaptació del senyal

3.2.1.1. Micròfon

El sensor és un micròfon de tipus electret, com el mostrat a la il·lustració 3-5. En aquesta imatge es veu l'encapsulat del micròfon, així com els terminals positiu i massa del mateix.

Figura 3-5 Càpsula pel micròfon de condensador electret [15]

La següent figura mostra l'esquemàtic de la imatge anterior:

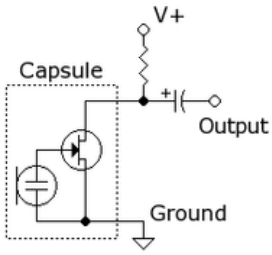


Figura 3-6 Esquemàtic del micròfon. També es mostra la polarització del mateix [16]

En primer lloc, un micròfon de tipus electret no és més que un condensador que produeix un voltatge variable a la seva sortida, depenent de la pressió acústica aplicada sobre la càpsula. Aquesta pressió acústica fa variar la capacitat del micròfon, fent variar la tensió a la seva sortida.

Els micròfons electret es construeixen amb un amplificador FET, que proporciona l'energia per crear la senyal a la sortida del micròfon.
La figura 3-6 mostra la càpsula del micròfon, formada pel condensador electret i per l'amplificador FET.

Degut a que la senyal és extreta a través d'un element actiu (FET), és necessari aplicar una tensió per alimentar-ho. Aquesta tensió està marcada a la figura 3-6 com V+, i és conduïda a la càpsula a través d'una resistència, anomenada resistència de polarització.

La resistència s'escull entre els valors d'1 KΩ i 10 KΩ, si volem alimentar el micròfon entre 1 i 9 V (terminal V+ a la figura 3-6) [18]. Aquestos valors s'escullen per adaptar la impedància d'entrada entre la font de tensió i el FET, i són els valors de resistència de sortida que vorà qualsevol sistema acoblat a la càpsula.

En el Capítol quart realitzarem la caracterització del micròfon, ja que té més sentit que fer-la en aquest capítol on s'introdueixen els conceptes teòrics i altres detalls a tenir en compte pel que fa al funcionament del sistema.

3.2.1.2. Processat analògic. Adaptació del senyal a l'Arduino

En primer lloc, la sortida que proporciona el micròfon és molt petita (del ordre dels mili, o inclús, microvolts), de forma que és necessària una etapa de processat analògic, per adaptar la senyal de sortida del micròfon a l'entrada del conversor A/D del processador de l'Arduino.

En segon lloc, aquesta etapa de processat està formada únicament per un amplificador específic per aplicacions d'àudio. Això és així perquè en aquest Projecte no es considera l'ús d'altres sensors. Tot i això, abans d'implementar l'amplificador d'àudio, es van estudiar altres opcions d'amplificadors operacionals que seran analitzades més endavant en aquesta subsecció.

Ja que les característiques de l'entrada analògica del Arduino són fixes, és lògic que comencem per elles per definir la resta d'especificacions en els components com l'amplificador.

La següent taula resumeix les característiques principals del conversor a tenir en compte.

Taula 3-4 Resum característiques principals conversor A/D [19]

Nombre de bits (n)	10
Voltatge de referència (V_{Ref})	5 V

Amb aquestes dades podem obtenir quina és la resolució del conversor A/D, és a dir, quin és el mínim voltatge a l'entrada que és capaç de detectar:

$$V_{min} = \frac{V_{Ref}}{2^n} = \frac{5\ V}{1024} \cong 4.88\ mV \qquad \textbf{(3.1)}$$

De l'expressió 3.1 anterior entenem la necessitat per l'etapa d'amplificació, ja que com hem comentat, la sortida que proporciona el micròfon és més baixa que la sensibilitat mínima del conversor.

Per determinar el guany de l'amplificador, hem de tenir en compte quin és el marge de tensions de sortida del micròfon, i quin el marge de tensions d'entrada del conversor. Pel conversor A/D, el rang de tensions d'entrada és de 0 a 5 V. Pel micròfon no ho podem determinar de forma teòrica, ja que el nivell de tensió de sortida depèn del que el micròfon estigui captant en aquell moment.

No obstant això, segons [18] o [20] sembla ser que valors típics de tensions de sortida són 1 o 2 mV. Si això fos cert, segons l'expressió 3.2 pel guany de l'amplificador.

$$G = \frac{V_{Ref}}{V_{out_{máx}} - V_{out_{min}}} = \frac{5\ V}{1\ mV} \cong 5000\ (37\ dB) \qquad \textbf{(3.2)}$$

Necessitem doncs un guant de 37 dB en tot el rang de freqüències de la nostra aplicació (dels 20 Hz als 20 kHz). En aquesta expressió, V_{Ref} és el voltatge d'entrada de l'Arduino (entre 0 i 5 V); i el denominador és el rang de voltatges de sortida suposats, entre un màxim i un mínim.

Aquesta fita és complicada d'assolir, principalment perquè el guany dels amplificadors operacionals depèn de la freqüència de la senyal a amplificar (el que es coneix com Gain-Bandwith Product). Segons 3.3 [21]:

$$G(f) = \frac{G_0}{1 + j\frac{f}{f_G}} \qquad \textbf{(3.3)}$$

tenim que el guany dels amplificadors operacions es redueix segons augmenta la freqüència de la senyal a amplificar. La figura següent (modificada de [21]) és una representació gràfica de l'expressió 3.3:

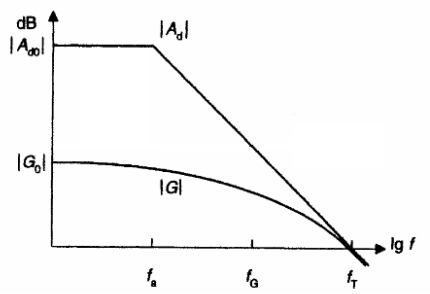
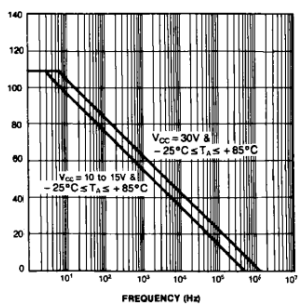
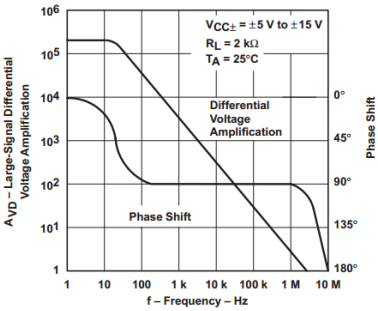


Figura 3-7 Gain-Bandwidth product [21]

En la figura 3-7 es veu que tant el guany en llaç obert A_d com en guany per realimentació negativa G, decreixen si la freqüència del senyal augmenta.

Com que el rang de freqüències de la nostra aplicació és eleva (aproximadament 20 kHz), a continuació mostrem les gràfiques de guany vs. freqüència de diferents amplificadors operacionals que podrien ser útils pel que volem.





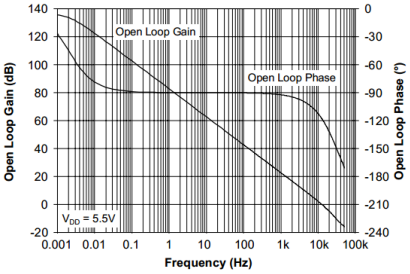


Figura 3-8 LM358 [22] **Figura 3-9** TL071 [23] **Figura 3-10** MCP6042 [24]

Aquestes gràfiques mostren el guany en llaç obert dels amplificadors operacionals. Si prenem com a referència la figura 3-7, entenem que el guany proporcionat pel llaç de realimentació negativa serà menor, i que per tant les freqüències properes a 20 kHz quedarien menys amplificades.

Tot i això, podem considerar el LM358 i el TL071, perquè podrien proporcionar un guany acceptable. No passa el mateix amb el MCP6042, que presenta un guany molt pobre per a freqüències superiors a 5 kHz, per la qual cosa ni el considerem com a opció.

Degut a que el rang de voltatges d'entrada de l'Arduino és positiu, i per simplificar el circuit del micròfon, descartem l'ús del TL071 ja que requereix d'alimentació positiva i negativa.

També descarem el LM358 en favor del LM386, específic per àudio i amb la següent resposta freqüencial:

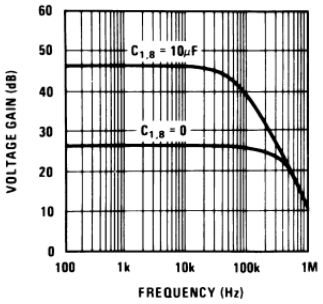


Figura 3-11 Gràfica guany vs freqüència del LM386 [25]

Això ho hem fet perquè mostra una resposta molt més plana per a les nostres freqüències d'interès.

El LM386 ofereix, amb un conjunt mínim de components, dos guanys: 26 dB i 46 dB aproximadament. Segons la nostra suposició inicial de que necessitaríem un guany de 37 dB, escollim el guany màxim de 46 dB. La figura 3-11 mostra els components necessaris per aconseguir-ho.

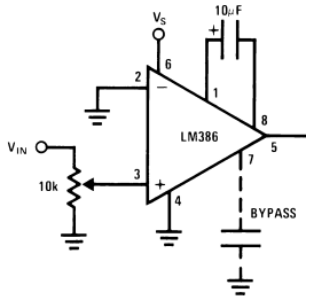


Figura 3-12 Components per aconseguir un guany de 46 dB [25]

Amb tot, teòricament tenim adaptada la senyal provinent del micròfon, que és una senyal dèbil, a l'entrada de l'Arduino. L'explicació del perquè dels components a la figura 3-12 es pot trobar a la fulla d'especificacions del fabricant [25].

En el Capítol 4 comprovarem si les suposicions són correctes o si per contra hem de realitzar ajustos en el guany per problemes de saturació a l'entrada del conversor A/D.

3.2.2. Conversió Analògica – Digital

El processador ATmega 328 que incorpora l'Arduino té un conversor A/D de 10 bits, del tipus d'aproximacions successives. A [x], a més de teoria sobre la conversió A/D, es pot trobar una explicació del funcionament d'aquests tipus de conversors.

Aquesta secció no pretén explicar els fonaments de la conversió A/D, si no que el que es fa és explicar les parts que intervenen en aquest procés en relació directa amb els objectius del Projecte.

En primer lloc a [26] podem trobar l'esquema de blocs del conversor A/D del ATmega 328, i a la taula següent es resumeixen les característiques del conversor, tretes de [19].

Taula 3-5 Resum de les característiques del conversor A/D [19]

Nombre de bits	10
V_{Ref}	5 V
Freqüència de rellotge	16 MHz
Nombre de canals d'entrada	6
No linealitat	0.5 LSB
Precisió	± 2 LSB

Aquestes especificacions són les referents al conversor juntament amb la plataforma Arduino. El que volem dir és que existeix cert nivell de configuració: el voltatge de referència pot ser diferent segons escollim la font interna de 1.1 V, o altres fonts externes.

A més, la freqüència del rellotge també pot ser diferent. El fabricant d'Arduino ha considerat col·locar un rellotge de quars de 16 MHz per conduir el conversor A/D. Hi ha que anar amb compte a l'hora d'escollir quina freqüència de mostreig volem (la freqüència de rellotge), ja que segons el fabricant del processador, freqüències per sobre del 200 kHz fan perdre resolució en nombre de bits [19].

La freqüència del senyal d'entrada és un aspecte molt important a tenir en compte a l'hora de convertir senyals. En teoria, per a la nostra aplicació amb un ample de banda de quasi 20 kHz i per tant freqüències de mostreig de 40 kHz, no hauríem de tenir problemes en assolir 10 bits de resolució.

Un altre aspecte molt important a l'hora de la conversió és la impedància d'entrada del conversor. Segons [19], les impedàncies de sortida han de ser iguals o inferiors a 10 KΩ. Això és degut a que poden aparèixer efectes de càrrega a l'entrada del conversor, fent que la tensió d'entrada sigui inferior a l'esperada amb un guany determinat.

L'únic aspecte de la configuració del A/D que hem modificat és la freqüència de mostreig. El voltatge de referència l'hem deixat a 5 V i no hem entrat a modificar els registres interns del conversor, és a dir, no hem modificat el funcionament del mateix afegint interrupcions a través de fonts externes.

A continuació passem a explicar el procés de modificació de la freqüència de mostreig del A/D.

3.2.2.1. Prescaler. Modificació de la f_{clock}

El *prescaler* és un divisor de la freqüència del rellotge (f_{clock}) de 16 MHz que incorpora la plataforma Arduino. El conversor A/D pren com a referència la freqüència d'aquesta font, de forma que és aquí a on podem determinar el temps de mostreig que necessitem per a la nostra aplicació.

La figura 3-13 mostra l'esquema del *prescaler*, que funciona de la següent manera: cada cop que la conversió acaba, el bit ADEN (ADC Enable) fa un reset del *prescaler*. Com podem veure, l'altre entrada és la freqüència de rellotge que en el cas de l'Arduino és de 16 MHz. Els factors de divisió s'escullen amb els bits ADPS0, ADPS1 i ADPS2. La taula 3-6 mostra la correspondència entre aquests bits i el factor de divisió.

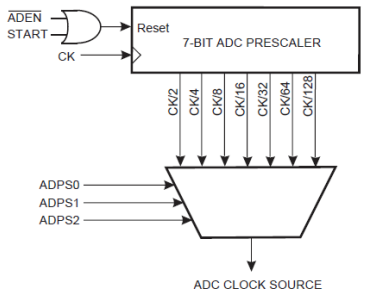


Figura 3-13 Esquema del Prescaler per a la f_{clock} [19]

Taula 3-6 Correspondència entre els bits ADPS i el factor de divisió [19]

ADPS0	ADPS1	ADPS2	Factor de divisió
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Si tenim en compte que una conversió normal són 13 cicles de rellotge [19], la freqüència de partia pel càlcul de les diferents freqüències de mostreig és:

$$f_{mostreig_{màx}} = \frac{f_{clock}}{13\ cicles/conversió} = \frac{16\ MHz}{13\ cicles/conversió} \cong 1.23\ MHz \qquad \textbf{(3.4)}$$

La taula 3-7 recull les diferents freqüències de mostreig (juntament amb els temps de mostreig), a on *n* fa referència al factor de divisió de la taula 3-6 anterior.

Taula 3-7 Freqüències i temps de mostreig segons el factor de divisió aplicat

Factor divisió *n*	$f_{mostreig} = \frac{f_{mostreig_{max}}}{n}\ (KHz)$	$t_{mostreig} = \frac{1}{f_{mostreig}}\ (\mu s)$
2	615.00	1.63
4	307.50	3.25
8	153.75	6.50
16	76.88	13.01
32	38.44	26.02
64	19.22	52.03
128	9.61	104.07

Si la nostra aplicació té un ample de banda d'uns 20 kHz, la freqüència de mostreig ha de ser del doble (Nyquist). Això comporta que haguem de seleccionar un factor de divisió per 32, aconseguint una freqüència de rellotge de 38.4 kHz. Això defineix una freqüència màxima pel senya d'entrada al conversor de 19.2 kHz (Nyquist).

En el capítol Quart es mostra el codi emprat per modificar el registre ADSCRA del processador, que conté els bits de configuració del *prescaler*.

3.2.3. Processat digital. Rutina FFT

En aquesta secció farem una introducció teòrica pel que fa a les transformades discretes de Fourier (DFT *Discrete Fourier Transform* en anglès). A continuació descriurem les característiques bàsiques de la FFT (*Fast Fourier Transform* en anglès) com a cas particular de la DFT.

Com en la secció anterior, no es pretén fer explicacions teòriques detallades, si no simplement explicar els conceptes relacionats directament amb els objectius del projecte. Aquest capítol està basat en [27].

Les motivacions per l'ús de la FFT (com a cas particular de la DFT) són molt senzilles: per una banda estem treballant amb senyals discretes en temps gràcies a l'etapa A/D explicada en la secció anterior; i de llargada finita. Això fa que els algorismes de càlcul de l'espectre de la senyal siguin lleugerament diferents que per a les senyals contínues en temps.

Per altre banda, la FFT està especialment dissenyada per reduir el nombre d'operacions necessàries pel seu càlcul, reduint la càrrega computacional sobre el processador. Tenint en compte les limitacions del processador amb el que treballem, aquest fet marca de forma clara l'elecció de la FFT com a rutina pel càlcul de l'espectre d'àudio.

3.2.3.1. DFT: Discrete Fourier Tranform

La DFT no és més que la transformada de Fourier d'una seqüència de dades discreta en temps, de durada finita. És a dir, la TFSD (Transformada de Fourier de Seqüències Discretes) està definida en un interval $(-\infty, +\infty)$ per a la senyal d'entrada (senyal de duració infinita). En canvi la DFT només està definida per un interval [0, N-1], és a dir, la seqüència d'entrada té una duració finita de N mostres.

Una altre forma de voreu és pensar que la DFT és una realització pràctica de la TFSD, al contemplar seqüències finites en el temps. Això li ofereix un gran idoneïtat a l'hora de treballar amb dispositius i sistemes digitals.

L'expressió 3.5 és la definició de la DFT d'una seqüència discreta f[n] en un interval [0, N-1]:

$$F[k] = \sum_{n=0}^{N-1} f[n] \cdot e^{-j\left(\frac{2\pi}{N}\right)nk} \quad \textbf{(3.5)}$$

L'índex n fa referència a l'índex de la mostra de la senyal i k és l'índex sobre l'eix freqüencial. És a dir, la DFT ens dona una seqüència discreta de components freqüencials entre (-π, π) (recordem que la DFT és periòdica amb període 2π, i que tant la podem representar entre 0 i 2π com entre –π i π).

La figura 3-14 és una representació del comentat en el paràgraf anterior, a on cada punt representa una component freqüencial de la DFT d'una seqüència de N mostres.

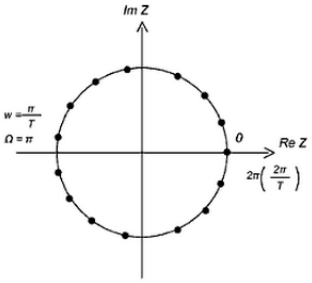


Figura 3-14 Representació sobre la circumferència de radi unitat de les freqüències de la DFT [27]

L'expressió 3.6 és la IDFT, la inversa de la DFT:

$$f[n] = \frac{1}{N} \sum_{k=0}^{N-1} F[k] \cdot e^{j\left(\frac{2\pi}{N}\right)nk} \quad \textbf{(3.6)}$$

Ara ja sabem que la DFT ens proporciona un espectre mostrejat, donada una senyal temporal discreta. Això implica que en funció del temps de mostreig, hi haurà una freqüència mínima i un ample de banda màxim.

L'expressió 3.7 representa la freqüència mínima o resolució de la nostra DFT, i a on N indica el número de mostres de la seqüència d'entrada i T és el temps de mostreig del conversor A/D.

$$W_{min} = \frac{2\pi}{NT} \rightarrow f_{min} = \frac{1}{NT} \quad \textbf{(3.7)}$$

L'expressió 3.8 indica la freqüència màxima o ample de banda de la DFT:

$$W_{max} = \frac{2\pi}{NT}(N-1) \rightarrow f_{max} = \frac{1}{NT}(N-1) \quad \textbf{(3.8)}$$

Abans hem comentat que l'índex k feia referència a l'índex de la component espectral en qüestió. Si prenem la freqüència mínima com el mínim salt entre components freqüencials, i prenem k com l'índex de la component freqüencial, és fàcil determinar quina component freqüencial hi ha a cada valor de l'índex:

$$f_k = \frac{1}{NT}k \quad \textbf{(3.9)}$$

Tot i que la DFT sigui, per definir-ho d'una forma simple, una realització pràctica de la TFSD, a l'hora de computar-la no és un algorisme eficient.

L'algorisme de la FFT és molt més eficient que la DFT, sempre i quan la longitud de la seqüència d'entrada sigui potència de 2. En la següent subsecció descrivim aquest algorisme, emprat pel càlcul espectral en aquest Projecte.

3.2.3.2. FFT

Aquest algorisme va se proposat per Cooley i Tukey en els anys 50. La seva potència radica en que si la longitud de la seqüència de mostres a l'entrada és potència de 2, el nombre d'operacions creix amb el logaritme en base 2 de n (n és el nombre de mostres en un període T).

La següent taula 3-14 mostra una comparació en quant a nombre d'operacions entre una DFT i una FFT [27].

Taula 3-8 Comparació en el nombre d'operacions entre DFT i FFT de N punts [27]

	DFT N punts (directa)	**FFT**
Multiplicacions complexes	N^2	$N^2/2 + N$
Multiplicacions reals	$4N^2$	$4\ (N^2/2 + N)$
Sumes complexes	N(N-1)	$N^2/2$
Sumes reals	$2N(N-1) + 2N^2$	$2N^2/2$

Si prenem un nombre N de 128 mostres per exemple, tenim:

Taula 3-9 Comparació en nombre d'operacions entre DFT i FFT per N = 128 mostres

	DFT N punts (directa)	**FFT**
Multiplicacions complexes	16 384	8 320
Multiplicacions reals	65 536	33 280
Sumes complexes	16 256	8 192
Sumes reals	65 280	16 384

Com es pot comprovar, l'estalvi computacional és considerable entre ambdós algorismes.

El funcionament de l'algorisme de la FFT consisteix en aprofitar el fet de la longitud parella de la seqüència, evitant operacions redundants. Per descriure el funcionament de l'algorisme primer hi ha que determinar si es fa mitjançant un delmat en temps o en freqüència: en el primer cas les mostres es reordenen i el resultat a la sortida de la rutina queda ordenat; en el segon cas, els càlculs es realitzen tal qual arriben les mostres, però el resultat s'ordena abans de la sortida. En qualsevol dels casos, el cost computacional és el mateix [27].

En aquest Projecte hem escollit un delmat en temps, de forma que la seqüència d'entrada es reordena separant les mostres en un vector d'índexs parells i d'altre de senars. La figura 3-15 és una mostra d'això:

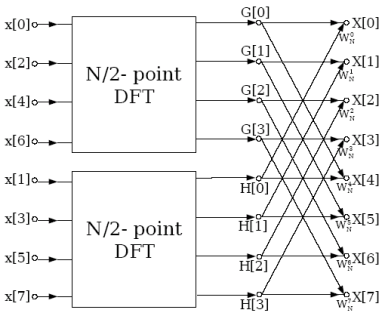


Figura 3-15 Esquema del algorisme de càlcul de la FFT [29]

A l'entrada dels blocs que realitzen la DFT podem veure la seqüència reordenada (en aquest cas, una seqüència de 8 mostres); i com a la sortida es mostra el càlcul de cada sortida X[k], a on k representa cada punt de la DFT.

La figura 3-15 és una representació gràfica de les següents expressions, tretes de [27], que no són més que les mateixes que l'expressió 3.5 anterior:

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} x_1[n] \cdot W_{N/2}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_2[n] \cdot W_{N/2}^{nk} \qquad \textbf{(3.10)}$$

Tenint en compte que:

$$W_{N/2}^{nk} = e^{-j\left(\frac{2\pi}{N/2}\right)nk} \qquad \textbf{(3.11)}$$

Simplificant:

$$X[k] = G[k] + W_N^k H[k] \qquad \textbf{(3.12)}$$

A la següent figura veiem què hi ha dins els blocs de la DFT:

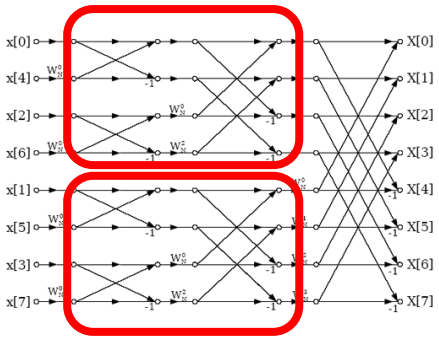


Figura 3-16 Seqüència completa de la FFT, mostrant l'interior dels blocs DFT [29]

El que podem observar dins aquests blocs és una repetició successiva d'una DFT de dos punts, la DFT més bàsica implementada per l'algorisme de la FFT. Aquí es justifica l'ús d'una seqüència potència de dos, perquè la podem dividir fins aconseguir una DFT de dues mostres, amb un cost computacional baix.

A la figura 3-17 observem que la DFT de dos punts es calcula amb de la següent forma, a on i = k per ser coherents amb la formulació anterior:

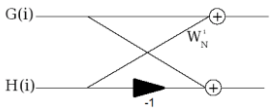


Figura 3-17 Operador en "papallona" [29]

La figura 3-18 mostra l'operador "papallona" (*butterfly* en anglès) que és l'operador bàsic pel càlcul d'una DFT de dos punts. Com veiem a la figura 3-16, aquest operador es va repetint al llarg de tot el procés, però sempre realitza una DFT de dos punts, prenent com entrades les sortides dels operadors anteriors.

Finalment, una FFT no és més que l'encadenament d'operadors "papallona", que realitzen una DFT de dos punts a cada pas. Per fer una FFT, primer es trenca la seqüència i després es fan els càlculs, i com s'ha vist anteriorment, aquest és un mecanisme més efectiu que realitzar una DFT amb un nombre de punts igual a la longitud de la seqüència d'entrada. En els annexes es troba la rutina implementada pel càlcul de la FFT en aquest Projecte.

Un cop arribats fins aquí, estem en disposició de fer els primers càlculs teòrics sobre l'ample de banda consumit per a la nostra aplicació.

Si establim una FFT de N = 128 punts, i tenint en compte el temps de mostreig, calculat a l'apartat anterior, de 26 µs tenim que:

$$f_{min} = \frac{1}{128 \cdot 26\ \mu s} \cong 300\ Hz \qquad \textbf{(3.13)}$$

$$f_{max} = \frac{1}{128 \cdot 26\ \mu s}(128 - 1) \cong 38.1\ KHz \qquad \textbf{(3.14)}$$

És a dir, amb una FFT de 128 punts la mínima freqüència que el nostre sistema pot detectar són 300 Hz, mentre que la màxima són 19.2 kHz.

A priori considerem que aquests resultats són suficientment bons, a falta de les proves definitives al Capítol 4.

3.2.4. Filtre A-Weighting

Segons [42], el A-Weighting (AW en endavant) és una família de corbes emprades per mesurar nivells de pressió acústica (*Sound Pressure Level* SPL en anglès). Aquestes corbes s'han convertit en un referent a l'hora de mesurar soroll ambiental, i formen part de les especificacions de nombrosos aparells de mesura acústica (sonòmetres).

Com l'objectiu principal d'aquest Projecte és realitzar mesures acústiques, aquest filtrat també s'ha implementat en el sistema. Com veurem més endavant, aquest filtre no és més que una taula de valors pels quals es multiplica (o suma en logaritmes) el resultat de la FFT, obtenint un resultat ponderat segons el filtre.

Les següents equacions defineixen la corba pel AW. En primer lloc, l'expressió 3.15 defineix el coeficient del filtre $R_A(f)$, depenent de la freqüència f.

$$R_A(f) = \frac{12200^2 \cdot f^4}{(f^2 + 20.6^2) \cdot \sqrt{(f^2 + 107.7^2)(f^2 + 737.9^2)(f^2 + 12200^2)}} \quad \textbf{(3.15)}$$

L'expressió 3.16 expressa de forma logarítmica els coeficients del filtre en dB. El terme constant es fa servir per compensar el "guany" a freqüències properes a 1 kHz.

$$A(f) = 2.0 + 20\log_{10}(R_a(f)) \quad \textbf{(3.16)}$$

Amb aquestes expressions podem calcular els coeficients del filtre, segons les freqüències que detectarà el nostre sistema. Recordem que la freqüència mínima és de 300 Hz, i que la resta de freqüències del sistema són 300·k Hz.

La següent gràfica mostra la corba obtinguda amb l'expressió 3.16.

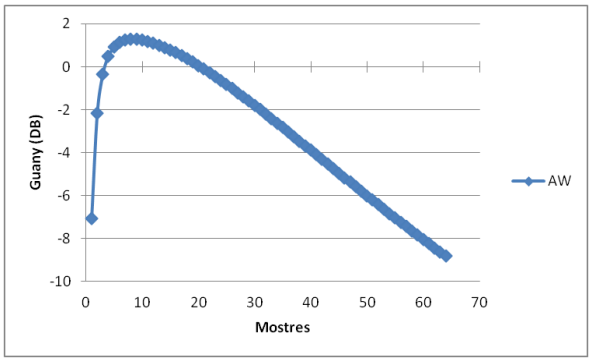


Figura 3-18 Filtre AW

Així doncs, una vegada obtinguda l'amplitud de la senyal de la FFT, podem aplicar aquest filtre i obtindre el resultat ponderat, ja sigui en una escala lineal o en dBA.

3.3. Comunicacions

Aquí cobrim l'apartat de comunicacions entre nodes de la WSN. Ja hem explicat com es capten i processen les dades, de forma que ara toca explicar com les enviem.

Donat que, com en els capítols anteriors, no es volen fer explicacions detallades dels diversos aspectes que inclourien les comunicacions entre nodes, ens centrarem en explicar les característiques del producte Xbee.

El Xbee emprat per aquest projecte funciona sobre un protocol propietari de Digimesh, creant una xarxa amb tipologia mesh [34]. Per tant Xbee no és un protocol, i a [34] podem veure nombrosos productes amb la marca Xbee que funcionen sobre diferents protocols.

Però per entendre Xbee, primer hem d'introduir el protocol del IEEE 802.15.4 [31, 33], i la família d'especificacions *Zigbee* creada per la *Zigbee Alliance*. El motiu és que la capa física i d'accés al medi del producte Xbee (i del protocol Digimesh associat en el nostre cas) està governada pel 802.15.4; mentre que les capacitats a nivell d'encaminament, seguretat, etc. del Xbee són heretades de Zigbee.

A continuació passem a explicar els diferents protocols. Hi ha que tenir en compte que els protocols per WSNs tenen una característica fonamental: es basen en la reducció del consum energètic mitjançant potències de transmissió baixes (baix radi de cobertura) i velocitats de transmissió reduïdes (del ordre dels centenars de kbps).

3.3.1. Introducció al IEEE 802.15.4

Segons [1 i 35], el 802.15.4 especifica les característiques de la capa física i de capa de control d'accés al medi per a les xarxes d'àrea personal amb un consum baix de potència.

Això vol dir que és un protocol que s'encarrega, pel que fa a la capa física (PHY), de gestionar les funcions del transmissor com modulació/demodulació del senyal, selecció de canals i altres tasques per gestionar el consum energètic.

Pel que fa a la capa d'accés al medi (MAC) s'encarrega de regular l'accés dels nodes al medi, és a dir, quan un node pot accedir al medi per transmetre, controlar o gestionar una trama cap un o varis nodes.

La següent figura, extreta de la primera versió del estàndard [35], és una representació gràfica del lloc que ocupa el 802.15.4 en la jerarquia de protocols basada en el model OSI.

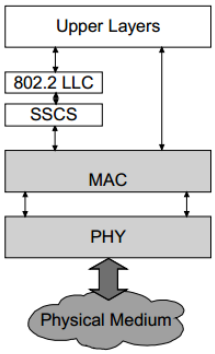


Figura 3-19 Pila de protocols pel 802.15.4 basat en el model OSI [35]

Aquest estàndard permet operar en diversos rangs de les bandes ISM: 868, 902-928 i 2400 MHz. L'estàndard està en constant revisió per permetre altres rangs freqüencials a altres regions que permeten l'ús sense llicència de l'espectre radio.

Les velocitats de transmissió també han anat evolucionant al llarg del temps. En la primera versió del protocol, per a les bandes de 868 i 902-928 MHz les velocitats de transmissió eren entre 20 i 40 kbps, mentre que a la banda dels 2400 MHz era de 250 kbps. Successives revisions d'aquest estàndard han augmentat la velocitat de transmissió per a les bandes inferiors als 2400 MHz.

Altres característiques importants d'aquest protocol és que empra DSSS (*Direct Sequence Spread Spectrum*) per modular la informació, de forma que les transmissions són molt robustes enfront a interferències i soroll. A més, incorpora mecanismes per evitar que els nodes es trepitgin entre ells a l'hora de transmetre. Finalment, com hem comentat al principi d'aquesta secció, al ser un protocol pensat per WSNs incorpora també mecanismes per permetre l'estalvi energètic dels nodes, fent que les transmissions de dades ocupin poc temps (i energia).

L'estàndard defineix dos tipus de node a la xarxa:

- *Full Function Device* o FDD: vol dir que un node pot adquirir el rol de coordinador de tota la xarxa, de coordinador d'un segment de la mateixa o de node simple.
- *Reduced Function Device* o RFD: només integra als nodes simples, no coordina ni enruta informació. Només es poden comunicar amb FDDs.

Les definicions d'aquests tipus de node porten a definir les topologies de xarxa que es permeten amb el 802.15.4, que bàsicament són dues:

- Topologia en estrella.
- Topologia punt a punt.

La xarxa pot estar formada per múltiples coordinadors (FDD), però hi ha d'haver al menys un coordinador de xarxa (també FDD).

La figura 3-20 mostra un exemple de les topologies permeses per l'estàndard.

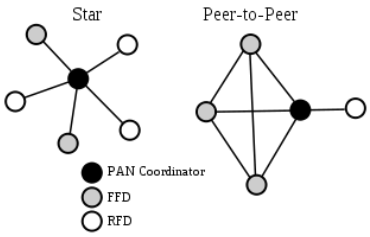


Figura 3-20 Esquema de les topologies permeses pel 802.15.4 [31]

3.3.2. Introducció al Zigbee

Per sobre del 802.15.4 hi ha tota una sèrie d'altres protocols que afegeixen altres funcionalitats a nivell d'encaminament de la informació, seguretat, etc. Un d'ells és el Zigbee, de la *Zigbee Alliance*.

Segons [32], Zigbee són un conjunt d'especificacions per expandir les capacitats del 802.15.4, afegint-les amb una capa superior. Les capacitats que afegeix són: enrutament de la informació, xifrat i serveis d'aplicació. A més, ja que el 802.15.4 està pensat per optimitzar el consum energètic i per ser desplegat en WSNs, amb Zigbee aquesta filosofia es manté.

A més, afegeix noves definicions pels nodes, ja que ara un node pot ser un router, coordinador o *end device*. La diferència entre ells és que el coordinador forma la xarxa i pot parlar amb altres coordinadors; el router envia informació entre nodes o entre nodes i coordinador; i el *end device* l'únic que pot fer és parlar amb el seu node superior, sigui un router o un coordinador.

Degut a aquesta estructura, un *end device* no pot comunicar-se amb un d'igual, i ha de passar necessariament per un node router o coordinador. També hi ha que mencionar que segons [33], la xarxa desplegada amb Zigbee ha de comptar amb al menys un coordinador per definir-la.

La figura 3-21 mostra l'esquema de la pila de protocols, integrant 802.15.4 i Zigbee.

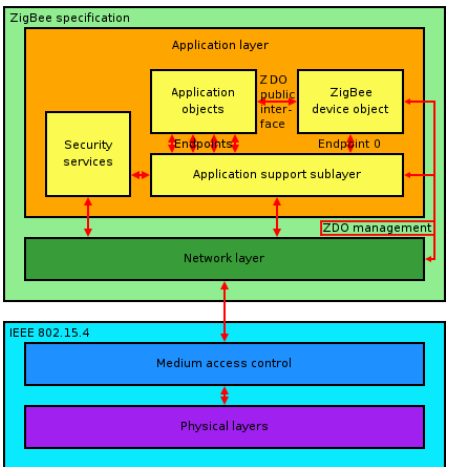


Figura 3-21 Pila de protocols integrant 802.15.4 i Zigbee [32]

Pel que fa a la capa de xarxa, per una banda encamina amb el protocol AODV [36] (*Ad-hoc On Demand distance vector*), que el que bàsicament fa es anar propagant un missatge per descobrir la ruta cap a un destí. Quan el missatge arriba al destí, es retorna la ruta per la quan ha viatjat de forma que el node emissor sap per a quina ruta enviar la informació.

Per altre banda, també s'encarrega de descobrir nous nodes i determinar si pertanyen o no a la mateixa xarxa.

La capa superior, Aplicació, està formada per diferents blocs:

- *Zigbee Device Object* o ZDO: assigna rols als nodes (coordinador, router o *end device*) i s'ocupa de descobrir nous nodes i d'identificar els serveis que pugin oferir.
- *Application Support Sublayer* o APS: fa de pont entre la capa de xarxa i la resta de components de la capa d'aplicació, manté les taules d'enrutament per a les necessitats d'aplicacions específiques.
- *Application Objects:* fan referència a les diferents aplicacions que puguin operar amb Zigbee.
- *Security Services:* Ofereix l'opció de xifrat d'informació mitjançant claus.

3.3.3. Descripció Xbee

Anteriorment ja hem introduït les característiques tècniques i d'altres més generals dels mòduls Xbee. En aquesta secció explicarem en més detall el seu funcionament.

La topologia de la xarxa creada amb aquest model de Xbee és una xarxa mesh. S'ha escollit així per a la facilitat a l'hora d'afegir nous nodes, ja que no calen configuracions addicionals.

Recordem que els mòduls Xbee permeten dos modes de funcionament: un mode "transparent" i un altre basat en "API" (*Application Programming Interface*). Amb el primer el mòdul actua com un port sèrie sense fils; mentre que amb el segon podem programar una aplicació que interactuï amb les capacitats de xarxa del mòdul. Per exemple, podem extreure l'adreça d'un mòdul, gràcies a que la informació s'encapsula en trames que així ho permeten.
Aquesta secció l'hem dividit en dues subseccions que cobreixen els aspectes fonamentals per entendre el funcionament dels mòduls Xbee.

3.3.3.1. Comunicacions sèrie µC ↔ Xbee

Aquesta subsecció es dedica a descriure com es comunica el mòdul Xbee amb l'Arduino. La figura 3-22 mostra l'esquema de comunicació sèrie entre el mòdul Xbee i un microcontrolador com el ATmega 328 de l'Arduino. En la figura es mostra tant el procés d'enviament de dades µC → Xbee i al revés.
.

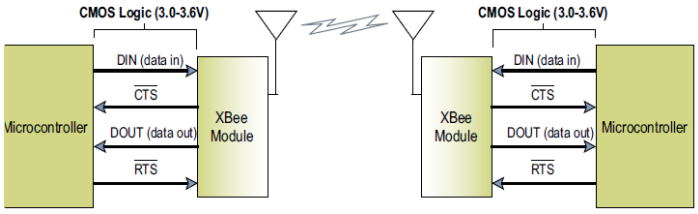


Figura 3-22 Esquema de comunicació sèrie entre Xbee i microcontrolador (Arduino en el nostre cas) [40]

Amb aquest esquema, el pin DIN del Xbee està connectat al pin de transmissió de dades sèrie del Arduino. El pin DOUT del Xbee està connectat al pin de recepció de dades sèrie del Arduino. Aquesta connexió es fa a través d'una altre placa Xbee Shield [41], com la mostrada a la imatge següent.

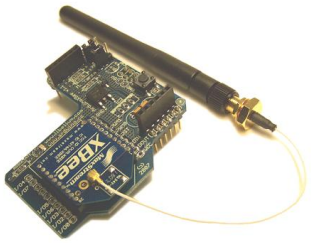


Figura 3-23 Xbee Shield amb mòdul Xbee, com els emprats en aquest PFC [41]

A la figura 3-22 apareixen els noms $\overline{CTS}$ i $\overline{RTS}$ que controlen el flux de dades cap als buffers de transmissió i recepció del Xbee. El CTS el que fa es impedir que l'Arduino enviï més dades al Xbee si aquest té el buffer de recepció de dades sèrie ple. El RTS fa la mateixa funció però a la inversa, és a dir, no envia dades des del Xbee fins l'Arduino si aquest darrer no li permet.

La figura 3-24 mostra l'esquema de blocs pel que fa als buffers del Xbee, i la relació amb el control de flux anteriorment mencionat.

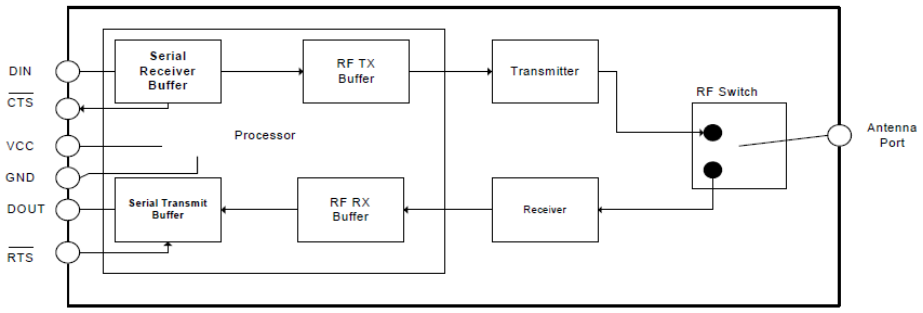


Figura 3-24 Esquema de blocs pel flux de dades al mòdul Xbee [40]

A la figura anterior podem observar el comportament anteriorment mencionat: si el buffer de recepció de dades sèrie del Arduino està ple, el CTS ho informa al

processador parant l'enviament de dades. Quan el buffer de recepció de dades sèrie està ple, les envia al buffer de transmissió.

Si es reben dades, aquestes són demodulades i enviades al buffer de transmissió de dades sèrie cap a l'Arduino. El RTS és el que permet que les dades rebudes pel Xbee siguin enviades al Arduino a través de DOUT.

3.3.3.2. Routing de la informació

En aquesta subsecció descrivim com els mòduls Xbee encaminen les trames i com descobreixen noves rutes.

En primer lloc, explicarem tres conceptes bàsics:

- Els dispositius poden ser configurats com *end device* o routers. Els primers no encaminen informació i els segons sí, per tant, per establir una xarxa mesh tots els nodes han d'actuar com routers.
- Identificador de xarxa, que agrupa un conjunt de nodes sota la mateixa adreça de xarxa. Nodes amb diferents identificadors de xarxa no establiran comunicació entre ells.
- Canal, que és la freqüència a la que operen els mòduls. Com passa amb l'identificador de xarxa, el canal ha de ser comú si es vol que els nodes es comuniquin entre ells.

Abans d'enviar un paquet i si no hi ha cap ruta establerta cap al node destí, el protocol que incorpora Xbee per xarxes mesh realitza un procés de descobriment de la ruta cap al node destí.

Aquest procés es realitza de la següent manera: en primer lloc el node origen realitza una petició de ruta per a una adreça destí determinada, que és transmesa a tots els nodes de la xarxa dins la distància d'un salt (és a dir, dins el radi de cobertura per part del node origen). En el missatge va inclosa l'adreça del node destí, de forma que si el pròxim node no té la mateixa adreça, reenvia el missatge de petició de ruta a tots els nodes que té a distància un salt.

El procés es repeteix fins que el missatge arriba al node destí, que llavors respon a l'adreça del node origen amb la ruta per a la qual li ha arribat. Aquesta resposta la pot enviar tants cops com peticions de ruta li arribin dels diversos nodes als quals hi tingui accés (cobertura). Això fa que el node origen tingui múltiples rutes per arribar a un mateix destí, tot i que només enviï les dades per a la que ofereixi menys retard en les respostes. La petició de ruta només serà executada un cop per adreça destí dels paquets. Si arriben paquets amb adreces destí diferents, tornarà a fer les peticions de ruta.

Finalment, els nodes entremitjos poden retransmetre o descartar una petició de ruta, depenent de si la nova petició ofereix unes millors mètriques en quant a retard en resposta. Si és així, la ruta antiga és descartada i la taula d'encaminament s'actualitza amb la nova, que també és retransmesa als demés nodes.

El descobriment de les rutes es fa d'acord amb el protocol *Ad-hoc On Demand Distance Vector* o AODV [36], de forma que la xarxa es manté en silenci fins que no hi ha una petició d'enviament de paquet. Als nodes es guarda una taula amb una

associació entre adreça destí i adreça del node del pròxim salt, pertanyent a la ruta establerta.

3.3.1. Esquema final plataforma WSN

La següent figura resumeix en un diagrama de blocs la plataforma WSN per SIAC.

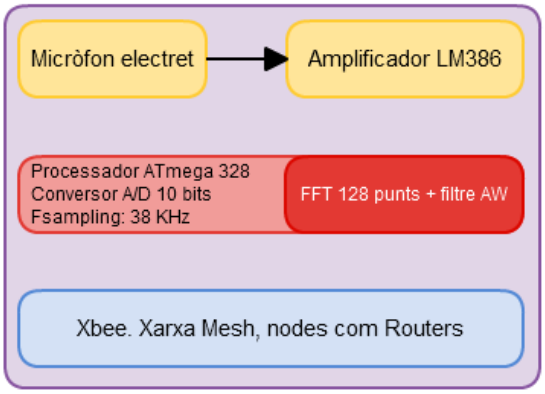


Figura 3-25 Esquema de blocs final de la WSN, indicant components

En el pròxim capítol, realitzem les proves del sistema en base a les suposicions d'aquest capítol. Dissenyarem el circuit d'adaptació del senyal del micròfon, establirem el guany de l'amplificador d'àudio, realitzarem el processat digital del senyal i finalment farem proves amb l'enviament de dades de prova i de les dades acústiques.

CAPÍTOL 4. RESULTATS EXPERIMENTALS

L'objectiu del capítol és mostrar les capacitats de la plataforma WSN com a xarxa de suport als sonòmetres per SIAC. També té com objectiu mostrar tot el procés de captació, processat i enviament de dades.

4.1. Processat de dades

En aquest apartat les proves han set realitzades amb el circuit pel micròfon explicat al Capítol 3, secció 3.2.1. Per evitar problemes de capacitats paràsites amb la protoboard, el circuit s'ha soldat sobre una placa, com mostra la figura 4-1.

Figura 4-1 Placa amb micròfon i circuit d'adaptació del senyal

Per provar les capacitats del sistema s'ha emprat un software de generació de tons. La resta de configuracions del sistema s'aniran explicant al llarg d'aquest apartat.

En aquest apartat expliquem tant el processat analògic com el digital del senyal. Comencem pel processat analògic, corresponent a la interfície electrònica per l'adaptació de la senyal provinent del micròfon a l'entrada del mòdul Arduino.

Abans de passar a l'explicació de cada una de les seccions, hi ha que dir que les gràfiques es veuran representen el nivell de senyal (0 a 1023) per mostra, i no els volts o la pressió acústica.

4.1.1. Processat analògic

La figura 4-2 mostra l'esquema del circuit d'adaptació del senya. Es mostra el micròfon, la resistència de polarització i l'amplificador LM386. També es presenten els valors dels components.

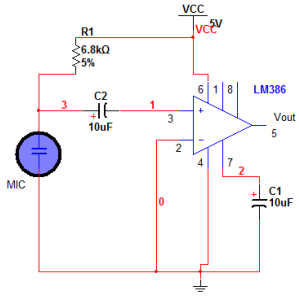


Figura 4-2 Esquemàtic del circuit d'adaptació del senyal

El primer que hem de fer és escollir el valor de la resistència de polarització, que farà passar més o menys corrent cap al FET que incorpora la càpsula del micròfon.

El ideal és tenir el senyal centrat a 2.5 V, ja que el marge d'entrada del Arduino és de 0-5 V. La següent figura 4-3 mostra com variant la resistència tenim més o menys offset a la sortida del micròfon.

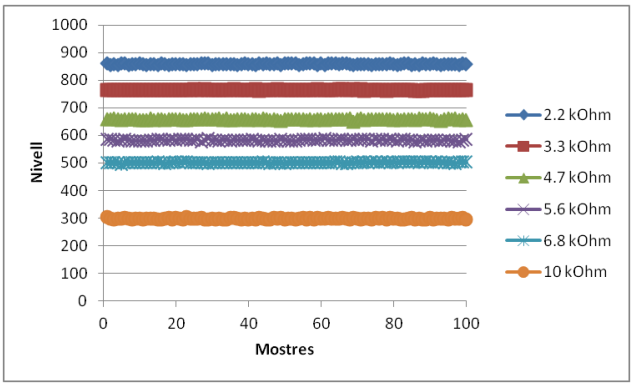


Figura 4-3 Nivell del senyal en funció de la resistència de polarització

Segons la figura anterior, el valor més òptim és per un valor de 6.8 kOhms.

La figura 4-4 mostra un to d'1 kHz mostrejat amb Arduino a la sortida del micròfon. El senyal surt efectivament centrat sobre els 2.5 V.

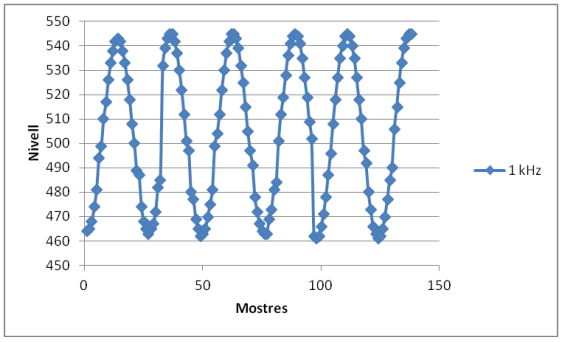


Figura 4-4 Captura d'un to a 1 kHz a la sortida del micròfon

De la figura anterior podem extreure l'amplitud del senyal per ajustar el guany de l'amplificador. En la següent expressió, el nivell de quantificació fa referència a l'amplitud del senyal i V_{Ref} al voltatge de referència del Arduino. Recordem que el conversor té 10 bits, per tant, 1023 nivells de quantificació del senyal.

$$V = \frac{nivell\ de\ quantificació \cdot V_{ref}}{nivells\ de\ quantificació\ totals} = \frac{(545 - 499) \cdot 5}{1023} \cong 225\ mV_p \qquad \textbf{(4.1)}$$

El guany de l'amplificador ha de ser:

$$G = \frac{V_{out}}{V_{in}} = \frac{5V}{225 \cdot 2\ mV} = 11.1 \quad \textbf{(4.2)}$$

El guany necessari és més baix que el guany mínim que ens pot proporcionar el LM386, de forma que amb la configuració pel guany de 20 ja n'hi ha prou.

Finalment, la combinació resistència – condensador forma un filtre passa altes. L'expressió 4.3 mostra la freqüència de tall d'aquest filtre, a on veiem que la component contínua no passa.

$$f_{tall} = \frac{1}{2\pi RC} = \frac{1}{2\pi \cdot 6.8\ K\Omega \cdot 10\ \mu F} \cong 2.34\ Hz \quad \textbf{(4.3)}$$

La figura 4-5 mostra una comparació entre la senyal d'entrada i la de sortida. Veiem que estem saturant a la sortida, però això és degut a l'alt nivell del senyal d'entrada. Si baixem el nivell (baixar el volum o allunyar la font de so), el micròfon perd sensibilitat i arriba un punt que no és capaç de detectar les variacions. Per millorar el sistema s'hauria d'adquirir un micròfon amb una sensibilitat més alta.

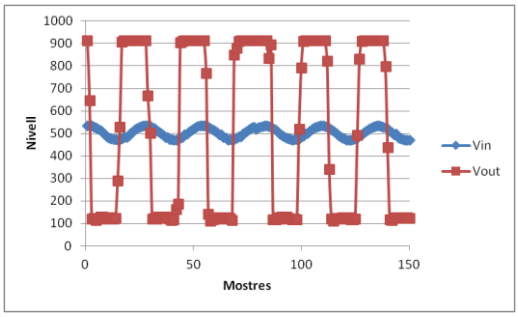


Figura 4-5 Senyal d'entrada al amplificador (Vin) vs. senyal de sortida (Vout)

Les gràfiques d'aquesta secció s'han extret amb Arduino. El codi per fer-ho s'explicarà més endavant juntament amb els altres codis del processat de dades.

4.1.2. Processat digital

Una vegada explicat el processat analògic, passarem al processat digital fet per la plataforma Arduino. Comentar que aquest processat s'ha aconseguit realitzar en un sol node, però a expenses de reduir la resolució de la FFT. Si es vol més resolució, s'haurà de distribuir el càlcul.

Aquesta secció s'estructura de la següent manera:

- Captació i visualització de dades temporals
- FFT de les dades temporals
- Verificació procés FFT – IFFT
- Captura espectre complet. Màxim ample de banda de l'aplicació

- Filtrat digital: aplicació de les corbes A-Weighting.

4.1.2.1. Captació i visualització de dades temporals

Com hem comentat a la secció anterior, aquí s'explica la part de processat del senyal que ha permès obtindre les gràfiques 4-3 a 4-5.

Fem una petita prova amb el següent codi, que captura les dades pel pin 2 del conversor A/D:

```
void setup() {
  // inicialitzem el port sèrie (9600 bauds/s)
  Serial.begin(9600);
}

void loop() {
  // escrivim pel port sèrie el que capta per l'entrada analògica A2 (canal número 2 del ADC)
  Serial.println(analogRead(A2));
  // afegim un retard abans de fer la següent iteració, per assegurar que el hardware ha capturat correctament una mostra i pot començar a convertir-la.
  delay(10);
}
```

En aquest codi podem observar les dues funcions bàsiques que conformen qualsevol programa en Arduino: *setup* i *loop*. A la funció *setup*, com el seu nom indica, escrivim qualsevol cosa que tingui a veure amb la configuració del dispositiu, com per exemple la velocitat de transmissió pel port sèrie. En la funció *loop* escriurem les rutines pròpiament dites, ja que aquesta funció s'executarà periòdicament. La funció *setup* tan sols s'executa un cop des de que encenem el dispositiu.

Tot i que en les primeres proves no semblava haver-hi cap problema amb el processat, es va descobrir que si es feia servir la configuració per defecte del mòdul ADC, les dades no es capturaven d'acord a les previsions.

Per aquest motiu, es va afegir el següent codi tret de [58] (segment *Setup*), que modifica la freqüència de mostreig del ADC, adaptant-la a les necessitats del sistema:

```
//Prescaler
  //ADPS2 - ADPS1 - ADPS0 - Division Factor
  //0              - 0     - 0     ->2
  //0              - 0     - 1     ->2
  //0              - 1     - 0     ->4
  //0              - 1     - 1     ->8
  //1              - 0     - 0     ->16
  //1              - 0     - 1     ->32
  //1              - 1     - 0     ->64
  //1              - 1     - 1     ->128
  //Configure to Prescaler=32
  bitWrite(ADCSRA,ADPS2,1);
  bitWrite(ADCSRA,ADPS1,0);
  bitWrite(ADCSRA,ADPS0,1);
```

Amb aquest codi el que fem es modificar els bits ADPS0, ADPS1 i ADPS2 del registre de ADCSRA del pic ATmega 328. Això s'ha explicat en el Capítol 3, subsecció 3.2.2.1. Les funcions *bitWrite()* el que fan és posar a 0 o a 1 el bit ADPS corresponent, en el registre ADCSRA.

A la secció anterior mostrem un to a 1 kHz capturar amb aquesta configuració (figura 4-4).

A continuació, passem al següent esgraó i computem la FFT d'un to a 1 kHz.

4.1.2.2. FFT de les dades temporals

Les primeres proves per fer la FFT es van fer amb els codis de [52] (extret de [53]) i [54]. Tot i això, al no poder verificar clarament si la IFFT tornava les dades originals es va decidir implementar una altre rutina per a la FFT, que va verificar la seva inversa.

La FFT, implementada amb la rutina de [51], la fem amb 128 punts com a màxim, ja que amb més nombre de punts la plataforma Arduino es queda sense memòria i no pot processar les dades (recordem que té 2 KB de memòria tan sols).

Això és degut al tipus de dades que fem servir per declarar les variables en el codi per calcular la FFT. Al fer servir variables de tipus *double* (per permetre coma flotant), estem consumint molta memòria en cada declaració. Amb això estem deixant poc espai per a que les pròpies rutines de l'Arduino puguin operar de forma correcta.

Si fem servir tipus de dades que no consumeixin tanta memòria (*int* per exemple), augmentem el nombre de punts de la FFT. Tot i això, perdem precisió en els resultats, fins al punt que al fer el procés FFT – IFFT, les dades de sortida d'aquest procés no són les mateixes que les d'entrada.

La següent figura 4-6 mostra que el to (pic més alt) es troba a k = 16. Segons les expressions 3.7 i 3.9 del Capítol anterior, i que aquí recordem, el to a 1 kHz hauria de situar-se per $k \in [3{,}4]$.

$$W_{min} = \frac{2\pi}{NT} \rightarrow f_{min} = \frac{1}{NT} \tag{4.4}$$

$$f_k = \frac{1}{NT} k \rightarrow$$

$$k = f_k \cdot NT = 1000\,Hz \cdot 128 \cdot 26\,\mu s = 3.328 \tag{4.5}$$

Aquesta figura recull les dades obtingudes amb la configuració per defecte del ADC de l'Arduino, a on k = 16 de forma que queda justificada la modificació de la freqüència de mostreig.

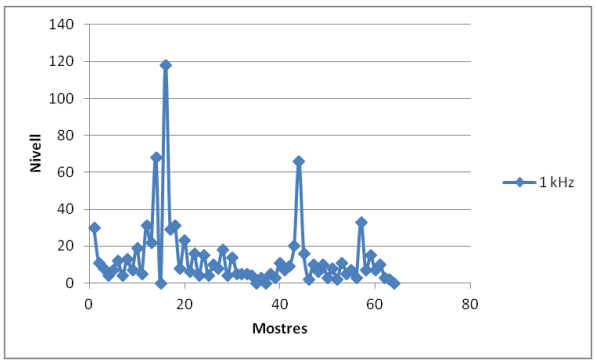


Figura 4-6 Espectre d'un to a 1 kHz mostrejat incorrectament

Si mostregem diversos tons, els resultats també són incoherents, com mostrem a la gràfica següent, a on el to a 10 kHz està al costat del d'1 kHz:

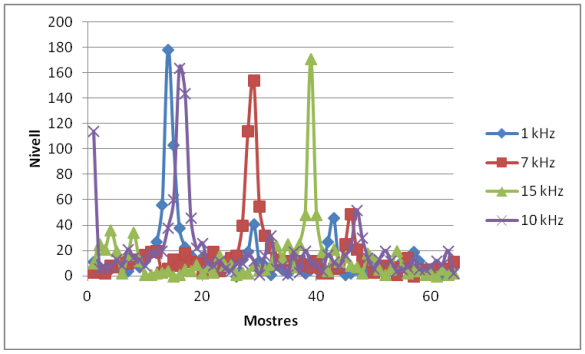


Figura 4-7 Espectre de diferents tons mostrejats incorrectament

La figura següent mostra el mateix to d'1 kHz, però mostrejat amb la configuració explicada a la subsecció anterior 4.1.2.1. Com podem observar, el pic més alt es troba a k = 4, de forma que amb la nova configuració determinem correctament l'espectre d'un senyal.

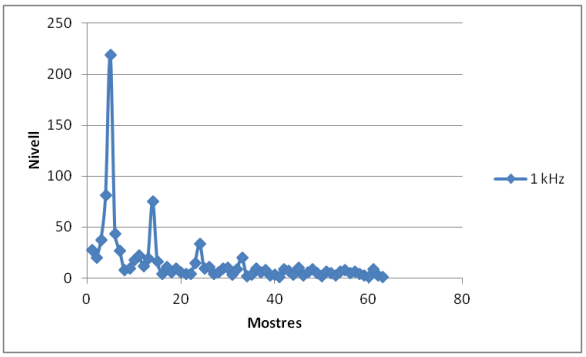


Figura 4-8 Espectre d'un to a 1 kHz mostrejat correctament (apareixen harmònics)

Com a contraposició a la figura 4-7, la figura 4-9 mostra l'espectre sonor mostrejat correctament:

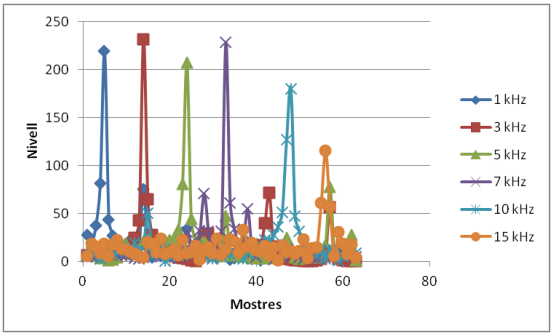


Figura 4-9 Espectre de diferents tons mostrejats correctament

Al fer aquest mostreig, comprovem que la màxima freqüència que realment podem mostrejar són 15 kHz, no els aproximadament 19 kHz que teòricament podíem mostrejar per Nyquist.

El la figura 4-9, les diferències en amplitud dels pics són degudes a que els altaveus mitjançant els quals s'han fet aquestes proves no tenen la mateixa resposta per a totes les freqüències.

Tant el codi com les llibreries per fer la FFT es poden trobar als annexes.

Com a conclusió tenim que per mostrejar correctament amb Arduino, s'ha de configurar el prescaler segons les necessitats de l'aplicació.

4.1.2.3. Verificació procés FFT - IFFT

Tot i haver validat la certesa en els resultats anteriors, hi que comprovar que la rutina emprada per a la FFT sigui reversible, és a dir, que les dades a l'entrada de la rutina siguin les mateixes una vegada fet el procés invers.

Les següents dades temporals han set extretes fent el procés de captura, FFT i IFFT. No hem aplicat cap to, simplement són del soroll ambient. Com podem comprovar, les dades a l'entrada de la rutina, són les mateixes al fer la IFFT, indicant que la rutina està ben implementada.

```
Dades IN:839.00 --- Dades OUT:839.00
Dades IN:744.00 --- Dades OUT:744.00
Dades IN:707.00 --- Dades OUT:707.00
Dades IN:279.00 --- Dades OUT:279.00
Dades IN:916.00 --- Dades OUT:916.00
Dades IN:916.00 --- Dades OUT:916.00
Dades IN:904.00 --- Dades OUT:904.00
Dades IN:565.00 --- Dades OUT:565.00
Dades IN:573.00 --- Dades OUT:573.00
Dades IN:455.00 --- Dades OUT:455.00
Dades IN:903.00 --- Dades OUT:903.00
Dades IN:884.00 --- Dades OUT:884.00
Dades IN:818.00 --- Dades OUT:818.00
Dades IN:263.00 --- Dades OUT:263.00
Dades IN:590.00 --- Dades OUT:590.00
Dades IN:479.00 --- Dades OUT:479.00
```

Cal indicar que si les variables a on guardem les dades capturades són de tipus *integer* (sense coma flotant), les dades a l'entrada no són les mateixes que a la sortida. Això podria fer pensar erròniament que la rutina està mal implementada, quan el que passa és un problema de precisió i càlcul amb els tipus de dades definits.

4.1.2.4. Filtre A-Weighting

La figura 4-10 mostra part de les components espectrals de la figura 4-9 processades amb les corbes A-Weighting. El que s'ha fet és multiplicar l'espectre de sortida per la taula de valors corresponents. Com es pot apreciar, la freqüència d'1 kHz quasi no queda atenuada, mentre que la resta sí.

Aquesta atenuació a freqüències properes a 1 kHz es compensa afegint, en forma logarítmica, un factor constant, de forma que compensa l'atenuació.

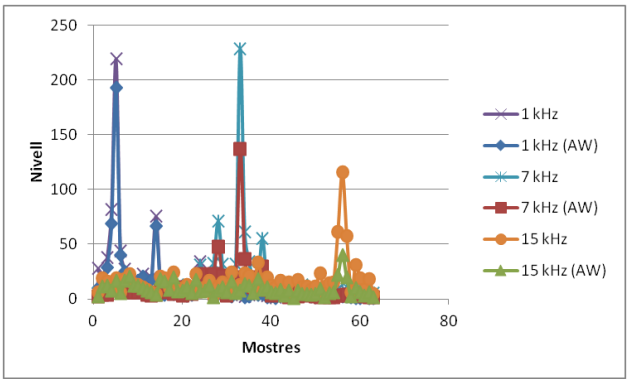


Figura 4-10 Comparació espectre amb filtre AW i sense

4.2. Comunicacions

En aquest apartat descrivim les proves realitzades per caracteritzar les comunicacions entre els nodes. Ja hem comentat que emprarem els mòduls Xbee de Digimesh, i que la topologia de la xarxa serà de tipus mesh.

En aquest apartat primer mostrem les configuracions dels nodes i després les proves que s'han fet: primer verificar les comunicacions enviant lletres i a continuació enviament de les dades capturades pel Arduino. També es fa una prova per verificar que podem identificar nodes a través de les adreces MAC.

La funció dels nodes Xbee (XBx a on x indica el número de node per identificar-los durant l'explicació de les proves) és la següent:

- XB3: envia dades
- XB2: envia dades
- XB1: rep dades

Això s'ha fet així per simular que els nodes XB3 i XB2 formen part de la WSN i envien les dades a un node gateway representat pel XB1. Recordem que aquest node

gateway utilitzaria una altre enllaç, per exemple 3G, per enviar la informació de la WSN a la BBDD.

La figura 4-11 és un esquema general de les proves, a on es mostra quins nodes són els que envien i quin és el que rep. En aquesta figura, les adreces de destí no són les del pròxim node. Les adreces destí representarien l'adreça final del *gateway* comentat.

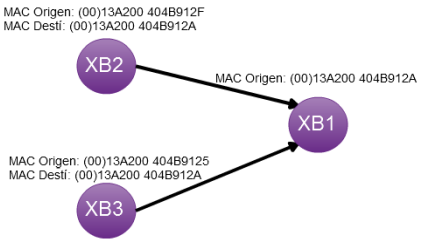


Figura 4-11 Esquema general de les proves

Tots tres nodes tenen l'API (API = 0) per defecte, el que implica que els mòduls es comporten com "un port sèrie" però sense fils.

Les adreces dels nodes són les següents:

1. XB1: (00)13A200 404B912**A**
2. XB2: (00)13A200 404B912**F**
3. XB3: (00)13A200 404B912**5**

En negreta estan marcades les diferències entre les adreces, per evitar confusions.

Finalment, comentar que tot i haver fet proves punt a punt, es van fer proves en la que el XB3 tenia com adreça destí la del XB1, i la informació enviada havia de ser encaminada a través de XB2. Això ens va permetre comprovar que efectivament els mòduls són capaços de descobrir rutes quan el node que tenen en el pròxim salt no és el node destí.

4.2.1. Configuracions dels nodes Xbee: API = 0

Els paràmetres més importants i bàsics per a la configuració dels nodes són el canal, l'adreça de xarxa, les adreces de destí i el tipus de node (*end device* o router).

La següent taula mostra els valors de les configuracions comunes. Les més importants pel correcte funcionament de la xarxa són el canal i l'adreça de xarxa. El tipus de node també és important si es fa una xarxa mesh.

Taula 4-1 Valor dels paràmetres bàsics per a la configuració de la xarxa

Paràmetre	Valor
Freqüència central	2400 MHz
Canal	12 (C en hexadecimal)
Adreça de xarxa	7FFF
Tipus de node	0 (router, per defecte)
Tipus de API	0 (per defecte)

A continuació, passem a mostrar les configuracions de cada mòdul. Les captures estan extretes del software XCTU, que a més d'actuar com hiperterminal, permet llegir i escriure els paràmetres per configurar els nodes, així com el firmware dels mateixos.

A les captures podem observar el valor dels paràmetres mostrats a la taula 4-1. A més es mostren, en els nodes XB3 i XB2, les adreces destí pertanyents al node que simula el *gateway* de la nostra xarxa.

- Configuració XB3:

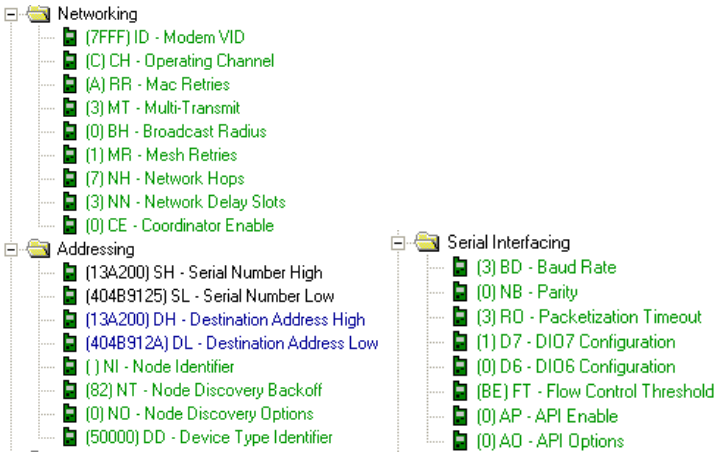


Figura 4-12 Configuració XB3

- Configuració XB2:

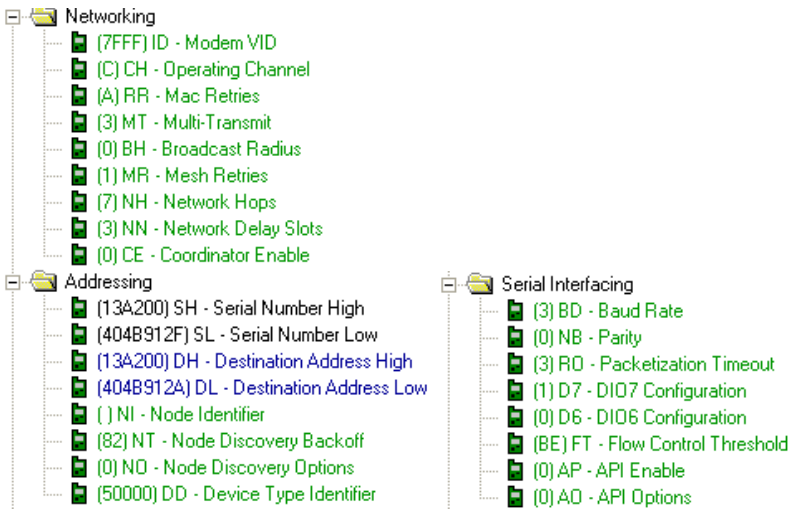


Figura 4-13 Configuració XB2

- Configuració XB1

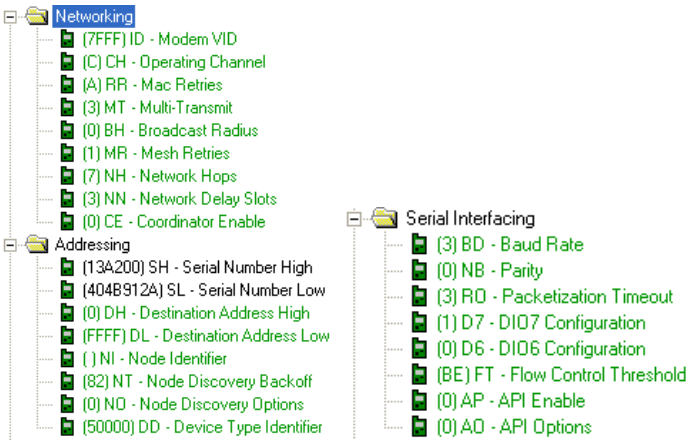


Figura 4-14 Configuració XB1

Per a les proves realitzades, hem deixat pràcticament tots els paràmetres de la taula 4-1 anterior per defecte. L'única diferència, en configuració, entre els nodes són les adreces de destí.

Com que la topologia és de tipus mesh, tots tres nodes serveixen de routers i no hi ha nodes '*endpoint*' o '*coordinator*', ja que no tindria sentit en una xarxa mesh.

Una vegada mostrades les configuracions, podem passar a realitzar la bateria de proves.

4.2.1.1. Prova 1. Enviament de dues lletres (H i L) entre dos nodes. Mostra de la comanda ND (Network Discovery) per descubrir nodes associats a la xarxa.

Aquesta prova es basa en enviar dues lletres (H i L) entre una placa Arduino amb mòdul Xbee i un segon mòdul Xbee.
També inclou una mostra de l'ús de la comanda ATND, que serveix per descobrir quins mòduls té associat un determinat node.

Consideracions prèvies:

- ABx: Placa Arduino número x (x = 1, 2 o 3).
- XBx': Placa Xbee número x' (x' = 1, 2 o 3).
- Cada ABx té el seu equivalent XBx', a on x = x'.

La prova es realitza entre AB3 i XB1. El muntatge és el següent:

- AB3 amb el següent codi que envia les lleters H i L en intervals d'un segon:

```
void setup()
  {
    Serial.begin(9600);
  }

  void loop()
  {
    Serial.print('H'); //enviem lletra H
    delay(1000); // esperem un segon (1000
milisegons)
    Serial.print('L'); //enviem lletra L
    delay(1000); //esperem
  }
```

- XB3 configurat per a enviar les dades al XB1 (adreça destí XB3 = adreça XB1).
- AB3 amb XB3
- XB1 tot sol connectat al PC, sense placa Arduino. Capturem les dades amb un Gateway de libelium, i les monitoritzem amb software XCTU que actua d'hiperterminal (específic de Digimesh).

La figura 4-15 mostra l'esquema de la prova.

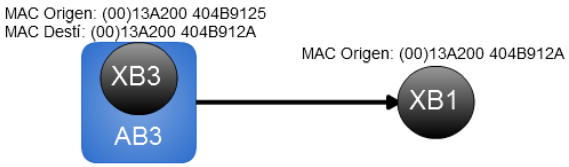


Figura 4-15 Esquema de la prova 1

Amb el programa mencionat, observem com arriben a intervals d'un segon les lletres enviades pel AB3. Les dues captures corresponen una als primers instants i l'altre a instants més tard.

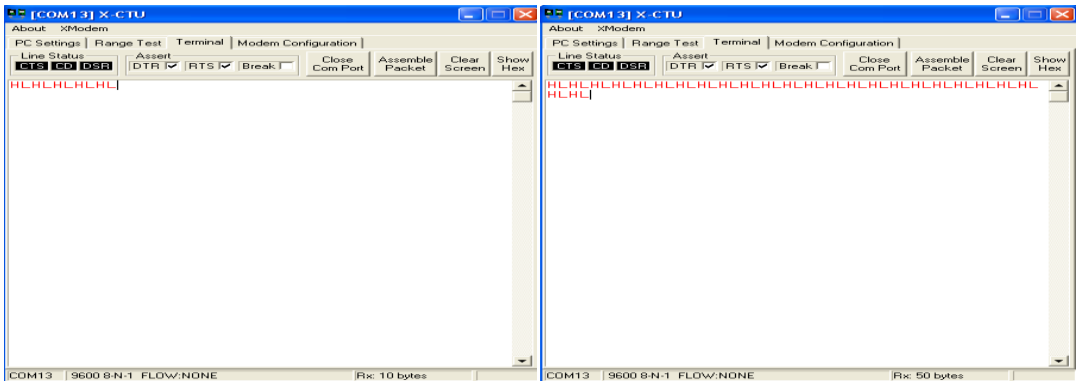


Figura 4-16 i **4-17** Captura amb XCTU de les dades rebudes en diferents instants de temps

La següent captura mostra el resultat de la comanda ATND (Network Discovery):

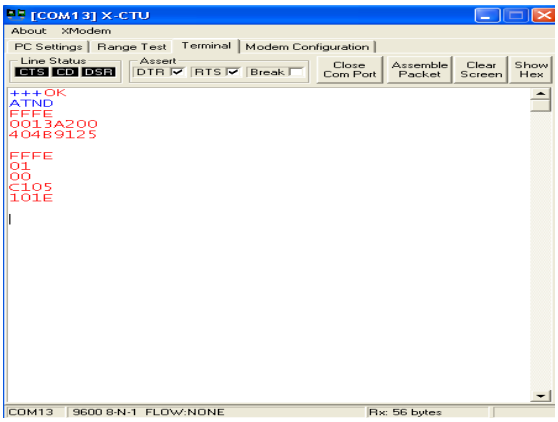


Figura 4-18 Descobriment d'un node a través de la comanda ATND

En la captura també es mostra la forma en la que s'introdueixen comandes per a que els mòduls l'executin: primer cal enviar al mòdul, que està connectat pel port sèrie, tres +++. Amb això l'indiquem que estigui preparat per rebre comandes pel port sèrie.

El mòdul respon amb un OK i a continuació enviem la comanda ATND, que mostra el següent:

Taula 4-2 Paràmetres i valors mostrats a la figura 4-17

Paràmetre	Valor segons figura 4-17
Adreça curta (16-bits) del node	FFFE (hexadecimal)
Adreça MAC del node (32 bits alts)	0013A200 (hexadecimal)
Adreça MAC del node (32 bits baixos)	404B9125 (hexadecimal)
Adreça de xarxa "pare"	FFFE

El valor de l'adreça curta del node indica que l'encaminament es farà amb les adreces MAC. El valor de la xarxa "pare" indica que de fet aquesta tipologia no s'aplica i que no hi ha relació pare – fill entre ells nodes.

Finalment, les dades recollides per aquesta comanda mostren que efectivament el node descobert és el XB3.

4.2.1.2. Prova 2. Enviament de dues lletres (H i L) entre dos nodes complets (Arduino més Xbee)

Aquesta prova segueix la mateixa filosofia que l'anterior, tret que ara enviem i recuperem dades entre mòduls Arduino (figura 4-19). L'objectiu és fer una primera aproximació a la identificació i tractament de dades enviades entre mòduls.

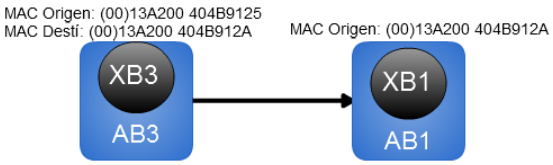


Figura 4-19 Esquema de la prova 2

La prova es realitza entre AB3 i AB1. El muntatge és el següent:

- AB3 amb el següent codi que envia les lleters H i L en intervals d'un segon.

```
void setup()
  {
    Serial.begin(9600);
  }

  void loop()
  {
    Serial.print('H');
    delay(1000);
    Serial.print('L');
    delay(1000);
  }
```

- XB3 configurat per a enviar les dades al XB1.
- AB3 amb XB3.
- AB1 amb el següent codi, connectat al PC:

```
int incomingByte;      // variable a on guardem les dades rebudes

void setup() {
  Serial.begin(9600);
}

void loop() {
  // mirem si hi hem rebut dades:
  if (Serial.available() > 0) {
    // llegim el darrer byte rebut:
    incomingByte = Serial.read();
    // si hem rebut una H, que ho escrigui per pantalla:
    if (incomingByte == 'H') {
      Serial.print("Lletra rebuda! (H ASCII 72) :");
      Serial.println(incomingByte);
    }
    // si hem rebut una H, que ho escrigui per pantalla:
    if (incomingByte == 'L') {
      Serial.print("Lletra rebuda! (L ASCII 76) :");
      Serial.println(incomingByte);
    }
  }
```

- AB1 amb XB1.

Ara capturem les dades amb el port sèrie del AB1. La següent imatge mostra que efectivament estem rebem les lletres H i L (codis ASCII 72 i 76 respectivament).

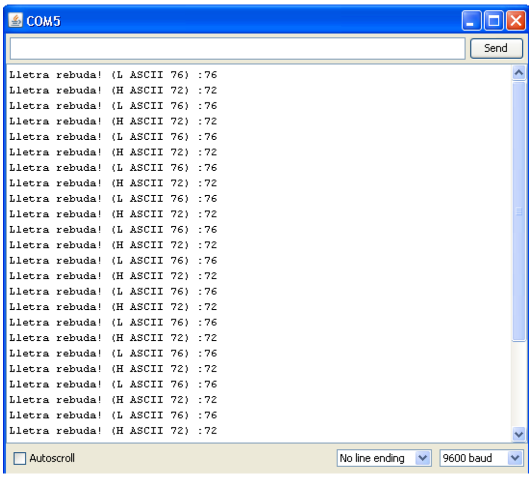


Figura 4-20 Captura de les dades. Recuperem les lletres H i L en codi ASCII

4.2.1.3. Prova 3. Enviament de les dades de la FFT.

Amb aquesta prova el que fem és enviar les dades de la FFT i testejar el sistema amb les dades que haurem de tractar.

Les configuracions dels nodes són les mateixes que en la prova anterior, tret del codi que fem servir al AB1, i el que fem servir al AB3 que és el de la rutina per calcular la FFT (Annexes, capítol 8 – Codis).
Si només capturem i mostrem les dades pel port sèrie, el que es mostra és el valor en ASCII del valor rebut (símbol 1 en decimal → valor 49 en ASCII). Això representa un problema ja que no obtenim directament el valor enviat pel AB3.

Per aquest motiu, es fa servir un codi [57] que permet mostrar els valors tal i com arriben, és a dir, valors no ASCII.

```
char cadena[24]; //string a on es guarda cada valor
int contador=0;
float valor=0; //el que es mostra per pantalla

void setup() {
  Serial.begin(9600);
}

void loop(){
  if(Serial.available()){
  memset(cadena, 0, sizeof(cadena));

  while (Serial.available()>0){
    delay(5);
    cadena[contador]=Serial.read(); //llegim cada valor i el guardem al 'buffer' cadena
    contador++;
  }
  valor=atol(cadena); //valor té el contingut de cadena
  Serial.println(valor);
  contador=0;
  }
}
```

La funció d'aquest codi és senzilla: emmagatzema dins *cadena* el valor rebut, dígit a dígit, formant un vector en el que cada posició del mateix és un dígit rebut. Així, si el valor enviat és 10, *cadena* tindrà dues posicions.

Valor és el valor no ASCII del nombre guardat dins el vector *cadena*.

La següent figura mostra les dades rebudes pel AB1, quan s'ha mostrejat un to a 3 kHz i s'ha realitzat la seva FFT pel AB3.

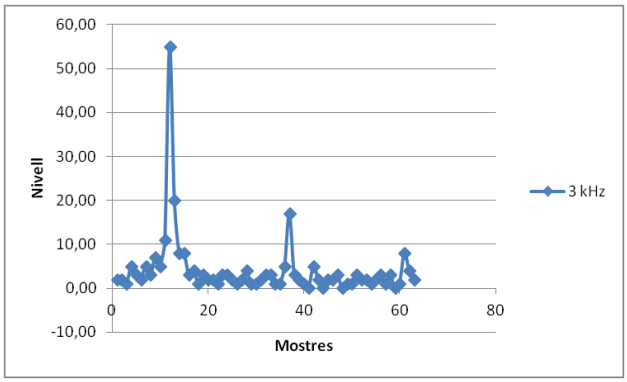


Figura 4-21 Espectre d'un to de 3 kHz enviat i recuperat pel AB1 correctament

Podem observar com el pic més alt és a la mateixa mostra que a la figura 4-9, a on s'havia mostrejat el to a 3 kHz en la mateixa placa AB3.

Gràcies a poder convertir els valors en ASCII llegits pel port sèrie, podem extreure les dades correctes i representar-les.

4.2.2. Configuració dels nodes Xbee: API = 2

L'objectiu d'aquesta secció és tractar amb l'API del Xbee i extreure més informació que no pas les dades. En concret, el que volem és identificar adreces de diferents nodes, per tal de realitzar càlculs com mitjanes de les dades rebudes per diversos nodes, per exemple. Aquesta identificació es porta a terme pel node AB1 i XB1 que simula el *gateway* en el nostre sistema.

La motivació d'aquesta prova, apart del comentat anteriorment, és poder verificar que podem crear "zones" dins la nostra xarxa, tot i mantenint la tipologia mesh. Verificar que podem agrupar dades acústiques d'una zona, fer la seva mitja i enviar-les cap a un altre node o cap al *gateway* final.

Per fer-ho, al Arduino AB1 carreguem les llibreries de Xbee [59] amb la qual cosa podem tractar amb els mòduls Xbee de la mateixa manera que els mòduls Arduino. La resta de nodes han mantingut la configuració per defecte de l'API.

Configuració:

- AB3 amb XB3: processem i enviem les dades (API = 0).
- AB3 amb el mateix codi que la prova anterior (rutina FFT).
- XB1: reb les dades (API = 2).
- AB1: amb el codi modificat de [59] (Annexes, capítol 8 - Codis), que interactua amb el XB1 per extreure'n l'adreça, les dades, i d'altres:

A continuació passem a explicar pas a pas aquesta prova. Això inclou anar presentant petites parts del codi carregat al AB1.

La següent imatge mostra el canvi en la configuración del XB1, a on la opció AP ara té el valor 2, indicant que fem servir API = 2:

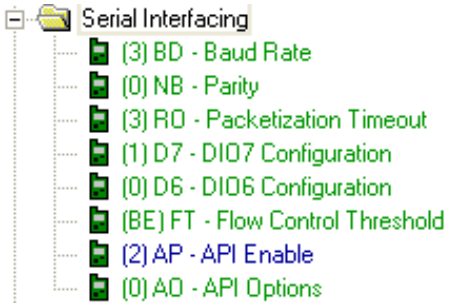


Figura 4-22 Detall per l'API = 2 al XB1

Amb aquesta configuració, podem interactuar amb els paquets que rebem, ja que estem habilitant poder interactuar amb les llibreries del mòdul Xbee a través de l'Arduino.
El codi de més avall (carregat al AB1) el que fa és indicar, quan demodula el paquet Dades2 Tx, si el ACK1 ha estat rebut pel transmissor quan ha enviat el primer paquet Dades1 Tx. És a dir, en el paquet N+1 s'informa si el ACK N va ser rebut pel Tx en enviar el paquet N.

El següent diagrama temporal de missatges és una explicació visual del que estem comentant.

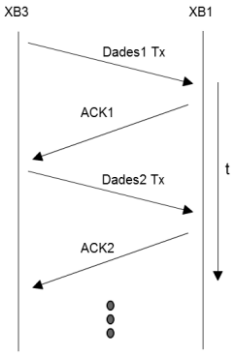


Figura 4-23 Diagrama temporal per l'enviament i confirmació de paquets

Codi carregat a AB1 (modificat de [59]):

```
xbee.readPacket();
//***** codi que s'executa un cop, per establir communicació entre nodes *******

   if (xbee.getResponse().isAvailable() && iter == 0) //si tenim algun paquet
   {

     if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
     {
       // demodulem el paquet
       xbee.getResponse().getZBRxResponse(rx);

       if (rx.getOption() == ZB_PACKET_ACKNOWLEDGED) {
           // el Tx ha rebut l'ACK
           Serial.println("ACK rebut pel Tx");
       } else {
           // Hem rebut una resposta, però l'ACK cap al Tx s'ha perdut.
           Serial.println("El Tx no ha rebut un ACK");
       }
     }
    // variables que fan que aquest codi només s'executi un cop
     iter=1;
     control=1;
   }
```

Les variables de control les posem simplement per a que aquest codi només mostri la informació un cop per pantalla.

En la següent captura de pantalla, extreta pel port sèrie del AB1, comprovem que efectivament el Tx ha rebut la confirmació del paquet.

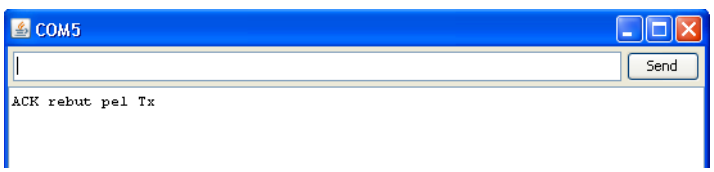


Figura 4-24 Comprovació de que el Tx ha rebut l'ACK

El següent de codi serveix per saber si tenim contacte o no amb l'altre node. Bàsicament serveix per avisar que el node Tx està desconnectat i no rep cap tipus de missatges.

```
if(control == 0)
    {
      // el XB3 està apagat o no està en el rang de cobertura de la xarxa
      Serial.println("NO HI HA CONTACTE");
    }
```

La següent captura de pantalla mostra el comentat anteriorment, a on s'ha deixat desconnectat l'AB3 i per tant, no podem establir comunicació entre ambdós nodes.

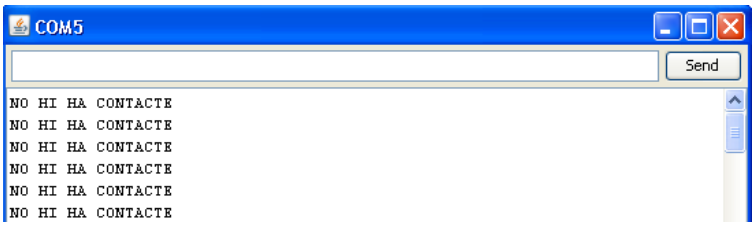


Figura 4-25 Captura quan no hi ha comunicació entre nodes

Si ara connectem l'AB3, després del procés per descobrir la xarxa i la ruta cap a l'altre node, veiem que aconseguim contactar amb el Tx.

La següent imatge mostra com després d'un temps d'estar connectat i d'haver detectat l'altre node, el Tx està llest per transmetre i el Rx per rebre.

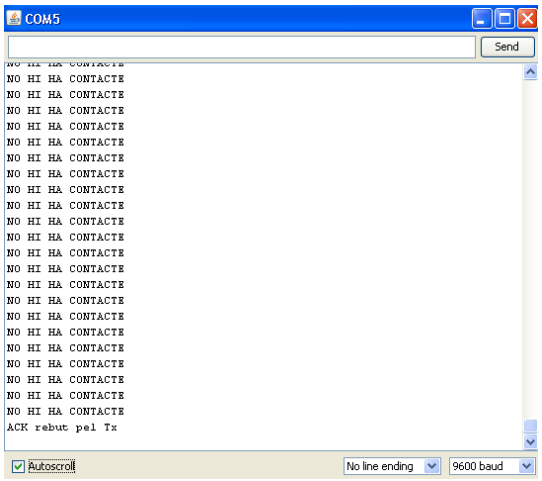


Figura 4-26 Captura per mostrar que fins que no descobreix la ruta, el TX no pot transmetre

El que passa és que fins que el XB3 (inicialment desconnectat) no està llest per emetre dades (ha de descobrir nodes i fer la taula de routing), el XB1 no rep cap paquet i per tant no envia cap ACK. Quan l'XB3 està llest, envia les dades i les confirma amb els ACKs del receptor.

Un cop sabem que tant Tx com Rx estan llestos, podem passar a mostrar les dades que conté cada paquet transmès.
La següent figura mostra el format de les dades amb API = 2, a on el segment *start delimiter* indica el inici del paquet; *length* és el nombre de bytes que conté el segment de dades; *frame data* que són les dades enviades i *checksum* per validar la integritat de les dades.

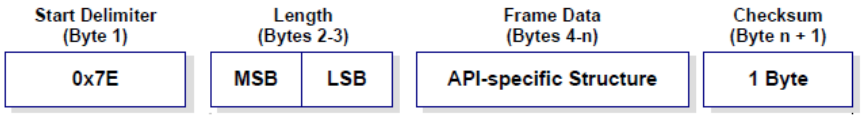


Figura 4-27 Format de la trama per Xbee amb API = 2 [40]

El fabricant no indica en cap moment com és el format de dades amb API = 0, i tampoc indica com transforma aquest format de dades (API 0, 1 o 2) a trama IEEE 802.15.4. En aquest punt en concret, treballem a nivell de API i el Xbee s'encarrega de transformar aquest format de dades a IEEE 802.15.4.

Comentar que el fabricant no indica en cap moment com és el format de dades amb API = 0, tipus utilitzat en les proves anteriors. Tampoc indica com es transporten les dades, és a dir el seu format, dintre les trames IEEE 802.15.4, sigui quina sigui l'API utilitzada.

Amb el següent segment de codi mostrem quines són les dades i l'adreça, a banda d'altres paràmetres com el *checksum*.

```
if(control!=0) { //començam a tractar les dades
      //Serial.println("preparat!");
      Serial.print("checksum is ");
      Serial.println(rx.getChecksum(), DEC);
      Serial.print("packet length is ");
      Serial.println(rx.getPacketLength(), DEC);

      for (int i = 0; i < rx.getDataLength(); i++) //payload
      {
       Serial.print("payload [");
       Serial.print(i, DEC);
       Serial.print("] is ");
       Serial.println(rx.getData()[i], DEC); //en el codi original és println //getData() = payload
        //payload[i] = int(rx.getData()[i]);
      }

     for (int i = 0; i <xbee.getResponse().getFrameDataLength(); i++) // frame: 64b address + receive options + received data
      {
        //adrs[i]=int(xbee.getResponse().getFrameData()[i+1]); //64 bit address stored in adrs[i], int type
        Serial.print("address frame [");
        Serial.print(i, DEC);
        Serial.print("] is ");
        Serial.println(adrs[i],DEC); //char
   Serial.println(xbee.getResponse().getFrameData()[i+1],DEC);
      }
      }
    }
```

A la següent captura veiem, en format hexadecimal, les dades que envia el XB3. Podem veure l'adreça del XB3 i les dades enviades pel AB3.

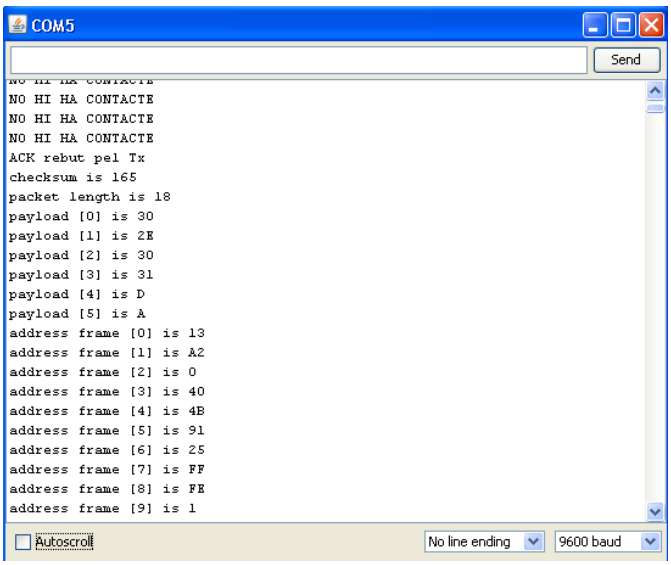


Figura 4-28 Mostra del paquet enviat pel XB3

Amb aquesta informació i el següent segment de codi, podem discriminar entre nodes basant-nos amb les adreces MAC.

```
String inString[] = {0x13,0xA2,0x00,0x40,0x4B,0x91,0x2F}; //adreça node XB2
String inString2[] = {0x13,0xA2,0x00,0x40,0x4B,0x91,0x25}; //XB3
[...]
for (int i = 0; i <7; i++)
      {
        if((adrs[i] == inString[i].toInt()) && (adrs[6] == inString[6].toInt()))
        {
          Serial.println("AB2");
        }
        if(adrs[i] == inString2[i].toInt() && (adrs[6] == inString2[6].toInt()))
        {
          Serial.println("AB3");
        }
      }
```

Aquest codi el que fa és comparar l'adreça guardada a *adrs[i]* i comparar-la amb el vector *inString[i]*. La condició és que primer, compari tota la adreça i després que miri la última posició, que és on es diferencien les tres adreces.

La imatge següent és una mostra del funcionament del codi anterior, a on hem discriminat positivament el node AB3 a través de la seva adreça.

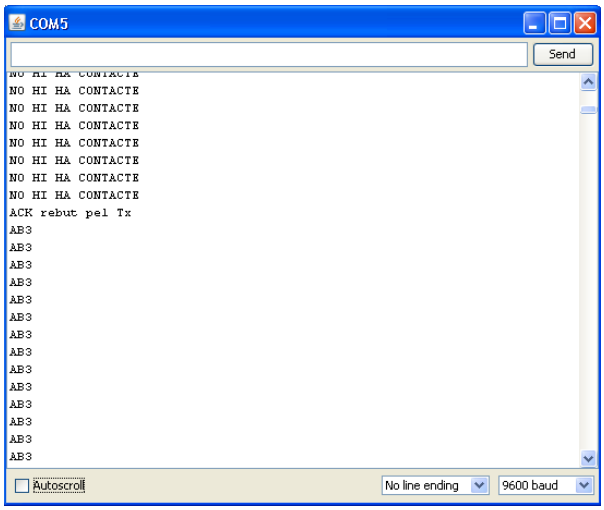


Figura 4-29 Verificació de que identifiquem correctament el node XB3

A l'identificar correctament els nodes, podem realitzar diversos càlculs com mitges a un node amb informació recollida d'altres nodes veïns. També podem distribuir el càlcul si volem més resolució a la FFT. També podem limitar el nombre de nodes associats a un altre node general, si volem crear "jerarquies" dins una xarxa mesh.

Finalment, la següent figura mostra la foto del sistema complet (node i gateway), que inclou la placa per adaptar la senyal del micròfon connectada al Arduino i el Xbee acoplat al Arduino amb el Xbee Shield. També es mostra el mòdul XB1 connectat a l'ordenador amb el gateway de Libelium.

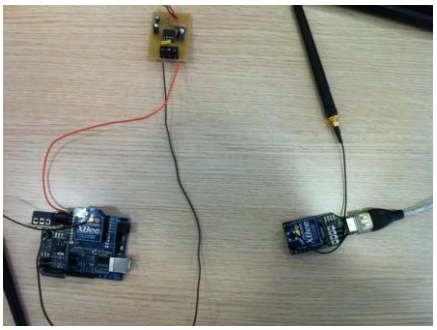


Figura 4-30 Sistema complet: Node amb sensor, Arduino i Xbee i mòdul XB1 fent de gateway

CAPÍTOL 5. CONCLUSIONS

En aquest projecte s'han estudiat les capacitats de la plataforma Arduino pel que fa a processat i comunicació de dades entre nodes. Això s'ha fet per poder dissenyar xarxes de sensor sense fils que siguin capaces d'adquirir dades acústiques, processar-les i enviar-les a un servidor central.

En primer lloc s'han comparat dues plataformes hardware per crear la WSN. Hem escollit la plataforma Arduino per ser de desenvolupament lliure i per tot el suport que es pot trobar tant a la xarxa com en publicacions en paper.

Com a plataforma per a comunicacions, al escollir Arduino hem hagut de treballar amb Xbee, que permet crear xarxes mesh, de forma que les dades poden viatjar per múltiples rutes. No hem creat figures de coordinadors de xarxa o similars, si no que tots els nodes poden enviar-se informació entre ells.

En segon lloc s'han estudiat les característiques dels micròfons com a sensors per a la captació acústica. S'ha descobert que els emprats en aquest projecte no tenen una sensibilitat adequada.

Seguidament, i com a part del sistema d'acondicionament del senyal, s'han analitzat les capacitats de diferents amplificadors operacionals per adaptar la sortida del micròfon a l'entrada del conversor A/D del Arduino. Se n'ha escollit un d'específic per aplicacions d'àudio, que presenta un guany molt estable en totes les freqüències d'interès.

En tercer lloc, s'han pogut implementar rutines per calcular la FFT. Aquesta FFT és de 128 punts, amb dades en coma flotant. Amb aquesta resolució som capaços de realitzar la FFT i demés tasques pel processat en un sol node, però si es vol incrementar la resolució hi ha que dividir el càlcul entre els nodes. No és recomanable emprar altres tipus de dades que no siguin en coma flotant, ja que es perd molta precisió en el resultat, fins al punt que el procés FFT – IFFT no es reversible. S'entén que això és així per la rutina FFT implementada. Altres rutines poden permetre més resolució.

També s'han implementat les corbes del A Weighting pel que fa a mesures de soroll ambient, ja que són les més emprades pels aparells comercials de captació de dades acústiques.

El processat digital ha fet que el ample de banda de l'aplicació sigui dels 300 Hz fins els 15 kHz, suficient per a la nostra aplicació.

Seguidament, s'ha verificat que amb els mòduls Xbee podem enviar/rebre les dades de la FFT amb total normalitat. En aquest projecte també s'han provat diferents configuracions en quan a la manera que es pot fer la interacció entre Arduino i Xbee. Es poden identificar paquets i per tant nodes, i considerar el càlcul distribuït.

Finalment, s'ha conclòs que podem emular les prestacions dels sonòmetres comercials, mitjançant una xarxa de sensors sense fils basada en Arduino i Xbee.

Futures tasques que han de seguir a aquest projecte són la verificació de que aquestes conclusions es mantenen si l'alimentació de la xarxa és autònoma. S'han de fer estudis del consum d'energia relatiu al processat, les comunicacions i als propis dispositius, i considerar un escenari a on les dades no es podran transmetre en temps real.

CAPÍTOL 6. REFERÈNCIES

[1] H. Karl and A. Willig, *Protocols and architectures for Wireless Sensor Networks*, John Wiley & Sons, Ltd, 2005.

[2] Wikipedia, «Wireless sensor network,» [En línia]. Available: http://en.wikipedia.org/wiki/Wireless_sensor_network.

[3] Xbow, [En línia]. Available: http://www.xbow.com/.

[4] Xbow, «MICA datasheet,» [En línia]. Available: http://stomach.v2.nl/docs/Hardware/DataSheets/Sensors/MICA_data_sheet.pdf.

[5] Xbow, «MICA2 datasheet,» [En línia]. Available: https://www.eol.ucar.edu/rtf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf.

[6] Xbow, «MICA2DOT datasheet,» [En línia]. Available: http://www.datasheetarchive.com/MICA2DOT/Datasheets-SW21/DSASW00417437.html.

[7] Y.-J. Wen, "Smart Dust Sensor Mote Characterization, Validation, Fusion and Actuation," [Online]. Available: http://best.berkeley.edu/research/smartLighting/support/yjMS_with_signature.pdf.

[8] Arduino, «Pàgina principal,» [En línia]. Available: http://arduino.cc/.

[9] Arduino, «Arduino Duemilanove,» [En línia]. Available: http://arduino.cc/en/Main/ArduinoBoardDuemilanove.

[10] Cooking Hacks, «Xbee Datasheet,» [En línia]. Available: http://www.cooking-hacks.com/skin/frontend/default/cooking/pdf/90000982_A.pdf.

[11] Digi International, «Xbee RF features,» [En línia]. Available: http://www.digi.com/pdf/chart_xbee_rf_features.pdf.

[12] Wikipedia, «Xbee,» [En línia]. Available: http://en.wikipedia.org/wiki/XBee.

[13] Digi International, «Pàgina principal,» [En línia]. Available: http://www.digi.com/.

[14] Digi International, «Xbee mesh modules,» [En línia]. Available: http://www.digi.com/pdf/ds_xbeemeshmodules.pdf.

[15] «Imatge càpsula electret,» [En línia]. Available: http://www.image.micros.com.pl/_icon_auto/rys.kpcm29b.jpg.

[16] Wikipedia, «Electret microphone schematic,» [En línia]. Available: http://upload.wikimedia.org/wikipedia/commons/thumb/5/57/Electret_condenser_microphone_schematic.png/220px-Electret_condenser_microphone_schematic.png.

[17] Wikipedia, «Electret microphone,» [En línia]. Available: http://en.wikipedia.org/wiki/Electret_microphone.

[18] ePanorama, «Powering microphones,» [En línia]. Available: http://www.epanorama.net/circuits/microphone_powering.html.

[19] Atmel, «ATmega 328 datasheet,» [En línia]. Available: http://www.atmel.com/Images/doc8161.pdf.

[20] [En línia]. Available: http://www.zen22142.zen.co.uk/Prac/ecm.html.

[21] R. Pallàs-Areny i J. G. Webster, *Analog signal processing*, John Wiley & Sons, Ltd, 1999.

[22] Fairchild, «LM358 datasheet,» [En línia]. Available: http://www.datasheetcatalog.org/datasheet/fairchild/LM358.pdf.

[23] Texas Instruments, «TL071 datasheet,» [En línia]. Available: http://www.datasheetcatalog.org/datasheet/texasinstruments/tl071.pdf.

[24] Microchip, «MCP6042 datasheet,» [En línia]. Available: http://www.datasheetcatalog.org/datasheet/microchip/21669b.pdf.

[25] Texas Instruments, «LM386 datasheet,» [En línia]. Available: http://www.ti.com/lit/ds/symlink/lm386.pdf.

[26] R. P. Areny, *Adquisición y distribución de señales*, Macombo, 1993.

[27] E. B. Albertí, *Procesado digital de señales. Fundamentos para comunicaciones y control*, Edicions UPC, 2006.

[28] Wikipedia, «Cooley - Tukey FFT algorithm,» [En línia]. Available: http://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm.

[29] Connexions, «Decimation in time,» [En línia]. Available: http://cnx.org/content/m12016/latest/.

[30] P. Zappi, «WSN Protocols,» [En línia]. Available: http://www-micrel.deis.unibo.it/MPHS/slidecorso0506/WSN_Protocol.pdf.

[31] Wikipedia, «IEEE 802.15.4,» [En línia]. Available: http://en.wikipedia.org/wiki/802.15.4.

[32] Wikipedia, «Zigbee,» [En línia]. Available: http://en.wikipedia.org/wiki/ZigBee.

[33] Wireless Sensor Network Research group, «802.15.4 vs Zigbee,» [En línia]. Available: http://www.sensor-networks.org/index.php?page=0823123150 .

[34] Digi International, «Xbee,» [En línia]. Available: http://www.digi.com/xbee/.

[35] IEEE, «IEEE 802.15.4,» 2003. [En línia]. Available: http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf.

[36] Wikipedia, «AODV,» [En línia]. Available: http://en.wikipedia.org/wiki/AODV.

[37] M. A. Perillo and W. B. Heinzelman, "Wireless Sensor Network protocols," [Online]. Available: http://www.ece.rochester.edu/courses/ECE586/readings/perillo.pdf.

[38] J. C. Castillo, T. Olivares i L. Orozco-Barbosa, «Routing protocols for wireless sensor networks-based network,» [En línia]. Available: http://www.dsi.uclm.es/descargas/thecnicalreports/DIAB-07-06-1/tehcnicalreport.pdf.

[39] Daintree networks, «Zigbee resource center,» [En línia]. Available: http://www.daintree.net/resources/index.php.

[40] Digi International, «Xbee 2.4 Digimesh module datasheet,» [En línia]. Available: ftp://ftp1.digi.com/support/documentation/90000991_D.pdf.

[41] Cooking Hacks, «Xbee Shield,» [En línia]. Available: http://www.cooking-hacks.com/index.php/documentation/tutorials/arduino-xbee-shield.

[42] Wikipedia, «A-Weighting,» [En línia]. Available: http://en.wikipedia.org/wiki/A-weighting.

[43] Wikipedia, «Butterfly diagram,» [En línia]. Available: http://en.wikipedia.org/wiki/Butterfly_diagram.

[44] «FFTW links,» [En línia]. Available: http://www.fftw.org/links.html.

[45] «FFT demystified,» [En línia]. Available: http://www.engineeringproductivitytools.com/stuff/T0001/.

[46] «Decimation in time FFT algorithms,» [En línia]. Available: http://www.systems.caltech.edu/EE/Courses/EE32b/handouts/FFT.pdf.

[47] «FFT,» [En línia]. Available: http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html.

[48] FFTW, «Pàgina principal,» [En línia]. Available: http://www.fftw.org/.

[49] The DSP Dimension, «DFT,» [En línia]. Available: http://www.dspdimension.com/admin/dft-a-pied/.

[50] FFTW, «Benchmarked FFT implementations,» [En línia]. Available: http://www.fftw.org/benchfft/ffts.html.

[51] P. Bourke, «FFT,» 1993. [En línia]. Available: http://paulbourke.net/miscellaneous/dft/.

[52] P. Bishop, «Audio spectrum analyzer,» [En línia]. Available: http://blurtime.blogspot.com.es/2010/11/arduino-realtime-audio-spectrum.html.

[53] Arduino, «FFT in 8 bits,» [En línia]. Available: http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1286718155.

[54] ELM, «FFT routine for Atmel,» 2005. [En línia]. Available: http://elm-chan.org/works/akilcd/report_e.html.

[55] [En línia]. Available: http://boolscott.wordpress.com/2010/02/04/arduino-processing-analogue-bar-graph-2/.

[56] «Xbee - Arduino documentation,» [En línia]. Available: http://xbee-arduino.googlecode.com/svn/trunk/docs/api/index.html.

[57] Ilektron, «Arduino puerto serie,» [En línia]. Available: http://www.ilektron.com/2011/04/arduino-el-puerto-serie-parte-2/.

[58] Arduino, «Aumentar frecuencia de muestreo,» [En línia]. Available: http://arduino.cc/forum/index.php/topic,68855.15.html.

[59] «Llibreries Xbee - Arduino,» [En línia]. Available: http://code.google.com/p/xbee-arduino/.

[60] M. Margolis, *Arduino Cookbook,* O'Reilly Media, Inc., 2011.

CAPÍTOL 7. BIBLIOGRAFIA CONSULTADA

http://arduino.cc/en/Reference/Setup
http://arduino.cc/en/Tutorial/Blink
http://arduino.cc/en/Tutorial/DigitalReadSerial
http://arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf
http://arduino.cc/en/Main/FAQ
http://arduino.cc/es/Main/ArduinoXbeeShield
http://sheepdogguides.com/arduino/FA1winktwo.htm
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1234443015
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1294526279
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1293143414.
http://arduino.cc/playground/Main/ManualsAndCurriculum
http://www.earthshineelectronics.com/files/ASKManualRev5.pdf
http://www.libelium.com/squidbee/index.php?title=Uploading_XBee_firmware
http://www.faludi.com/projects/common-xbee-mistakes/
http://forums.digi.com/support/forum/printthread_thread,8921
http://www.kobakant.at/DIY/?p=163
http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module#overview
http://forums.digi.com/support/forum/printthread_thread,8879
http://forums.digi.com/support/forum/index
http://www.libelium.com/squidbee/index.php?title=Hardware
http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module#more
http://www.digi.com/support/productdetail?pid=3352&osvid=57&type=utilities
http://www.libelium.com/squidbee/index.php?title=How_to_create_a_SquidBee%27s_network
http://www.libelium.com/squidbee/index.php?title=Sending_Data
http://www.blogelectronica.com/redes-zigbee-i-introduccion/
http://www.blogelectronica.com/zigbee-ii-mas-definiciones/
https://www.eol.ucar.edu/rtf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf
http://stomach.v2.nl/docs/Hardware/DataSheets/Sensors/MICA_data_sheet.pdf
http://www.xbow.jp/mts_mda_datasheet.pdf
http://www.todopic.com.ar/foros/index.php?topic=19699.0
http://users.otenet.gr/~athsam/index.htm
http://users.otenet.gr/~athsam/index.htm#Audio_Preamplifier
http://www.forosdeelectronica.com/f27/microfono-electret-943/
http://electronicayciencia.blogspot.com/2010/06/utilizar-un-microfono-electret.html
http://en.wikipedia.org/wiki/Miller_effect
http://electronicayciencia.blogspot.com/2010/05/preamplificador-microfono-electret.html
http://www.unicrom.com/topic.asp?TOPIC_ID=7099&FORUM_ID=15&CAT_ID=5&Forum_Title=Otros+en+circuitos+y+dise%F1os&Topic_Title=Circuito+para+microfono+electret
http://www.electronicafacil.net/circuitos/Polarizacion-microfono-electret.html
http://www.aes.org/technical/heyser/aes113.cfm
http://hyperphysics.phy-astr.gsu.edu/hbase/audio/mic2.html
http://www.pablin.com.ar/electron/circuito/audio/premic/index.htm
http://www.diyaudio.com/forums/everything-else/95193-matching-transistors-electret-microphone.html
http://www.neumann.com/forums/view.php?bn=neumann_micrec&key=1090257060&v=f

http://www.nerdkits.com/videos/
http://www.unicrom.com/Tut_DAC.asp
http://www.asifunciona.com/electronica/af_conv_ad/conv_ad_5.htm
http://www.dtic.upf.edu/~jlozano/interfaces/blow_sensor.html
http://www.reconnsworld.com/audio_electretamp.html
http://www.nerdkits.com/
http://www.abcdatos.com/tutoriales/electronicayelectricidad/electronica/circuitos/audio.html
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1208575180
http://arduino.cc/blog/category/sensors/electret-microphone/
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1202056662/15#20
http://arduino.cc/en/Tutorial/AnalogReadSerial
http://didier.longueville.free.fr/arduinoos/?p=1134.
http://blurtime.blogspot.com/2010/11/arduino-spectrum-with-video-out.
http://forum.sparkfun.com/viewtopic.php?t=13979
http://didier.longueville.free.fr/arduinoos
http://sheepdogguides.com/arduino/FA1main.
http://sheepdogguides.com/arduino/FA1adc.htm
http://arduino.cc/playground/Main/InterfacingWithHardware
http://robotgrrl.com/blog/2010/01/15/arduino-to-matlab-read-in-sensor-data/
http://didier.longueville.free.fr/arduinoos/?p=1057
http://arduino.cc/playground/Main/InterfacingWithHardware#audio_input
http://arduino.cc/en/Tutorial/Graph
http://www.arduino.cc/playground/Interfacing/Processing
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1259666136
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1244921933.
http://neuroelec.com/2011/03/fft-library-for-arduino
http://arduino.cc/en/Serial/Println
http://arduino.cc/en/Reference/Serial
http://arduino.cc/en/Reference/Sizeof
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1256725778
http://www.shuningbian.net/2009/01/tips-for-interfacing-electret-mic-with.
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1247113873
http://processing.org/learning/basics/
http://processing.org/learning/library/forwardfft.html
http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module#specs
http://hackaday.com/2008/11/02/wireless-arduino-programming-with-zigbee/
http://bildr.org/2011/04/arduino-xbee-wireless/
http://forum.sparkfun.com/viewtopic.php?p=68597
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1220541203
http://arduino.cc/playground/Shields/Xbee015
http://www.faludi.com/itp_coursework/meshnetworking/XBee/XBee_firmware_upgrade.html
https://sites.google.com/site/xbeetutorial/
http://www.cooking-hacks.com/forum/viewtopic.php?f=19&t=448&sid=13c79dcef78bfe5f92d081e5b7eed56a
http://en.wikipedia.org/wiki/Dynamic_Source_Routing
http://tools.ietf.org/html/rfc3561
http://antipastohw.blogspot.com.es/2009/01/xbee-shield-to-xbee-shield.html
http://www.arduino.cc/playground/Main/InterfacingWithHardware#audio_tutorial
http://www.ac-limoges.fr/sti_ge/IMG/pdf/X-CTU_Manual.pdf
http://arduino.cc/forum/index.php/topic,37751.0.html
http://www.libelium.com/squidbee/index.php?title=Main_Page

http://www.libelium.com/squidbee/index.php?title=Projects
http://arduino.cc/en/Main/ArduinoXbeeShield
http://www.homebrewtalk.com/f51/arduino-xbee-dual-stage-temp-controller-279657/
http://www.arduino.cc/en/Guide/ArduinoXbeeShield
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1293558696
http://code.google.com/p/xbee-arduino/source/browse/trunk/docs/api/class_frame_id_response.html
http://www.arduino.cc/en/uploads/Main/XbeeShieldSchematic.pdf
http://arduino.cc/en/Tutorial/DigitalPins
http://arduino.cc/en/Reference/PinMode
http://arduino.cc/en/Reference/Stream
http://letsmakerobots.com/node/12336
http://arduino.cc/forum/index.php?topic=63785.0
http://arduino.cc/en/Reference/RandomSeed
http://arduino.cc/en/Reference/Delay
http://msdn.microsoft.com/es-es/library/84787k22(v=vs.80).aspx#Y57
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1260726209
http://stackoverflow.com/questions/7557549/xbee-origin-address-in-arduino
http://www.tristantech.net/articles/xbee_tutorial/1.php
http://code.google.com/p/arduino-integer-fft/downloads/detail?name=fix_fft.cpp&can=2&q=
http://arduino.cc/en/Reference/ByteCast
http://upcommons.upc.edu/pfc/simple-search?query=arduino&sort_by=0&order=DESC&rpp=10&etal=0&start=20
http://www.ladyada.net/learn/arduino/lesson4.html
http://electronics.stackexchange.com/questions/54/saving-arduino-sensor-data-to-a-text-file
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1220897178
http://www.sengpielaudio.com/calculator-transferfactor.htm
http://www.diracdelta.co.uk/science/source/a/w/aweighting/source.html
http://www.ffte.jp/
http://code.google.com/p/arduinoscope/downloads/list
http://code.google.com/p/arduinoscope/downloads/detail?name=arduino-arduinoscope.pde&can=2&q=

Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANNEXOS

TÍTOL DEL PFC: Disseny d'una xarxa de sensors acústics i ambientals pel projecte SIAC

TITULACIÓ: Enginyeria de Telecomunicació (segon cicle)

AUTOR: Josep Cardona Fernández

DIRECTOR: Adán Amor Garriga Torres

SUPERVISOR: Rafael Vidal Ferré

DATA: 18 d'abril de 2012

ÍNDEX

I. PRESSUPOST WSN SIAC

Element	**Preu unitari**
Arduino Duemilenanove	21.89 €
Xbee Shield (amb antenna)	15 €
Mòdul Xbee Digimesh	14.24 €
LM386	0.83 €
Altres components electrònics	2 €
Total	53.96 €

II. CODIS

- Mostreig + funció FFT

```
// FFT 128 punts
#include <fix_fft.h>
#include <stdint.h>

float dataR[128];
float im[128];
int i,val;

void setup()
{
  Serial.begin(9600);
  //Prescaler
  //ADPS2 - ADPS1 - ADPS0 - Division Factor
  //0       - 0     - 0     ->2
  //0       - 0     - 1     ->2
  //0       - 1     - 0     ->4
  //0       - 1     - 1     ->8
  //1       - 0     - 0     ->16
  //1       - 0     - 1     ->32
  //1       - 1     - 0     ->64
  //1       - 1     - 1     ->128
  //Configure to Prescaler=32
  bitWrite(ADCSRA,ADPS2,1);
  bitWrite(ADCSRA,ADPS1,0);
  bitWrite(ADCSRA,ADPS0,1);
}

void loop()
{
  for (i=0; i < 128; i++){
      dataR[i] = analogRead(A2); //llegim dades pel pin 2
      im[i]=0; l'àudio és una senyal sense part imaginaria
    }

   FFT(1,7,dataR,im); //FFT
  //Aw(dataR);//A weighting: multipliquem el resultat pel filtre
  //FFT(-1,7,dataR,im); iFFT

  Serial.println("inicio");
  for (i=1; i< 64; i++)
  {
    dataR[i]=sqrt(dataR[i]*dataR[i]+im[i]*im[i]); //valor absolut
    Serial.println(dataR[i]);

  } Serial.println("fin");

    delay(10);
}
```

- Rutina per verificar que el procés de la FFT és reversible

```
//rutina fft ifft – codi per verificar la rutina de la FFT, comprovant que la sortida = entrada

#include <fix_fft.h>

float dataIn[128], dataOut[128], im[128];
int val, i;

void setup()
{
  Serial.begin(9600);
}

void loop ()
{
  for (i=0; i<128; i++)
  {
    val = analogRead(A2);
    dataIn[i]=val;
    im[i]=0;
    dataOut[i]=val;//guardem les dades original, per després comparar-les
  }

  FFT(1,7,dataIn,im); //FFT
  FFT(-1,7,dataIn,im); //iFFT

  for(int i=0; i<128; i++)
  {
    Serial.print("Dades IN:");
    Serial.print(dataOut[i]);
    Serial.print(" --- Dades OUT:");
    Serial.println(dataIn[i]);
  }
}
```

- Programa per tractar amb paquets Xbee

```
#include <XBee.h>

XBee xbee = XBee();
XBeeResponse response = XBeeResponse();
// creem objectes per guardar la resposta que rebem
ZBRxResponse rx = ZBRxResponse();
ModemStatusResponse msr = ModemStatusResponse();

XBeeAddress64 a_AB2 = XBeeAddress64();

int iter = 0; //per controlar que només demani ACKs el primer cop, no cada vegada que envia dades perquè ralentitza el sistema.
int control=0; //per controlar que hi hagi conexió
int adrs[7]; //guardam l'adreça
int payload[6];
String inString[] = {0x13,0xA2,0x00,0x40,0x4B,0x91,0x2F}; //adreça node XB2
String inString2[] = {0x13,0xA2,0x00,0x40,0x4B,0x91,0x25}; //XB3

void setup() {
  // init serial
  xbee.begin(9600);

  // assignem valors inicials als vectors
  for(int i =0; i<7; i++)
  {
    adrs[i]=0;
  }
  for(int i =0; i<6; i++)
  {
    payload[i]=0;
  }
}

// llegim continuament esperant rebre paquets

void loop() {

    xbee.readPacket();

//************************************************ codi que hauria de ser executat un cop, ************************

    if (xbee.getResponse().isAvailable() && iter==0)
    {
        // rebem dades
        if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
        {
          //tenim un paquet, "omplim" l'objecte amb les dades
          xbee.getResponse().getZBRxResponse(rx);


          if (rx.getOption() == ZB_PACKET_ACKNOWLEDGED) {
              // el Tx ha rebut l'ACK
              Serial.println("ACK rebut pel Tx");
          } else {
              // el Tx no ha rebut ACK
              Serial.println("El Tx no ha rebut un ACK");
          }
        }
```

```
        iter=1;
        control=1;
      }


     if(control==0)
      {
        Serial.println("NO HI HA CONTACTE");
      }
//*************************************************************************************************************
 ********************************
      if(control!=0) { //començam a tractar les dades
        Serial.println("preparat!");


        Serial.print("checksum es ");
        Serial.println(rx.getChecksum(), DEC);

        Serial.print("longitud del paquet: ");
        Serial.println(rx.getPacketLength(), DEC);



        for (int i = 0; i < rx.getDataLength(); i++) //payload
        {
         Serial.print("payload [");
         Serial.print(i, DEC);
         Serial.print("] is ");
         Serial.print(rx.getData()[i], HEX); //getData() = payload
         //  payload[i] = int(rx.getData()[i]);
        }

        for (int i = 0; i <xbee.getResponse().getFrameDataLength(); i++) // frame: 64b adreça +
receive options + dades
        {
           adrs[i]=int(xbee.getResponse().getFrameData()[i+1]); //guardem l'adreça de 64 bits a
adrs[i], tipus int
          Serial.print("address frame [");
          Serial.print(i, DEC);
          Serial.print("] is ");
          Serial.println(adrs[i],DEC); //char
          Serial.println(xbee.getResponse().getFrameData()[i+1],DEC);

        }

        // comparem les adreces i indentifiquem nodes
        for (int i = 0; i <7; i++)
        {
          if((adrs[i] == inString[i].toInt()) && (adrs[6] == inString[6].toInt()))
          {
            Serial.println("AB2");
          }
          if(adrs[i] == inString2[i].toInt() && (adrs[6] == inString2[6].toInt()))
          {
            Serial.println("AB3");
          }
        }
      }
}
```

- Llibreries que inclouen la rutina de la FFT i el filtre AW
  - fft.h

```
#ifndef FIXFFT_H
#define FIXFFT_H

#include <WProgram.h>

/*
        inclou declaracions per FFT i Aw
*/

short FFT(short int dir,long m,float *x,float *y);
short Aw(float *dataIn);

#endif
```

  - fft.cpp

```
#include "fix_fft.h"
#include <WProgram.h>


short FFT(short int dir,long m, float *x, float *y)
{
  long n,i,i1,j,k,i2,l,l1,l2;
  double c1,c2,tx,ty,t1,t2,u1,u2,z;
  //float arg,wr,wi;

  /* Calculate the number of points */
  n = 1;
  for (i=0;i<m;i++)
    n *= 2;

  /* Do the bit reversal */
  i2 = n >> 1;
  j = 0;
  for (i=0;i<n-1;i++) {
    if (i < j) {
      tx = x[i];
      ty = y[i];
      x[i] = x[j];
      y[i] = y[j];
      x[j] = tx;
      y[j] = ty;
    }
    k = i2;
    while (k <= j) {
      j -= k;
      k >>= 1;
    }
    j += k;
  }
```

```
  /* Compute the FFT */
  c1 = -1.0;
  c2 = 0.0;
  l2 = 1;
  for (l=0;l<m;l++) {
    l1 = l2;
    l2 <<= 1;
    u1 = 1.0;
    u2 = 0.0;

    for (j=0;j<l1;j++) {
      for (i=j;i<n;i+=l2) {
        i1 = i + l1;
        t1 = u1 * x[i1] - u2 * y[i1];
        t2 = u1 * y[i1] + u2 * x[i1];
        x[i1] = x[i] - t1;
        y[i1] = y[i] - t2;
        x[i] += t1;
        y[i] += t2;
      }
      z =  u1 * c1 - u2 * c2;
      u2 = u1 * c2 + u2 * c1;
      u1 = z;
    }
    c2 = sqrt((1.0 - c1) / 2.0);
    if (dir == 1)
      c2 = -c2;
    c1 = sqrt((1.0 + c1) / 2.0);
  }

  /* Scaling for forward transform */
  if (dir == 1) {
    for (i=0;i<n;i++) {
      x[i] /= n;
      y[i] /= n;
    }
  }

  return(1);
}
```

```
/* Implementem el filtre a-weighting */

short Aw(float *dataIn)
{
   int i;
   float filter[128] = {0.352595793, 0.618729528, 0.763275873 ,0.84004871, 0.881505342,
0.903822463, 0.915011451, 0.91923674, 0.918800048,0.915054953,0.908846413,
0.900732776,0.891103437,0.880243969,0.868373575,0.855667344,0.842269868,
0.828303777,0.813875224,0.799077473,0.783993322,0.768696758,0.753254141,
0.73772507,0.72216305,0.706616025,0.691126826,0.675733556,0.660469946,
0.645365673,
0.630446668,
0.615735402,
0.601251157,
0.587010292,
0.57302649,
0.559310993,
0.54587283,
0.532719026,
0.519854801,
0.507283759,
0.495008062,
0.483028585,
0.471345075,
0.459956279,
0.448860074,
0.438053584,
0.42753328,
0.417295078,
0.407334423,
0.39764637,
0.388225647,
0.379066722,
0.370163859,
0.361511163,
0.353102626,
0.344932165,
0.336993656,
0.329280964,
0.321787965,
0.314508575,
0.307436761,
0.300566562,
0.293892105,
0.287407608,
};

   for(i = 0; i<64; i++)
   {
          dataIn[i] = dataIn[i]*filter[i];
   }

  return(1);
}
```

- Codi original per configurar el ADC del Arduino.

```
#define mysize 128

int mydata[mysize];

void setup()
{
  Serial.begin(9600);

  //Prescaler
  //ADPS2 - ADPS1 - ADPS0 - Division Factor
  //0       - 0     - 0     ->2
  //0       - 0     - 1     ->2
  //0       - 1     - 0     ->4
  //0       - 1     - 1     ->8
  //1       - 0     - 0     ->16
  //1       - 0     - 1     ->32
  //1       - 1     - 0     ->64
  //1       - 1     - 1     ->128
  //Configure to Prescaler=32
  bitWrite(ADCSRA,ADPS2,1);
  bitWrite(ADCSRA,ADPS1,0);
  bitWrite(ADCSRA,ADPS0,1);

  //Entrada A2

ADMUX=(1<<ADLAR)|(0<<REFS1)|(1<<REFS0)|(0<<MUX3)|(0<<MUX2)|(1<<MUX1)|(0<<
MUX0);
}


void loop()
{

  for (int i=0; i<mysize;i++)
  {
     mydata[i]=analogReadFast();
  }


  for (int i=0; i<mysize;i++)
  {
    Serial.println(mydata[i]);
  }

}

int analogReadFast()
{
          ADCSRA|=(1<<ADSC);
          // ADSC is cleared when the conversion finishes
          while (bit_is_set(ADCSRA, ADSC));
       return ADCH;
}
```

- Codi FFT adaptat al processador ATmega 328. No sabem si IFFT OK.

```
/*
This Example acquire analog signal from A0 of Arduino, and Serial out to Processing
application to visualize.

Analog signal is captured at 9.6 KHz, 64 spectrum bands each 150Hz which can be change
from adcInit()
Load the this file to Arduio, run Processing application.

Original Fixed point FFT library is from ELM Chan, http://elm-
chan.org/works/akilcd/report_e.html
Ported to the library and demo codes are from AMurchick
http://arduino.cc/forum/index.php/topic,37751.0.html
Processing code is from boolscott http://boolscott.wordpress.com/2010/02/04/arduino-
processing-analogue-bar-graph-2/
*/

#include <stdint.h>
#include <ffft.h>

#define  IR_AUDIO  2 // ADC channel to capture

volatile  byte  position = 0;
volatile  long  zero = 0;

int16_t capture[FFT_N];                 /* Wave captureing buffer */
complex_t bfly_buff[FFT_N];             /* FFT buffer */
uint16_t spektrum[FFT_N/2];             /* Spectrum output buffer */

void setup()
{
  Serial.begin(9600);
  adcInit();
  adcCalb();
  establishContact();  // send a byte to establish contact until Processing respon
}

void loop()
{
  if (position == FFT_N)
  {
    fft_input(capture, bfly_buff);
    fft_execute(bfly_buff);
    fft_output(bfly_buff, spektrum);

    for (byte i = 0; i < 64; i++){
      Serial.print(spektrum[i],BYTE);
    }
   position = 0;
  }
}
void establishContact() {
 while (Serial.available() <= 0) {
      Serial.print('A', BYTE);   // send a capital A
      delay(300);
  }
}
```

```
// free running ADC fills capture buffer
ISR(ADC_vect)
{
  if (position >= FFT_N)
    return;

  capture[position] = ADC + zero;
  if (capture[position] == -1 || capture[position] == 1)
    capture[position] = 0;

  position++;
}
void adcInit(){
  /*  REFS0 : VCC use as a ref, IR_AUDIO : channel selection, ADEN : ADC Enable, ADSC
: ADC Start, ADATE : ADC Auto Trigger Enable, ADIE : ADC Interrupt Enable,  ADPS :
ADC Prescaler  */
  // free running ADC mode, f = ( 16MHz / prescaler ) / 13 cycles per conversion
  ADMUX = _BV(REFS0) | IR_AUDIO; // | _BV(ADLAR);
//  ADCSRA = _BV(ADSC) | _BV(ADEN) | _BV(ADATE) | _BV(ADIE) | _BV(ADPS2) |
_BV(ADPS1) //prescaler 64 : 19231 Hz - 300Hz per 64 divisions
  ADCSRA = _BV(ADSC) | _BV(ADEN) | _BV(ADATE) | _BV(ADIE) | _BV(ADPS2) |
_BV(ADPS1) | _BV(ADPS0); // prescaler 128 : 9615 Hz - 150 Hz per 64 divisions, better for
most music
  sei();
}
void adcCalb(){
  Serial.println("Start to calc zero");
  long midl = 0;
  // get 2 meashurment at 2 sec
  // on ADC input must be NO SIGNAL!!!
  for (byte i = 0; i < 2; i++)
  {
    position = 0;
    delay(100);
    midl += capture[0];
    delay(900);
  }
  zero = -midl/2;
  Serial.println("Done.");
}
```

- Codi per visualitzar l'anterior en Processing.

```
// Feel Free to edit these variables ////////////////////////
String  xLabel = "Frequency";
String  yLabel = "Values";
String  Heading = "Arduino FFT";
String  URL = "01/02/2010";
float Vcc = 255.0;    // the measured voltage of your usb
int NumOfVertDivisions=5;      // dark gray
int NumOfVertSubDivisions=10;  // light gray


int NumOfBars=64;    // you can choose the number of bars, but it can cause issues
                     // since you should change what the arduino sends


// if these are changed, backgroung image has problems
// a plain background solves the problem
int ScreenWidth = 800, ScreenHeight=600;
//////////////////////////////////////////////////////

//  Serial port stuff ////////////////////
import processing.serial.*;
Serial myPort;
boolean firstContact = false;
int[] serialInArray = new int[NumOfBars];
int serialCount = 0;
/////////////////////////////////////////

int LeftMargin=100;
int RightMArgin=80;
int TextGap=50;
int GraphYposition=80;
float BarPercent = 0.4;

int value;

PFont font;
PImage bg;

int temp;
float yRatio = 0.58;
int BarGap, BarWidth, DivisounsWidth;
int[] bars = new int[NumOfBars];

void setup(){

  bg = loadImage("BG.jpg");

  /// NB SETTINGS ////////////////////////////////////////////////////
  myPort = new Serial(this, Serial.list()[1], 9600);
  ////////////////////////////////////////////////////////////////////

  DivisounsWidth = (ScreenWidth-LeftMargin-RightMArgin)/(NumOfBars);
  BarWidth = int(BarPercent*float(DivisounsWidth));
  BarGap = DivisounsWidth - BarWidth;

  size(ScreenWidth,ScreenHeight);
  font = createFont("Arial",12);
```

```
  textAlign(CENTER);
  textFont(font);
}

void draw(){

//  background(bg);     // My one used a background image, I've
  background(250);      // commented it out and put a plain colour

  //  Headings();          // Displays bar width, Bar gap or any variable.
  Axis();
  Labels();
  PrintBars();
//  Line();
//  Dots();
}

// Send Recieve data //
void serialEvent(Serial myPort) {
println("1");
  // read a byte from the serial port:
  int inByte = myPort.read();

  if (firstContact == false) {
    if (inByte == 'A') {
      myPort.clear();          // clear the serial port buffer
      firstContact = true;     // you've had first contact from the microcontroller
      myPort.write('A');       // ask for more
    }
  }
  else {
    println("2");
    // Add the latest byte from the serial port to array:
    serialInArray[serialCount] = inByte;
    serialCount++;

    // If we have 6 bytes:
    if (serialCount > NumOfBars -1 ) {
      println("3");

      for (int x=0;x<NumOfBars;x++){
        bars[x] = int (yRatio*(ScreenHeight)*(serialInArray[x]/256.0));
      }

      // Send a capital A to request new sensor readings:
      //myPort.write('A');
      // Reset serialCount:
      serialCount = 0;
    }
  }
}
```

```
/////// Display any variables for testing here//////////////
void Headings(){
  fill(0 );
  text("BarWidth",50,TextGap );
  text("BarGap",250,TextGap );
  text("DivisounsWidth",450,TextGap );
  text(BarWidth,100,TextGap );
  text(BarGap,300,TextGap );
  text(DivisounsWidth,520,TextGap );
}


void PrintBars(){

  int c=0;
  for (int i=0;i<NumOfBars;i++){

    fill((0xe4+c),(255-bars[i]+c),(0x1a+c));
    stroke(90);
    rect(i*DivisounsWidth+LeftMargin,  ScreenHeight-GraphYposition,  BarWidth,  -bars[i]);
    fill(0x2e,0x2a,0x2a);
//    text(float(bars[i])/(yRatio*(ScreenHeight))*Vcc,
i*DivisounsWidth+LeftMargin+BarWidth/2,  ScreenHeight-bars[i]-5-GraphYposition );
//    text("A",  i*DivisounsWidth+LeftMargin+BarWidth/2 -5,  ScreenHeight-
GraphYposition+20 );
    text(i,  i*DivisounsWidth+LeftMargin+BarWidth/2 +5,  ScreenHeight-GraphYposition+20
);
  }
}

void Axis(){

  strokeWeight(1);
  stroke(220);
  for(float x=0;x<=NumOfVertSubDivisions;x++){

    int bars=(ScreenHeight-GraphYposition)-
int(yRatio*(ScreenHeight)*(x/NumOfVertSubDivisions));
    line(LeftMargin-15,bars,ScreenWidth-RightMArgin-DivisounsWidth+50,bars);
  }
  strokeWeight(1);
  stroke(180);
  for(float x=0;x<=NumOfVertDivisions;x++){

    int bars=(ScreenHeight-GraphYposition)-
int(yRatio*(ScreenHeight)*(x/NumOfVertDivisions));
    line(LeftMargin-15,bars,ScreenWidth-RightMArgin-DivisounsWidth+50,bars);
  }
  strokeWeight(2);
  stroke(90);
  line(LeftMargin-15, ScreenHeight-GraphYposition+2, ScreenWidth-RightMArgin-
DivisounsWidth+50, ScreenHeight-GraphYposition+2);
  line(LeftMargin-15,ScreenHeight-GraphYposition+2,LeftMargin-15,GraphYposition+80);
  strokeWeight(1);
}
```

```
void Labels(){
  textFont(font,18);
  fill(50);
  rotate(radians(-90));
  text(yLabel,-ScreenHeight/2,LeftMargin-45);
  textFont(font,10);
  for(float x=0;x<=NumOfVertDivisions;x++){

    int bars=(ScreenHeight-GraphYposition)-
int(yRatio*(ScreenHeight)*(x/NumOfVertDivisions));
    text(round(x),-bars,LeftMargin-20);
  }

  textFont(font,18);
  rotate(radians(90));
  text(xLabel,LeftMargin+(ScreenWidth-LeftMargin-RightMArgin-50)/2,ScreenHeight-
GraphYposition+40);
  textFont(font,24);
  fill(50);
  text(Heading,LeftMargin+(ScreenWidth-LeftMargin-RightMArgin-50)/2,70);
  textFont(font);

  fill(150);
  text(URL,ScreenWidth-RightMArgin-40,ScreenHeight-15);
  textFont(font);

}
```