EPS
Escola Politècnica Superior
UdG

# Projecte/Treball Fi de Carrera

**Estudi:** Eng. Tècn. Informàtica de Sistemes. Pla 2001

**Títol:** Integració d'un sensor sonar d'escombrat lateral en un robot submarí

**Document:** Memòria

**Alumne:** Joel Vidal Verdaguer

**Director/Tutor:** Marc Carreres/Narcís Palomeras
**Departament:** Arquitectura i Tecnologia de Computadors
**Àrea:** ATC

**Convocatòria** (mes/any): 9/11

# Índex

# 1 Introducció

## 1.1 Antecedents

El grup de Visió per Computador i Robòtica (VICOROB) disposa de varis robots submarins per a la recerca i inspecció subaquàtica. Recentment s'ha adquirit un sensor sonar d'escombrat lateral el qual s'utilitza per realitzar imatges acústiques del fons marí quan aquest es mou principalment a velocitat constant i mantenint el rumb.

## 1.2 Motivació i abast

Els robots del grup VICOROB estan equipats amb diferents tipus de sensors i càmeres per analitzar el fons marí. Aquests sensors són de gran qualitat i permeten conèixer de manera bastant satisfactòria l'entorn a les proximitats del robot. Freqüentment però, aquest sensors estant sotmesos a diferents restriccions depenent de la seva naturalesa de funcionament, de tal manera que es necessària la seva combinació per resoldre determinats problemes en diferents situacions.

En el camp de l'anàlisi del fons marí, actualment, el principal focus de recerca està centrat en la utilització de càmeres de vídeo, amb tècniques com la generació de mosaics del fons. En aquest context, existeixen algunes tasques per les quals els sistemes actuals no estant preparats i per les quals es requereix un gran esforç o es fa difícil el seu plantejament. En general, aquestes tasques estant centrades en la cerca o l'observació de quelcom en grans extensions del fons marí i a vegades en condicions de visibilitat reduïdes, tasques que a dia d'avui són de difícil realització i requereixen un gran esforç amb els sensors actuals.

Amb aquest projecte, es pretén integrar un nou sistema de captura d'imatges sonores del fons marí, en un dels robots. Amb la integració d'aquest nou sensor s'espera obtenir una opció alternativa als sistemes actuals que pugui aportar informació addicional sobre el fons. Aquest sistema podrà ser utilitzat per realitzar tasques per les quals els altres sensors no estant preparats o bé per complementar informació d'altres sensor.
.

## 1.3 Objectius

Els objectius d'aquest projecte són els següents:

- Estudi del sensor sonar: principis teòrics, modes de funcionament i comunicació.
- Desenvolupament del software per accedir al sensor i per integrar-lo en la arquitectura software del robot.
- Desenvolupament del software per representar les imatges capturades directament pel sensor.
- Comprovació del funcionament del sensor: realització de proves bàsiques per a comprovar el correcte funcionament i l'acompliment de les especificacions.
- Reconstrucció de la imatge tenint en compte la informació de navegació del robot.

## 1.4 Planificació

A continuació es llisten per ordre cronològic les fases per les qual ha passat la realització del projecte.

1. ***Presa de contacte amb el sensor***
   El projecte s'inicia amb l'estudi de la documentació que acompanya el sensor, el primer contacte amb l'aparell, un assemblatge temporal i un cop en funcionament, la realització d'algunes proves amb el software comercial.

   Durant aquesta primera fase també s'estableix contacte amb l'empresa Imagenex per aclarir alguns dubtes sorgits respecte el sensor i la seva instal·lació.

2. ***Desenvolupament d'una aplicació en C++ per llegir les dades dels sensors.***
   Un cop estudiat el funcionament del sensor, es procedeix a realitzar un aplicació en C++ que permeti capturar dades del sensor, guardar-les en un fitxer i poder visualitzar-les amb el programa comercial. Part del disseny i el codi d'aquesta aplicació seran reutilitzats a l'hora d'integrar el sensor a l'arquitectura de control del robot.

3. ***Introducció a l'arquitectura de control COLA2.***
   La següent fase consisteix en estudiar l'arquitectura COLA2 utilitzada per controlar el robot Girona500, on es vol integrar el sensor. Durant aquesta fase es llegeix la documentació existent sobre l'arquitectura, s'estudien implementacions d'altres sensor, i s'aclareixen alguns dubtes respecte el funcionament del sistema o la integració del sensor.

4. ***Disseny i desenvolupament dels components necessaris per integrar el sensor a l'arquitectura de control COLA2.***
   Una vegada es coneix el funcionament de l'arquitectura de control COLA2, es procedeix a dissenyar i desenvolupar els components necessaris per integrar el sensor. En aquest punt és necessari desenvolupar a més a més un component d'entrada/sortida TCP/IP ja que no existia a l'arquitectura.

5. ***Disseny i desenvolupament d'una interfície gràfica per visualitzar les dades obtingudes pel sensor en temps real.***
   En la fase anterior, les dades obtingues pel sensor eren guardades en un fitxer per ser visualitzades offline. Així doncs, es dissenya i desenvolupa una interfície gràfica que permetrà visualitzar les dades obtingudes en temps real.

6. ***Disseny i desenvolupament d'una aplicació per visualitzar les imatges sonores obtingudes pel sensor de manera offline.***
   En aquesta fase es dissenya i desenvolupa una aplicació que ens permetrà tornar a visualitzar les imatges sonores del sensor de manera offline una vegada s'hagi acabat de maniobrar.

7. ***Disseny i desenvolupament d'una aplicació per reconstruir la imatge acústica segons la navegació del robot.***
   Una vegada desenvolupada una aplicació per visualitzar les imatges sonores offline, es dissenya i desenvolupa una altra aplicació per reconstruir la imatge acústica obtinguda pel robot segons la navegació.

8. ***Realització d'una prova final per provar la reconstrucció d'una imatge acústica segons la navegació del robot.***
   En aquesta fase es realitza un ultima prova que consisteix en tirar un objecte a la piscina, fer-hi passar per sobre el Girona 500 amb el sonar activat i reconstruir la imatge resultant segons la navegació del robot.

9. ***Elaboració de la documentació.***
   Finalment, s'elabora la documentació necessària per mostrar el procés de realització del projecte.

## 1.5 Estructura de la memòria

L'estructura del present document és la següent. En els punts 1, 2, 3 i 4 es realitza la presentació del projecte i s'expliquen i estudien les eines necessàries per dur-lo a terme. Els punts 5 i 6 es centren en l'explicació del procés d'integració del sonar en el robot. El dos següents apartats, el 7 i 8, expliquen el desenvolupament d'aplicacions per observar les dades capturades pel robot de manera offline. Finalment, en el punt 9 es tracten les conclusions del projecte.

# 2 El robot submarí Girona 500

El Girona 500 és un lleuger i compacte vehicle autònom submarí que entre les seves principals característiques té la possibilitat de ser reconfigurat per dur a terme tasques molt diverses. Les seves capacitats van des del reconeixement del fons marí fins a tasques d'inspecció i intervenció.

Pel que fa a les seves característiques mecàniques, cal remarcar, que tal com indica el seu nom, aquest vehicle està preparat per treballar a profunditats de fins a 500 metres. Les seves dimensions són 1 metre d'alt, per 1 metre d'ample per 1,5 de llarg i compte amb 3 cascs en forma de torpede, els quals contenen la amplia majoria de l'equipament, de manera que estan disposats com un triangle, amb dos cascos a la part superior i un a la part inferior (vegeu la figura 2.1). Cada un dels cascs fa 0.3 metres de diàmetre. El pes del robot no supera els 200 Kg i té suficients blocs de flotació com per considerar que te una flotabilitat quasi neutra.

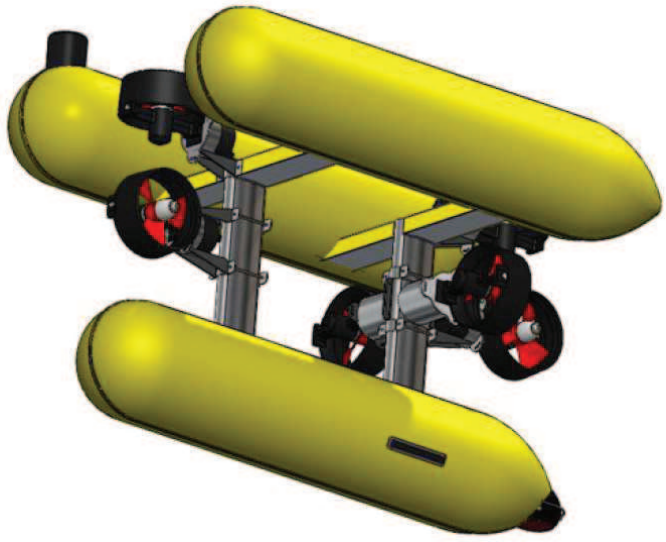


*Figura 2.1: Representació en 3D del robot submarí Girona 500.*

El sistema de propulsió admet una gran varietat de possibilitats, des de l'ús de únicament 3 motors, disposant solament de 3 graus de llibertat, fins a l'ús de 8 motors aconseguint així, 6 graus de llibertat.

El sistema d'alimentació utilitza una bateria d'uns 2,2 kW · h d'energia.
Durant el procés de disseny del vehicle les bateries es van concebre per aguantar unes 8 hores de funcionament, però la seva durada real depèn de les diferents possibilitats de configuració.

L'ordinador principal del vehicle és un PC-104 amb un processador Intel Atom 1.6GHz el qual executa una distribució de GNU/Linux. Per altra banda, el sistema també té un segon ordinador PC-104 amb un processador Core2Duo 1.2GHz el qual s'utilitza per realitzar tasques d'un alt requeriment computacional. Existeix una xarxa de 8 ports de 100Mbps la qual estableix una xarxa entre els ordinadors i aquells sensors que requereixin transferir una gran quantitat de dades. El sistema també compta amb una antena wifi (vegeu la figura 2.2).

Pel que fa a l'arquitectura de control, actualment s'utilitza la COLA2, la qual també és explicada en el següent capítol.

Finalment, referent els sensors, cal destacar que a més dels sensors específics els quals formen part de la càrrega útil de cada configuració, el sistema compta amb un conjunt de sensors permanents que es poden utilitzar en totes les missions. Aquests sensors permeten conèixer entre d'altres paràmetres, la velocitat, l'angle o la posició absoluta del robot. També hi ha altres sensors, com un sonar per percebre possibles obstacles al voltant del vehicle o una càmera de vídeo CCD amb color acompanyada amb un parell de llums LED de 40W.

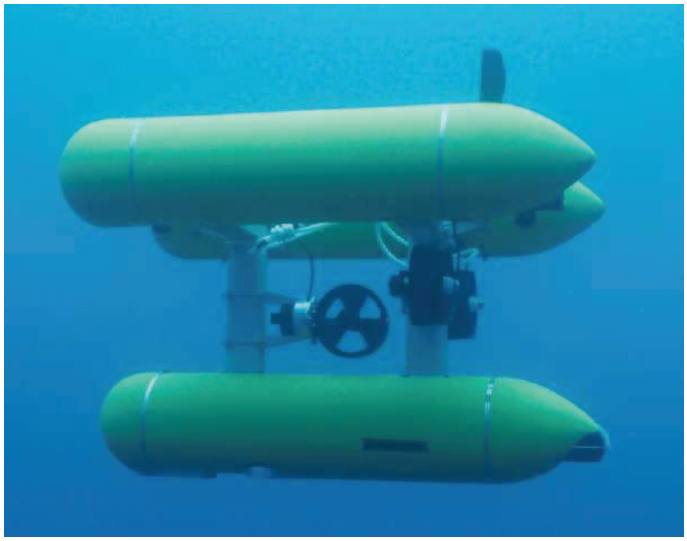


*Figura 2.2: El robot submarí Girona 500 a les instal·lacions del Centre d'Investigació de Robòtica Submarina (CIRS).*

# 3 L’arquitectura de control COLA2

## 3.1 Introducció

L’arquitectura de control COLA2 (Component Oriented Layer-Based Architecture for Autonomy) és una solució software que es va concebre com una base on construir software per robots. Aquesta base consta d’un conjunt de components individuals capaços d'interactuar fàcilment entre ells i amb el hardware del robot però que estan netament desacoblats els uns dels altres. D’aquesta manera s’evita que el programador dels components del robot s’hagi de preocupar de tasques repetitives i tedioses.

Actualment és l’arquitectura utilitzada per tots els robots del laboratori de robòtica submarina, incloent el robot Girona 500, i per tant és l’arquitectura en la qual s’ha d’integrar el sensor.

Entenent que el sonar d’escombrat lateral serà un component més de l’arquitectura, serà necessari conèixer el funcionament general des del punt de vista d’un component i tot el que fa referència a la gestió d’aquests, manipulació de dades i el sistema d’E/S.

## 3.2 Model de l’arquitectura

La arquitectura COLA2 segueix un model basat en capes, que organitza els components en tres capes diferents: la capa de missió, la capa d’execució i la capa reactiva. En la figura 3.1 podem observar una exemple d’aquest model.

**Capa Reactiva**

La *capa Reactiva* és una capa molt dependent als sensor i actuadors utilitzats, no obstant es divideix en tres mòduls per reduir la seva dependència:

- *Vehicle interface*: Aquest mòdul conté els components, anomenats drivers, que interactuen directament amb el hardware, llegint dades dels sensor o enviant comandes als actuadors. Els drivers converteixen totes les dades en unitats coherents i les fa referents al *fixed body frame* del vehicle.

- *Percepció*: Aquest mòdul rep dades reunides pel mòdul *vehicle interface*. Està format per diversos components anomenats *processing units*. El navegador, detector d'obstacles i detector d'objectius.

- Guia i control: Aquest mòdul inclou un conjunt de comportaments, el coordinador i el controlador de velocitat. Els comportaments són funcionalitats bàsiques del robot que poden anar des d'un procés per controlar al nivell de la bateria fins a un procés que segueixi trajectòries. En general, tots els comportaments busquen complir un objectiu. Els comportaments reben dades tant del *vehicle interface* com de la *percepció*, d'aquesta manera és independent del sensors físics i actuadors. El coordinador combina les respostes generades pels comportaments habilitats i el control de velocitat converteix aquestes dades a un vector de força per cada motor.

Encara que el COLA2 és una arquitectura basada en capes, la capa reactiva pot ser vista com una arquitectura basada en comportaments, en la qual hi ha un conjunt de comportaments habilitats que són coordinats per arribar a un objectiu. No obstant, qui decideix quin comportament està habilitat és la capa executiva seguint les instruccions de la capa de missió.

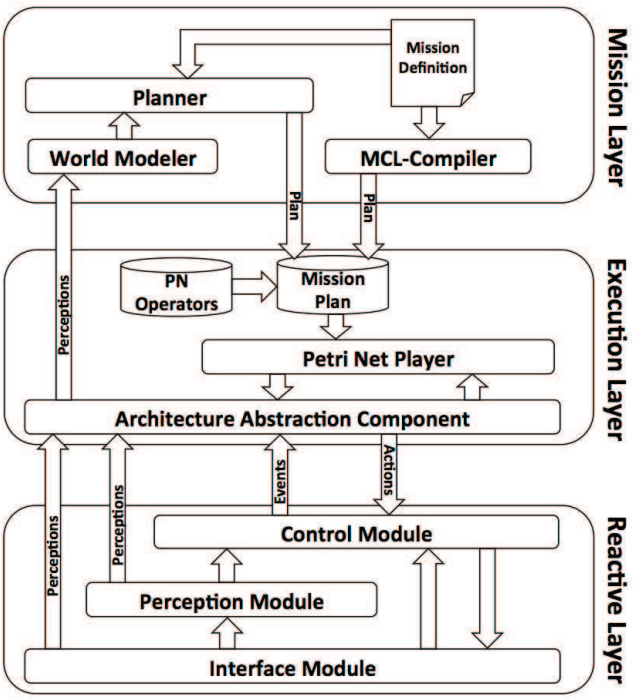



*Figura 3.1 Exemple d'una arquitectura de control basada en 3 capes.*

## Capa executiva

La capa executiva actua com una interfície entre la *capa reactiva* i *la capa de missió*. Aquesta capa tradueix els plans d'alt nivell a comandes de baix nivell habilitant i deshabilitant comportaments de la capa reactiva. Està

formada per dos components, *Architecture Abstraction Component* i *Petri Net Player*.

El ***AAC*** ens ofereix una interfície a la capa reactiva a traves de tres senyals:

- Accions: Activen o desactiven comportaments basics.
- Esdeveniments: Enviats per la capa reactiva, notifiquen canvis en els estats dels seus comportaments.
- Percepcions: Són valors de sensors específics o de *processing units* que es transmeten des de la capa reactiva a la de missió, per extreure informació rellevant sobre l'estat del mon actual quan s'està utilitzant un planificador.

El ***PNP*** executa la missió planejada, donada per *la capa de missió*. La missió és definida per mitjà de *xarxes de Petri* que descriuen quina acció ha de ser executada depenent dels esdeveniments rebuts. Bàsicament, actua com *un Discrete Event System* (DES) connectant plans discrets amb comportaments continus.

**Capa Missió**

Els plans predefinits són el sistemes utilitzats actualment en missions d'AUV. No obstant els plans offline poden fallar durant l'execució. Per altra banda, l'ús de plans generats online poden arribar a un comportament impredictible del vehicle. Per tant, val la pena trobar un compromís entre els plans offline i els plans online. COLA2 introdueix un llenguatge d'alt nivell, anomenat *Mission Control Language*(MCL), per descriure les missions offline que són automàticament compilades a una *xarxa de Petri*. Aquest llenguatge pot incloure un operador de planificació. D'aquesta manera es pot planificar una missió capaç de seleccionar quin és l'objectiu més apropiat per completar la missió.

## 3.3 Disseny de l'arquitectura

L'arquitectura està dividida en 5 mòduls de manera que hi ha un mòdul nucli i quatre mòduls addicionals que proporcionen al nucli un conjunt de funcions concretes. Podem observar un esquema d'aquest disseny a la figura 3.2.

Els mòduls són els següents:

- *El mòdul Core:* És el nucli del sistema, i s'encarrega de les funcions principals com la gestió de components, comunicació entre components, gestió de configuracions i enregistraments de logs, entre altres.

- *Networking:* Permet a les estacions comunicar-se entre elles, i al sistema amb altres sistemes externs a traves de la xarxa.

- *Data Manipulation:* Aquest mòdul permet convertir dades entre el format XML, format en que les dades s'intercanvien dins el sistema, i un format de dades de tipus aritmètic o lògic.

- *Threading:* Proporciona les classes per crear nous fils d'execució, periòdics o no periòdics, i poder executar tasques de manera simultània.

- *I/O:* Proporciona un accés uniforme i fàcil als dispositius d'entrada/sortida.

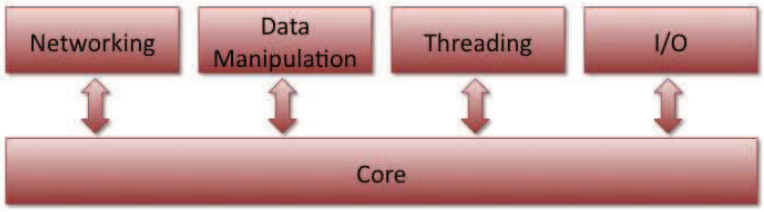



Figura 3.2: Esquema del disseny modular emparat per l'arquitectura COLA2.

Podem observar el diagrama de classes de l'arquitectura en la figura 3.3.

Tal com ja s'ha esmentat, degut el fet que l'arquitectura COLA2 està concebuda per treballar em components individuals desacoblats entre ells, no es necessari la comprensió de tot el sistema per integrar el nostre sensor. Així doncs, únicament dedicarem especial atenció a aquells punts de l'arquitectura més relacionats amb el nostre projecte.

Dins el ***mòdul Core*** cal destacar els següents punts.

Primerament, la interfície *IComponents* representa un component del sistema robòtic. Les seves operacions són les següents:

- *getName*: Obté el nom del component.
- *isEnable*: Comprova si el component està habilitat.
- *setEanble*: Habilitar o deshabilitar el component.
- initialize: Operació que es cridada pel sistema i inicialitza el component.
- getConfiguration: Obté la configuració del component.
- setConfiguration: Estableix la configuració del component.
- *handleServiceRequest*: Realitza una petició de servei i n'obté el resultat.
- *update*: Permet publicar dades que el component rep com a subscriptor.

És important destacar que la classe base *CComponent*, la qual és heretada per tots els components, dona accés a la funció d'enregistrament de logs, a traves de l'objecte *logger,* instancia de la classe *CLogger*. A més a més, també compte amb la operació *publish,* que ens permetrà publicar les dades del component.

La classe *CComponentFactory* és l'encarregada de crear aquests components a partir dels noms i paràmetres especificats al fitxer de configuració del sistema.

La classe *CStation* s'encarrega d'aglutinar tots els components presents en un mateix node de la xarxa. Aquesta classe també dona accés als dispositius d'entrada/sortida del node on es troba.

Observem que hi ha 2 tipus d'estacions*; CMasterStation* i *CSlaveStation*. Aquest fet es deu a la necessitat de tenir la informació centrelitzada en una sola estació. La estació Master és instanciada una única vegada i és la que gestiona el fitxer de configuració del sistema i el de tots els components. Les estacions Slaves han de demanar aquesta informació a la estació Master ha traves de la xarxa.

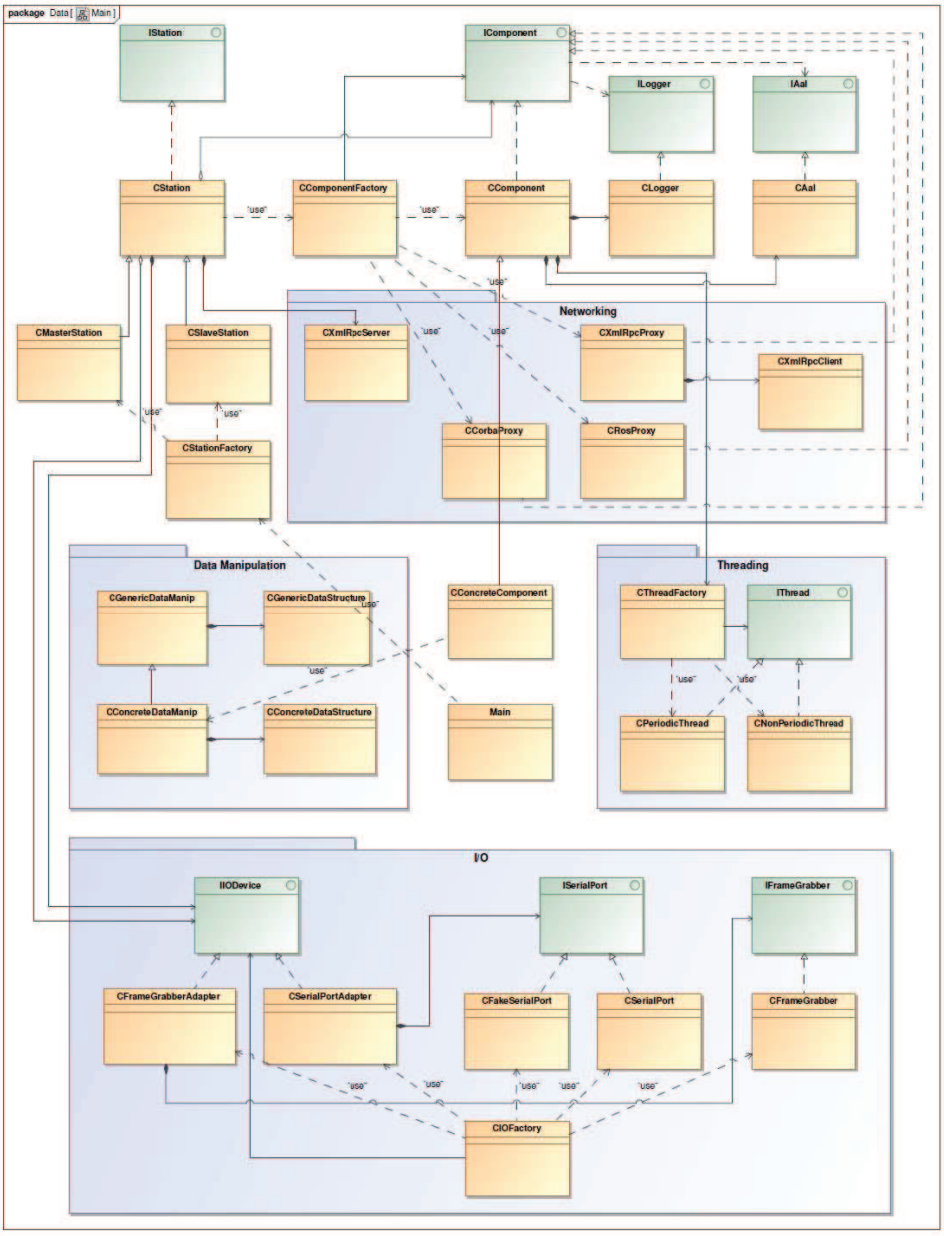



*Figura 3.3: Diagrama de classes de l'arquitectura COLA2*

Dins el ***mòdul Data Manipulation,*** em de destacar la classe *CXmlDataManip* la qual és l'encarregada de parsejar, comprovar la estructura i carregar a memòria els documents XML.

La classe *CGenericDataManip* fa *servir CXmlDataManip* per convertir el text XML en valors aritmètics i lògics o realitzar la operació inversa.

Cada tupla de dades, en el document XML, és un topic identificat per un nom i necessita un subclasse *CGenericDataManip* per poder-la traduir des de i cap a XML. Les operacions de la classe *CGenericDataManip* són les següents:

- *getName*: Obté el nom del topic em el qual s'està operant.
- *loadXmlString*: Carrega a memòria un document XML amb una estructura determinada.
- *data*: Proprociona accés de lectura i d'escriptura a la tupla
- *getXmlString*: Obté la representació actual de la tupla en XML.
- *stringifyNames*: Retorna una cadena de caràcters amb els noms dels camps de la tupla.
- *stringifyValues*: Retorna una cadena de caràcters amb els valors dels camps de la tupla.

Dins el ***mòdul Threading,*** observem la classe *CThreadFactory* la qual ens permetrà crear un fil d'execució periòdic, utilitzant la classe *CPeriodicThread*, o no periòdic, utilitzant *CNonPeriodicThread*, mitjançant el mètode *createThread*. Ambdues classes implementen la interfície *IThread,* la qual mitjançant l'operació *start* executa un funció passada.

Dins el ***mòdul I/O*** ens interessa fixar-nos en la interfície *IIODevice,* ja que és la que descriu qualsevol interfície d'entrada i sortida. En aquest punt és important conèixer que tots els dispositius són descrits com un *data stream*.

Les seves operacions són les següents:

- *read*: Llegeix el nombre de bytes especificats des de l'stream.
- *readByte*: Llegeix un sol byte des de l'stream.
- *readLine*: Llegeix una cadena de caràcters des de l'stream.
- *readUntile*: Llegeix de l'stream fins que troba un byte especificat.
- *write*: Escriu el nombre de bytes especificats a l'stream.

## 3.4 Estructura dels fitxers de configuració

Per poder integrar completament el sensor a l'arquitectura COLA2 és necessari, a més de conèixer el seu disseny, conèixer l'estructura dels seus fitxers de configuració.

Podem distingir dos fitxers basics de configuració; el fitxer de sistema i el fitxer de components.

Fitxer de sistema:

El fitxer de sistema té dues estructures possibles depenent si és troba en una estació Master o una estació Slave. Únicament es farà incís en el fitxer de l'estació Master ja que, com s'ha comentat, l'estació Slave obté les dades de configuració del sistema i els components a través de l'estació Master, per tant aquest fitxer conte únicament informació sobre el seu nom i la localització de l'estació Master.

El node arrel d'aquet fitxer conté l'atribut role, on s'indica si es tracta d'un fitxer de l'estació Master o Slave.

El fitxer de l'estació Master és el que especifica totes les configuracions de totes les estacions. En aquest fitxer podem distingir les seccions següents:

- *logger*: A la secció logger hi ha configuració referent a l'enregistrament de logs.

- *stations*: Descriu quines estacions intervenen en el sistema. En la descripció de les estacions, serà necessari especificar els dispositius d'entrada/sortida i també els components que aglutina l'estació.

- *subscriptions*: Indica quins components estant subscrits a quins altres i quins topics.

La figura 3.4 mostra un exemple d'un fitxer de sistema.

```
<station_cfg role="master">

<logger>
    <single_line_comment_chars>% </single_line_comment_chars>
        <system_pattern>%Y-%m-%d %H:%M:%S.%i:%p: %t</system_pattern>
        <mission_pattern>%Y %m %d %H %M %S %i           %t</
           information_pattern>
        <enable_console_logger>true</enable_console_logger>
        <enable_file_logger>true</enable_file_logger>
</logger>


<stations>

<station name="aquesta_estacio">
        <io>
                <xmlotcp_port>12000</xmlotcp_port>
                <devices>
                        <device type="serial_port" name="
                           port_dels_motors" path="/dev/ttyACM0"
                           baud_rate="19200" char_size="8" stop_bits="
                           1" parity="NONE" flow_control="NONE" />
                        <device type="framegrabber" name="camera" path
                           ="/dev/video0" width="640" height="480"
                           depth="8" />
```

```
                </devices>
        </io>

        <components>
                <component name="productor_de_dades" enabled="true"
                   via="xmlotcp" />
        </components>
</station>

<station name="estacio_esclava">
        <io>
                <xmlotcp_port>12000</xmlotcp_port>
                <devices/>
        </io>

        <components>
                <component name="consumidor_de_dades" enabled="true"
                   via="xmlotcp" />
        </components>
</station>

</stations>


<subscriptions>
        <subscription publisher="productor_de_dades" subscriber="
           consumidor_de_dades" topic="dades" />
</subscriptions>

</station_cfg>
```

*Figura 3.4: Exemple d'un fitxer de sistema per una estació master.*

Fitxer de components:

Aquest fitxer esta format per un node arrel *config*, del qual poden penjar tants topics com és vulgui, on cada topic és un component. Dins de cada topic d'un component hi ha tots els paràmetres de configuració del component concret, amb el seu nom, tipus i valor. La figura 3.5 mostra un exemple d'aquest fitxer.

```
<config>
  <component_a_configuration>

    <timeout_port_serie type="real" value="5000" />
    <nom_dispositu_a_usar type="string" value="port_serie_tal" />
    <calcular_x type="boolean" value="true" />

  </component_a_configuration>

  <component_b_configuration>

    <offset type="real" value="1.6" />
    <nom_dispositiu_camera type="string" value="camera0" />

  </component_b_configuration>
</config>
```

*Figura 3.5: Exemple d'una configuració típica de dos components.*

# 4 Sonar d'escombrat lateral Imagenex

## 4.1 Introducció

Un sonar d'escombrat lateral és un sistema sonar utilitzat per crear eficientment grans imatges sonores del fons marí.

Entre els seus principals usos cal destacar la cerca i reconeixement d'objectes superficials o semi-enterrats, la detecció de perills per a la navegació o els estudis de transport de sediments, sent àmpliament utilitzat en qualsevol procediment en el qual sigui necessari observar amplies zones del fons marí.

Generalment, aquest sensors adopten la forma d'un torpede el qual és arrastrat per una embarcació mitjançant un cable, de manera que és possible visualitzar, a través d'un monitor, les zones del fons marí creuades per l'embarcació. També es pot donar el cas que el torpede sigui arrastrat per un submarí o fins i tot que el sensor estigui integrat en el casc de l'embarcació o del submarí, sent aquest últim cas, el sistema emparat per el nostre sensor.

*Figura 4.1: Torpede Imagenex model 782 "Yellowfin", cable i ordinador.*

Durant el transcurs d'aquest projecte s'utilitzarà el sensor Imagenex model 782, però a diferencia de la versió "comercial" anomenada "Yellowfin" (vegeu la f*igura 4.1*), aquest està compost únicament per un conjunt de 2 transductor i una placa de control, sense el torpede, de manera que pugui ser integrat en el casc del submarí.

## 4.2 Principis teòrics

El seu principi de funcionament esta basat en l'ús de la propagació de so sota l'aigua em la finalitat de detectar la descripció geofísica del fons marí. El sistema "insonifica" una zona del fons i n'obté una imatge sonora a partir de la reflexió d'aquest so.

El sensor és bàsicament un sonar actiu, el qual hi ha un emissor de so i un receptor. Inicialment, l'emissor emet un pols de so anomenat "ping". Quan el pols impacte amb un objecte, es reflecteix i el receptor capta la reflexió d'aquesta mateixa ona, anomenada "echo". La figura 4.2 mostra un esquema d'un sonar actiu.

És possible, conèixer la distancia a la que es troba l'objecte de l'emissor a partir de la diferencia de temps que hi ha entre que l'ona és emesa i es detecta la seva reflexió, si és coneix la velocitat del so a l'aigua, així com també és possible, conèixer propietats de l'objecte a partir de la intensitat de l'ona reflectida.

Aquest pols de so, generalment, és emes per un generador de senyal, un amplificador de potencia y un transductor o matriu electroacústica.

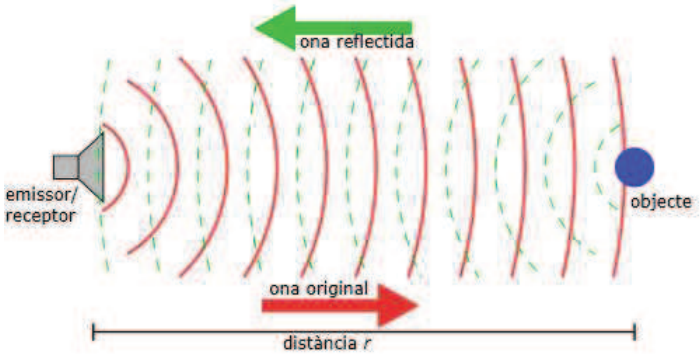



*Figura 4.2: Esquema bàsic del principi de funcionament d'un sonar actiu.*

En un sonar d'escombrat lateral aquest proses és realitza emeten els plusos de so paral·lelament a la trajectòria del sensor a traves de l'aigua, de manera que aquests s'emeten en direcció al fons marí. En aquest sistemes, per conveni, es solen utilitzar dos transductors col·locats de manera que apunten en direccions oposades, en una inclinació de 20º de caiguda respecte l'horitzontal, tal com es mostra a la figura 4.3.

Aquesta distribució ens permet obtenir una imatge lineal sonora del fons marí situat immediatament sota els transductors, encara que inevitablement, sempre hi haurà una zona sota del torpede que no serà insonificada.

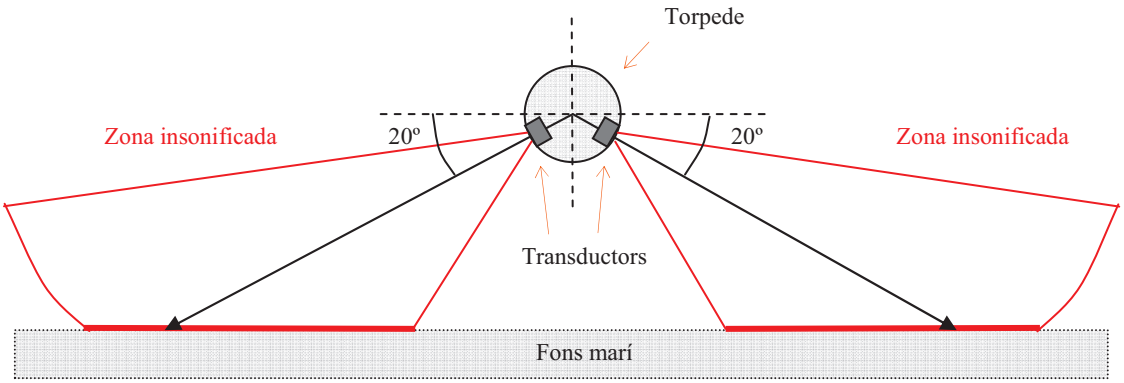



*Figura 4.3: Esquema basic del funcionament d'un sonar d'escombrat lateral*

Així doncs, utilitzant aquest sistema, es possible generar una imatge sonora del fons marí desplaçant els sensors de manera paral·lela el fons i agrupant en una sola imatge totes les imatges lineals obtingudes durant el desplaçament (vegeu la figura 4.4).

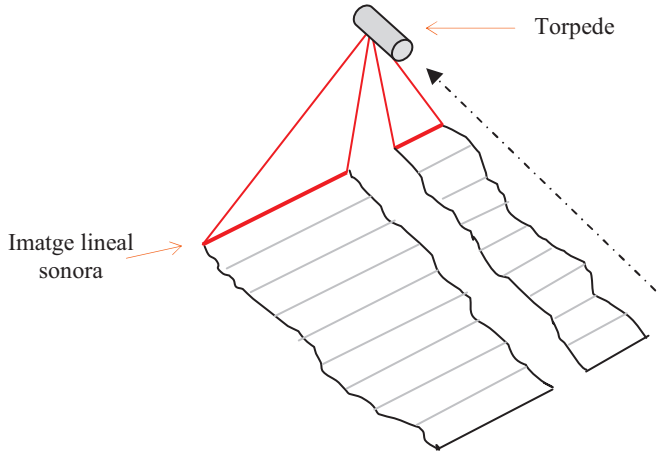



*Figura 4.4: Esquema de la generació d'una imatge sonora del fons marí a partir d'un conjunt d'imatges lineals*

La amplada de la imatge obtinguda, dependrà de l'amplada del feix i del rang de treball (vegeu la figura 4.5).

L'amplada del feix, normalment expressada en graus, dependrà directament de la freqüència utilitzada per generar el pols de so. Una freqüència més alta generarà una feix més estret i una freqüència més baixa un feix més ampla, aquestes característiques es refereixen tant a l'amplada com a la gruixudària del feix.

Per altra banda, el rang, normalment expressat en metres, està relacionat amb el temps màxim que s'inverteixi en esperar l'ona reflectida, de manera que un major temps d'espera significarà la detecció d'objectes més llunyans. Així i tot,

la freqüència també té una incidència important sobre el rang, sent les baixes freqüències les més adequades per a rangs grans.

També és important el guany inicial del pols emes, que haurà de ser més alt o més baix depenent del rang, les característiques del fons i si es dona el cas, dels objectes cercats.

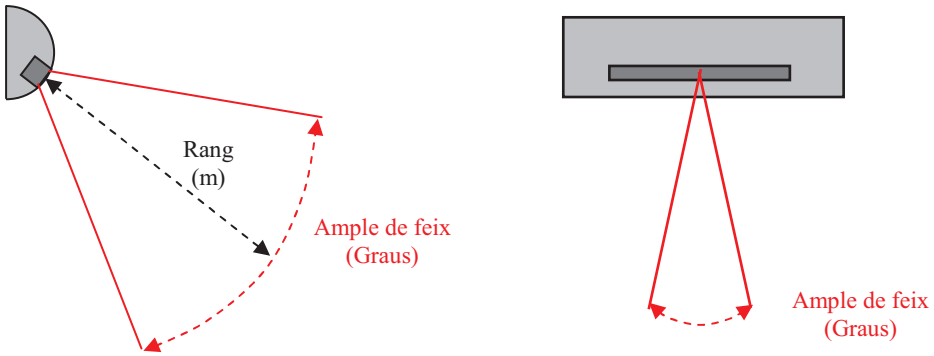



*Figura 4.5: Esquema de les característiques "ample de feix" i "rang" d'un pols de so amés per un transductor*

## 4.3 Interpretació de les imatges sonores

Un tema fonamental en la majoria de sonars i en especial en el sonar d'escombrat lateral és el correcte anàlisis de les imatges sonores obtingudes pel sensor. Es necessita una gran experiència en la interpretació d'aquest tipus d'imatges per distingir correctament la majoria dels elements que hi poden aparèixer.

En aquest projecte simplement veurem els conceptes bàsica de la seva interpretació, necessaris per identificar els principals elements d'una imatge.

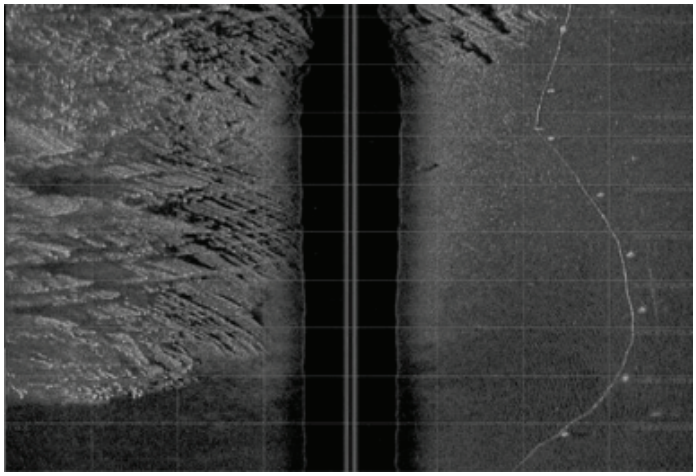


*Figura 4.6: Exemple d'un imatge sonora capturada mitjançant un sonar d'escombrat lateral*

En un primer moment l'aspecte d'algunes imatges (vegeu la figura 4.6) ens podria fer pensar que són imatges normals, com les obtingudes mitjançant una càmera, però simplement nomes ho pareixen, ja que el que realment estem veien és una imatge del so reflectit en el fons marí. Així doncs, primerament hem de observar quines seran les característiques bàsiques de les imatges obtingudes mitjançant aquest procés.

Inicialment em de diferenciar els valors que presenten els eixos de coordenades d'aquest tipus d'imatges respecte a les imatges convencionals.

En primer lloc, la coordenada horitzontal d'una imatge no es correspon amb la mateixa coordenada horitzontal del fons, sinó que fa referència a la distancia que es troba una determinada reflexió del punt d'emissió. El centre de la imatge és el valor zero i a mesura que ens desplacem cap a algun dels dos costats, augmenta la distancia del punt concret respecte el torpede. Podem entendre millor aquest fet observant la figura 4.7.

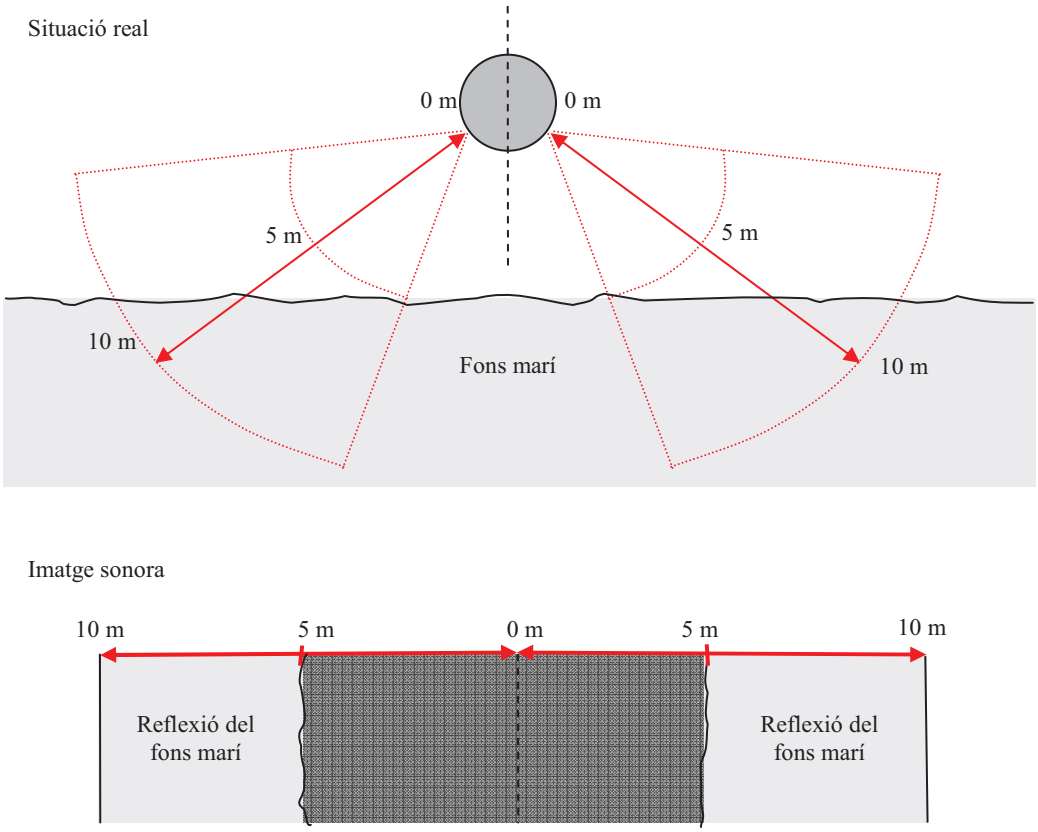



*Figura 4.7: Esquema per entendre la coordenada horitzontal en una imatge sonora.*

Es interessant observar que en les imatges sonores sempre tindrem una zona central en la qual no és detectarà cap reflexió degut a que és la distancia que hi ha entre el torpede i la primera reflexió del fons marí. No s'ha de confondre aquest espai sense reflexió amb la zona que no és insonificada de sota el torpede.

Pel que fa a la coordenada horitzontal, dependrà del tipus de imatge que analitzem. Algunes imatges, típicament les obtingudes en temps real, la coordenada horitzontal és temps, de manera que les distancies dependran de la velocitat del torpede. En altres imatges, típicament aquelles que han estat tractades, la coordenada horitzontal és en metres de manera molt aproximada a la realitat.

Un altre característica d'aquest tipus de imatge és que solen estar representades per diferents nivells d'intensitat d'un sol color, normalment gris. En aquest cas, cada nivell d'intensitat del color representa un nivell d'intensitat de l'ona reflectida, de manera que, depenent de la textura de cada objecte o del nombre d'objectes situats a la mateixa distancia, variarà el nivell d'intensitat de cada punt.

Un altre aspecte important d'aquest tipus d'imatges són les anomenades ombres acústiques. Aquestes ombres són produïdes per zones en les quals degut a al presencia d'un objecte no arriba el pols de so, de manera que no és produeix cap reflexió. Això produeix, que en una determinada distancia del sensor no s'observi cap retorn (vegeu la figura 4.9).

Aquest fet, que tant es pot produir per un forat com per un objecte alt, ens permet aproximar l'altura dels objectes.

Per aproximar l'altura d'un objecte a partir de la seva ombra és necessari conèixer l'altura del torpede, la distancia de l'ombra del torpede i la allargada de l'ombra. Usualment, s'utilitza la distancia del torpede al terra per obtenir una aproximació de l'altura d'aquest. Es pot observar un esquema d'aquest procediment a la figura 4.8.

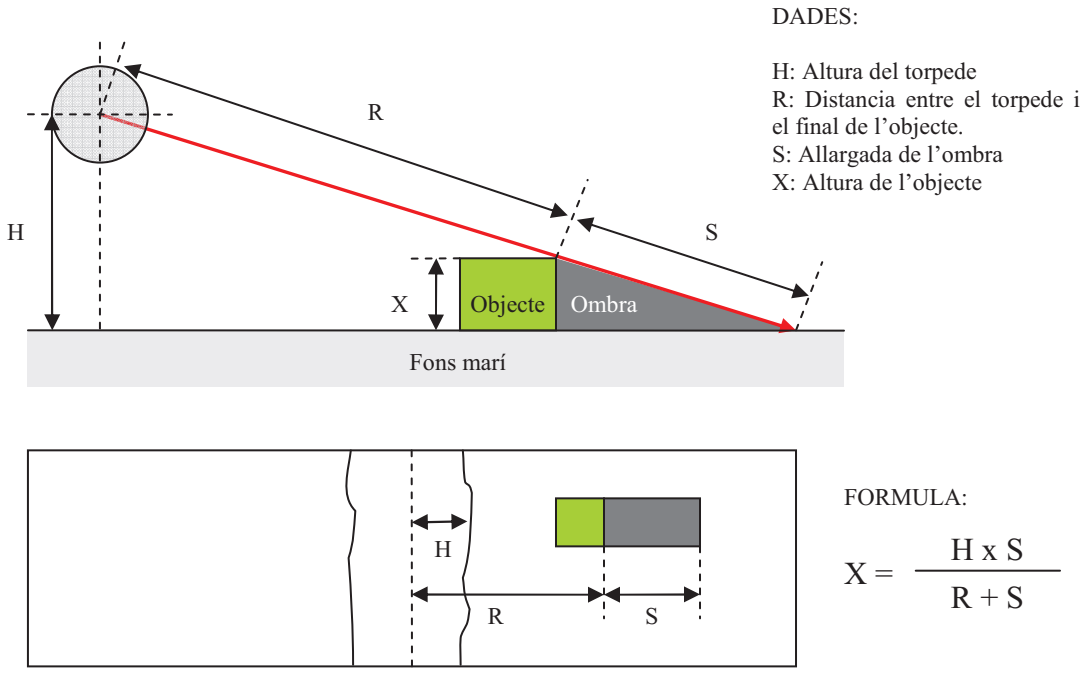



*Figura 4.8: Esquema del sistema d'aproximació de l'altura d'un objecte a partir de la seva ombra.*

Utilitzant el que s'ha explicat en aquest apartat i altres mètodes més avançats és possible realitzar una interpretació bastant fidel de la realitat, a partir de la imatge obtinguda per un sensor.

*Figura 4.9: Imatge sonora d'un vaixell enfonsat realitzada amb un sonar d'escombrat lateral*

## 4.4 Característiques

Aquestes són les principals característiques especificades pels sonars d'escombrat lateral i en concret, els valors que compleix el sonar d'escombrat lateral Imagenex model 872:

**Freqüència de funcionament**.
És refereix a la freqüència utilitzada per generar els polsos de so.
Aquest sonar compte amb tres valors possibles: 260kHz, 330kHz, 770kHz.

**Ampla del feix.**
Fa referència a l'amplada i la gruixudària dels pols de so.
Aquest valor depèn de la freqüència i pot tenir els següents valors en el nostre model, tal i com mostra a la figura 4.10:

- 260kHz :2.2º x 75º
- 330kHz: 1.8º x 60º
- 770kHz: 0.7º x 30º

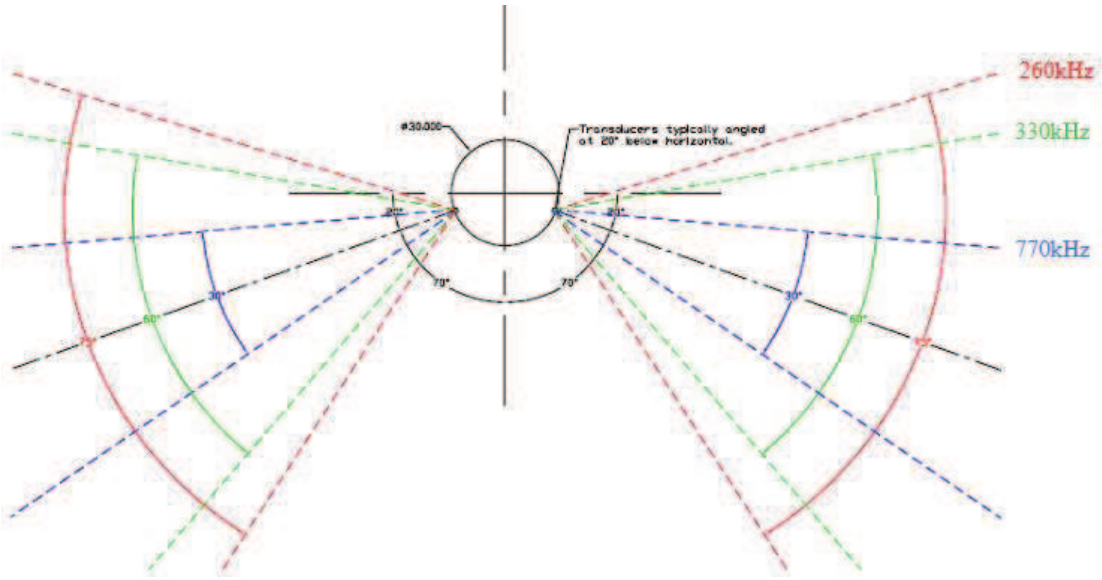



*Figura 4.10: Esquema de l'amplada del feix del model 872 d'Imagenex.*

**Rang màxim.**
Fa referència a la distancia màxima de detecció d'ones reflectides a la qual el sonar pot treballar.
Aquest valor és 200m en el nostra model.

**Resolució.**
Fa referència a la distancia real mínima entre dos punts de cada imatge lineal obtinguda pel sensor. En el nostre sonar aquest valor depèn del rang i compleix la següent formula: Rang / 1000.

**Font d'alimentació.**
Fa referència a la font d'alimentació que necessita el sonar per funcionar correctament. En el nostra sonar aquest valor és 24 VDC i menys de 2.5 Watts.

**Interfície de comunicació**.
Fa referència a la interfície de comunicació utilitzada pel sonar.
El nostra sonar utilitza la interfície Ethernet. També és necessari conèixer que el sonar té assignada la IP 192.168.0.7 i el port 4040 des de la seva fabricació.

## 4.5 Funcionament i protocols

En aquest apartat ens centrarem a estudiar el funcionament i el protocol utilitzat pel sonar d'escombrat lateral Imagenex model 872.

El sistema és controla a traves d'una interfície Ethernet, utilitzant el protocol TCP sobre el model TCP/IP. La comunicació segueix una arquitectura client-servidor en la qual el sonar actua com a servidor, responent totes les peticions del client, tal i com es mostra a la figura 4.11. L'ordinador que es vol connectar

el sensor haurà d'actuar com a client, realitzant totes les peticions que necessiti el servidor.

El sonar té una IP i un port estàtic ja que estant assignats des de la seva fabricació.

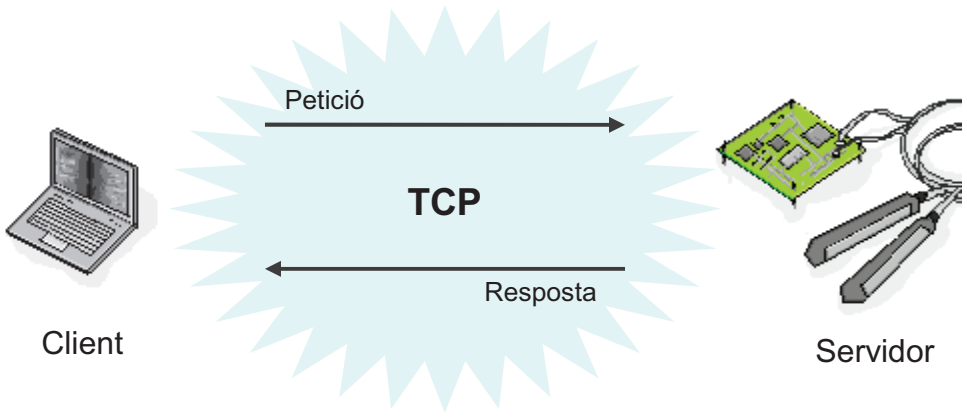



Figura 4.11: Esquema de petició i resposta d'una arquitectura client-servidor

En aquest protocol, bàsicament, es poden diferenciar dos tipus de paquets TCP diferents; els paquets de petició i els paquet de resposta, els quals es van intercalant durant tota la comunicació seguint una estructura de pregunta i resposta.

És important tenir present, que el protocol i els formats dels paquets estant preparats per una amplia varietat de productes amb diferents característiques i que no entra dins els objectius d'aquest projecte, analitzar tot el sistema de comunicació, sinó només els utilitzats pel model 782. Així dons, hi ha una seria de valors en els paquets, etiquetats com a reservats, que sempre tindran un valor fix, o en alguns casos no s'utilitzaran.

El paquet de petició:

Aquest paquet amb el nom "Comanda de canvi de dades (Switch Data Command)" és el format de paquet TCP utilitzat per fer una petició el sonar em l'objectiu de rebre les dades de l'eco.

Les dades de resposta són enviades en dos paquets diferents, les dades de babord i les dades d'estribord, de manera que és necessari fer dos peticions per obtenir un ping sencer. La primera, la qual és una petició per fer un ping i rebre el primer paquet de dades, i la segona, la qual nomes és una petició per rebre el segon paquet de dades. Aquestes dos peticions es diferencien a traves d'un numero que conte el paquet.

En aquest paquet és necessari passar-li cada vegada totes les dades de configuració per realitzar el ping, a més d'un numero per diferenciar de quin tipus de petició es tracta. La figura 4.12 mostra un taula amb el seu format.

| Bytes | Descripció | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 - 7 | **0xFE** | **0x44** | Reservat | Rang | Reservat | Reservat | Reservat | Freq. |
| 8 - 15 | Guany | Reservat | Balanç del guany | Reservat | Reservat | Reservat | Reservat | Reservat |
| 16 - 23 | Reservat | Reservat | Num. de paquet | Reservat | Reservat | Reservat | Reservat | Reservat |
| 24 - 26 | Reservat | Reservat | Final **0xFD** | | | | | |

*Figura 4.12: Taula de format del paquet de petició Switch Data Command*

El paquet té una llargada de 27 Bytes i el seu format és el següent:

*Byte 0 :* Capçalera (Switch data header).
**0xFE**

*Byte 1:* Capçalera (Switch data header).
**0x44**

*Byte 2:* Reservat.
**0**

*Byte 3:* El rang.
Aquest valor esta limitat segons la freqüència de treball configurada. Els valors vàlids segons la freqüència són els següents:
260kHz : 10, 20, 30, 40, 50, 60, 80, 100, 125, 150, 200m
330kHz: 10, 20, 30, 40, 50, 60, 80, 100, 125, 150, 200m
800kHz: 10, 20, 30, 40, 50m

*Byte 4 – 6:* Reservats.
**0**

*Byte 7:* La freqüència.
Els valors vàlids, de 1 a 3, són els següents:
Un 1 per 260kHz.
Un 2 per 330kHz.
Un 3 per 800kHz.

*Byte 8:* El guany.
Ha de ser un valor entre 0dB i 40dB.

*Byte 9:* Reservats.
**0**

*Byte 10:* Balanç del guany.
Ha de ser un valor entre 0 i 60 de manera que és possible establir una diferencia de +/-3dB entre els canals amb diferencies de 0.1dB

*Byte 11 – 17:*Reservats.
**0**

*Byte 18:* Numero de paquet.
Aquest numero és utilitzat per diferenciar el tipus de petició. Ha de ser un numero de 0 a 7, però únicament se'n utilitzen dos, els altres són reservats.

0x00 Petició de realització del ping i retorn del primer paquet TCP de dades (dades de babord).

0x02 Petició de retorn del segon paquet TCP de dades (dades de estribord).

*Byte 19 – 25:*Reservats.
**0**

*Byte 26:* Byte final.
Aquest byte indica el final del paquet. El sistema deixa de esperar dades quan es troba aquest byte. És important tenir present que aquest byte no pot ser utilitzat en cap altre camp del paquet.
**0xFD.**

El paquet de resposta:

Aquest paquet anomenat "Interfície de retorn de dades (Interface Return Data)" és el format de paquet TCP/IP utilitzat per rebre les dades del sistema.
Encara que el format és exactament igual, es poden diferenciar dos tipus de resposta, les dades de babord i les d'estribord. Aquest paquets de resposta són enviats pel sistema una vegada a rebut correctament la petició corresponent.
La figura 4.13 mostra un taula em el format d'aquest paquet.

és interessant remarcar que dins de la resposta hi ha un camp que ens indica si s'ha produït algun error durant el procés.

| **Bytes** | **Descripció** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 - 7 | **0x49** | **0x56** | **0x58** | Estat petició | Rang | Freq. | Versió | Reservat |
| 8 – 11 | Reservat | Reservat | Num. bytes(HI) | Num. bytes(LO) | | | | |
| 12 fins 2011 | Dades de l'eco de babord o estribord (1000 bytes) | | | | | | | |
| 2012 | Final **0xFC** | | | | | | | |

*Figura 4.13: Taula de format del paquet de resposta Interface Return Data*

El paquet té una mida de 1013 bytes i segueix el següent format:

Byte 0: Capçalera de retorn de dades (Imagenex return data header)
**0x49** (Caràcter ASCII 'I')

Byte 1: Capçalera de retorn de dades (Imagenex return data header)
**0x56** (Caràcter ASCII 'V')

Byte 2: Capçalera de retorn de dades (Imagenex return data header)
**0x58** (Caràcter ASCII 'X')

Byte 3: Estat de la petició.
S'utilitza cada bit del byte per informar de l'estat de la petició.
Aquests són els estats que indiquen cada bit, sent el bit 7 el de més pes, i el 0 el de menys.

Bit 7: Accés de caràcters. Indica que s'ha desbordat algun valor de les dades de configuració.

Bit 6: Petició acceptada. Indica que el paquet de petició ha estat acceptat correctament.

Bit 5: Indica un error en el numero de paquet.

Bit 4: Indica un error en el valor del guany.

Bit 3: Indica un error en el valor de la freqüència.

Bit 2: Indica un error en el valor del rang.

Bit 1 – 0 : Canal de resposta. Pot ser un valor del 0 al 3, però hi ha 2 valors reservats i s'utilitzen únicament el 0 i el 2.
Un 0 indica que són les dades de babord.
Un 2 indica que són les dades d'estribord.

Byte 4: El rang utilitzat.

Byte 5: La freqüència utilitzada, sent:
Un 1 per 260kHz.
Un 2 per 330kHz.
Un 3 per 800kHz.

Byte 6: Versió del Firmware, inicialment 0x00.

Byte 7 – 9: Reservats.

Byte 10 – 11: Numero de bytes de dades, aquest valor sempre serà 1000.
El byte 10 és el de més pes i el 11 el de menys.

Byte 12 - 1011: Dades de retorn de babord o estribord. Es retornen 1000 bytes on cada un és un valor diferent. Els bytes estant ordenats per distancia respecte el transductor, de manera que el primer byte correspon a la mínima distancia, pròxima a zero, i l'últim byte correspon a la màxima distancia, igual al rang configurat.

Byte 1012: Byte final. Aquest byte indica el final del paquet.
**0xFC**

Funcionament general

Com ja s'ha explicat el sistema utilitza bàsicament un protocol de comunicació de pregunta i resposta.

En primer lloc s'envia el sistema un paquet de petició de dades *"Switch Data Command"* indicant que és una petició per realitzar un ping i retornar les dades de babord, obtingudes del eco d'aquest. Un cop s'han obtingut el paquet de resposta *"Interface Return Data"* amb les dades de babord, es torna a enviar un paquet de petició de dades, indicant que es una petició per rebre les dades d'estribord. El primer ping s'acaba quan s'ha rebut el paquet de resposta amb aquestes dades. Acte seguit, és comença el segon ping repetint el procés.

L'esquema de comunicació seria el següent:

1. S'envia al sistema un paquet TCP que contingui el Switch Data Command amb el byte 18 valen 0x00.

2. Es rep del sistema un paquet TCP amb el format Interface Return Data que conte les dades del canal de babord.

3. S'envia al sistema un altre paquet TCP que contingui el Switch Data Command aquesta vegada amb el byte 18 valen 0x02.

4. Es rep del sistema un paquet TCP amb el format Interface Return Data que conte les dades del canal d'estribord.

En aquest punt es tornaria a començar en el punt 1.

## 4.6 Programa Yellowfin

Juntament amb el sensor, la empresa Imagenex distribueix un software per a la plataforma Windows, anomenat "Yellowfin", el qual et permet comunicar-te amb el sensor, guardar-ne les dades i veure-les amb temps real.

Per guardar les dades, de manera que puguin ser reproduïdes més tard, el programa utilitza un format de fitxer anomenat .872 el qual està explicat a *l'annex A* d'aquest treball.

El programa compte bàsicament amb una única finestra principal en la qual es mostren les imatges capturades (vegeu figura 4.14). El sistema de representació que utilitza es basa en mostrar únicament les ultimes dades capturades, de manera que es mostra la imatge de tot el fons recorregut, com si fos una sola imatge que es va desplaçant per la pantalla.

Així doncs, contínuament van apareixen les noves imatges lineals capturades i van desapareixen les més velles.

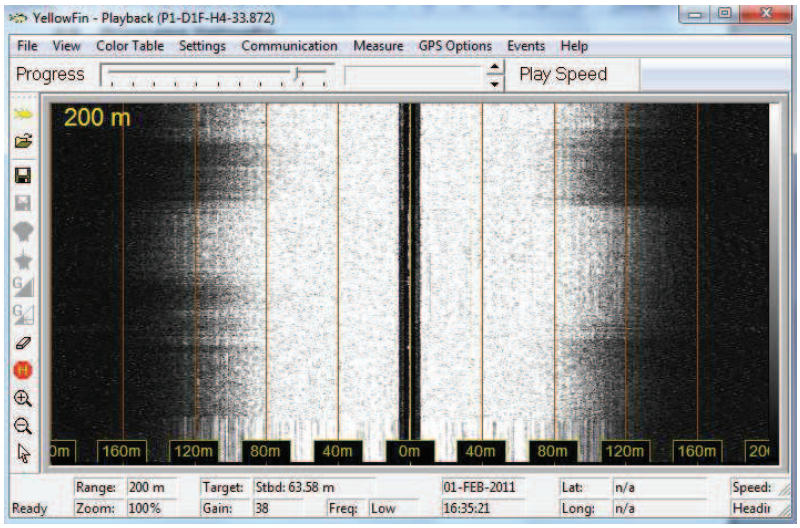


Figura 4.14: Captura del programa Yellowfin en execució.

A més de la imatge principal, cal remarcar un típic menú desplegable a la part superior de la finestra, una barra d'eines a la part esquerra, principalment per treballar amb les imatges, i a la part inferior, un requadre on es donen els paràmetres de configuració em els quals han estat preses les imatges.

A més a més, al programa també compte amb una barra de progrés i un seleccionador de velocitat per les reproduccions offline dels arxius .872.

## 4.7 Experimentació

Una vegada estudiat el sonar i el seu funcionament, abans de continuar amb el desenvolupament d'una primera aplicació de prova, és decideix provar el sensor i comprovar que funcioni correctament utilitzant el programa comercial *Yellowfin* que acompanya l'aparell.

D'entrada, es decideix realitzar una assemblatge temporal dels components del sonar, únicament per realitzar unes quantes proves.

L'assemblatge consisteix en connectar els dos transductors a la placa de control i connectar la placa de control a una font d'alimentació. Una vegada realitzat aquest muntatge, es connecta la placa de control a l'ordinador

mitjançant un cable Ethernet i acte seguit és configura la placa Ethernet de l'ordinador de manera que treballi a la mateixa subxarxa que el sensor.

Una cop acabats els preparatius, es col·loquen els transductors a la piscina del CIRS únicament per provar el programa i les principals característiques del sensor.

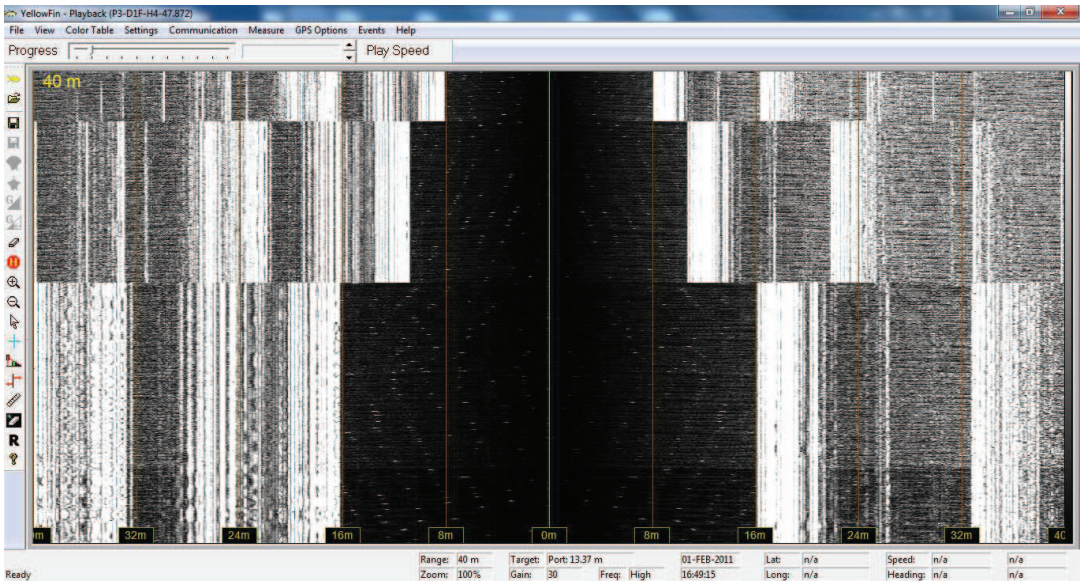


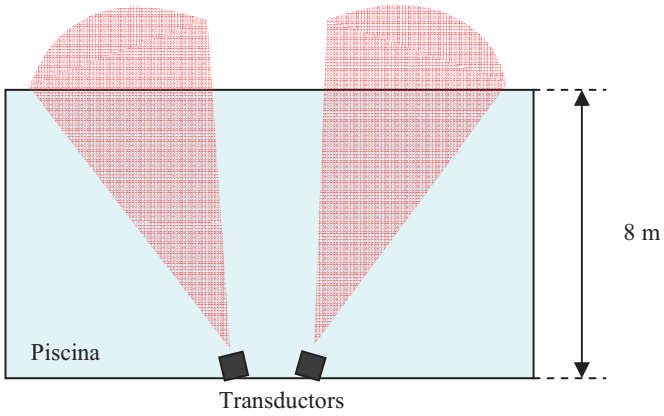

*Figura 4.15: Captura de la reproducció de la primera prova realitzada amb el programa Yellowfin.*

En aquesta primera prova, únicament es va verificar que tot funcionava correctament i es van provar de realitzar canvis en la configuració, com per exemple mostra la figura anterior, en el rang. En aquesta captura, es van disposar els transductors de manera que quedessin paral·lels a la piscina i acte seguit es va engegar el programa Yellowfin.

Coneixent que la piscina té una amplada de 8m es van provar diferents rangs i es va observar la distancia a la qual obteníem la primera reflexió. Correctament, en tots els rangs, la distancia canviava immediatament als 8 metres. A la figura 4.15 és poden diferenciar en una mateixa imatge, de dalt a baix, les proves amb els rang 40, 30 i 20 m.

La figura 4.16 mostra un esquema de la prova.

Situació real:




Imatge sonora obtinguda:

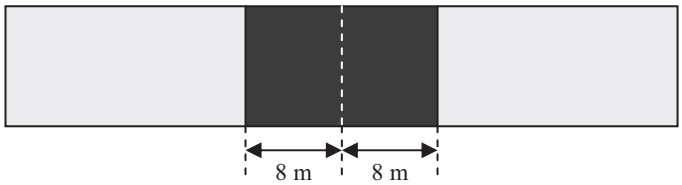



*Figura 4.16: Esquema d'una prova de funcionament realitzada amb el sensor*

Durant aquestes proves també és capturant el paquets TCP utilitzat en la comunicació entre el sensor i el programa Yellowfin, mitjançant el programa per capturar paquets per Windows *IP tool (vegeu la figura 4.17)*. D'aquesta manera és possible corroborar que el protocol s'ha antes correctament.

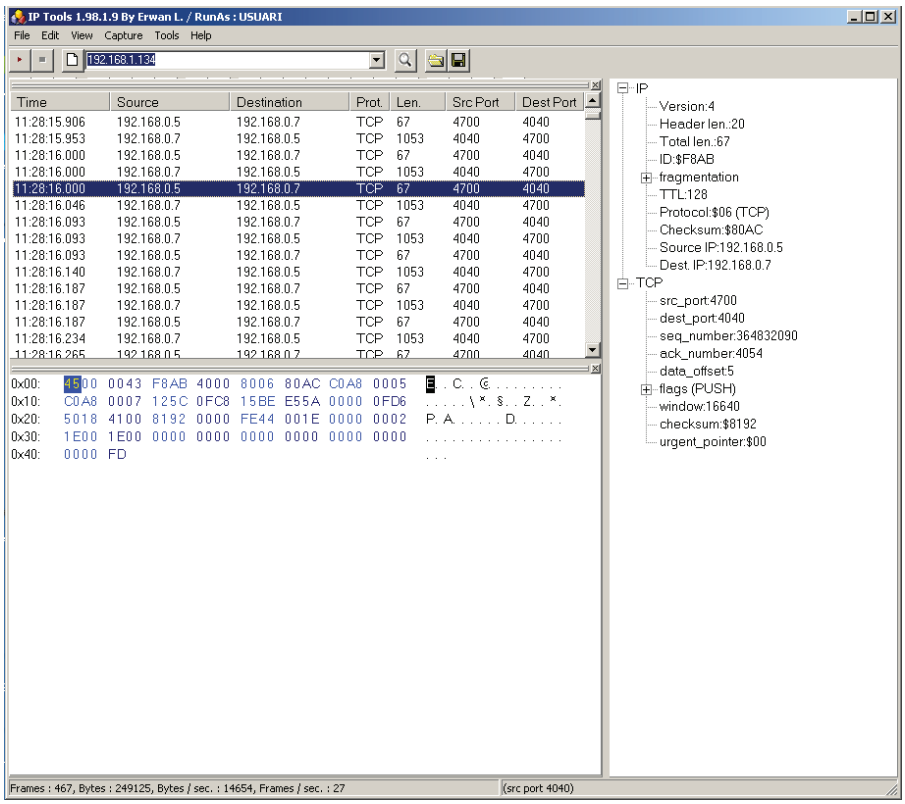



*Figura 4.17: Captura del programa IP tools amb les dades d'una comunicació amb el sensor.*

Si analitzem la figura 4.18, observem que els paquets segueixen una estructura de pregunta i resposta durant tota la comunicació. L'ordinador envia una petició el sensor, amb la IP 192.168.0.7 i port 4040, i aquest contesta amb la resposta, que de vegades seran les dades de l'eco de babord i de vagades les dades de l'eco d'estribord.

| | | | | | | |
|---|---|---|---|---|---|---|
| 11:28:15.906 | 192.168.0.5 | 192.168.0.7 | TCP | 67 | 4700 | 4040 |
| 11:28:15.953 | 192.168.0.7 | 192.168.0.5 | TCP | 1053 | 4040 | 4700 |
| 11:28:16.000 | 192.168.0.5 | 192.168.0.7 | TCP | 67 | 4700 | 4040 |
| 11:28:16.000 | 192.168.0.7 | 192.168.0.5 | TCP | 1053 | 4040 | 4700 |
| 11:28:16.000 | 192.168.0.5 | 192.168.0.7 | TCP | 67 | 4700 | 4040 |
| 11:28:16.046 | 192.168.0.7 | 192.168.0.5 | TCP | 1053 | 4040 | 4700 |
| 11:28:16.093 | 192.168.0.5 | 192.168.0.7 | TCP | 67 | 4700 | 4040 |
| 11:28:16.093 | 192.168.0.7 | 192.168.0.5 | TCP | 1053 | 4040 | 4700 |
| 11:28:16.093 | 192.168.0.5 | 192.168.0.7 | TCP | 67 | 4700 | 4040 |
| 11:28:16.140 | 192.168.0.7 | 192.168.0.5 | TCP | 1053 | 4040 | 4700 |
| 11:28:16.187 | 192.168.0.5 | 192.168.0.7 | TCP | 67 | 4700 | 4040 |
| 11:28:16.187 | 192.168.0.7 | 192.168.0.5 | TCP | 1053 | 4040 | 4700 |
| 11:28:16.187 | 192.168.0.5 | 192.168.0.7 | TCP | 67 | 4700 | 4040 |
| 11:28:16.234 | 192.168.0.7 | 192.168.0.5 | TCP | 1053 | 4040 | 4700 |

*Figura 4.18: Captura del programa IP tools amb les direccions dels paquets capturats.*

També podem analitzar cada tipus de paquet observant que efectivament segueixen el format de paquets estudiats.

```
0x00:  4500 0043 F8AB 4000 8006 80AC C0A8 0005   E..C..@.........
0x10:  C0A8 0007 125C 0FC8 15BE E55A 0000 0FD6   .....\.....Z....
0x20:  5018 4100 8192 0000 FE44 001E 0000 0002   P.A.....D......
0x30:  1E00 1E00 0000 0000 0000 0000 0000 0000   ................
0x40:  0000 FD                                   ...
```

*Figura 4.19: Captura del programa IP tools d'un paquet Switch Data Command.*

En el paquet de petició *Switch Data Command* enviat per l'ordinador de la figura 4.19, deixant de banda els valors reservats hi podem identificar els valors:

- 0xF4 i 0x44, la capçalera del paquet
- 0x1E (30), el rang.
- 02, la freqüència.
- 0x1E (30), el guany.
- 0x1E (30), el balanç del guany.
- 00, el numero de paquet, això vol dir que es tracta de la primera petició.
- 0xFD, el byte final.

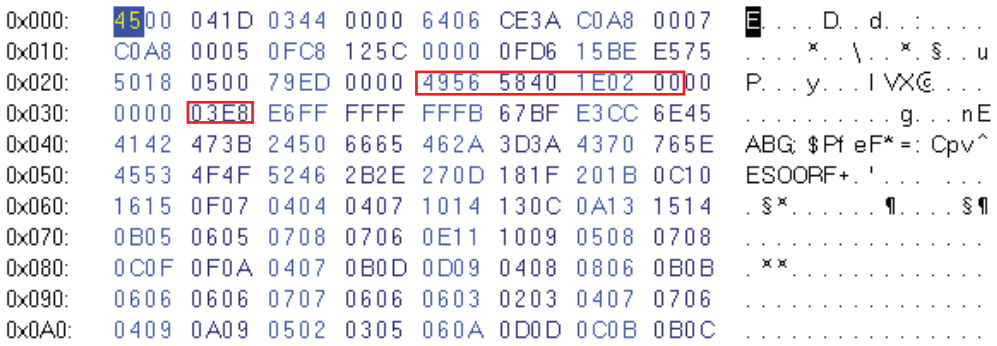



*Figura 4.20: Captura del programa IP tools d'un fragment d'un paquet Interface Return Data.*

De la mateixa manera que hem fet amb el paquet de petició, també analitzem el format i els valors del paquet de resposta *Interface Return Data* enviat pel sensor que podem observar, en part, a la figura 4.20. A part dels valors reservats, els valors identificats són els següents:

- 0x49, 0x56, 0x58, corresponent a la capçalera del paquet
- 0x40, l'estat de la petició. En binari 01000000, per tant, tot correcta.
- 0x1E (30), efectivament el rang introduït a la petició.
- 02, la freqüència també es correspon.
- 00, la versió, sempre 0.
- 0x03 i 0xE8, efectivament és el valor invariable 1000, 0x03 el byte de més pes i 0xE8 el byte de menys pes.
- A continuació, comencen les dades de l'eco de babord.

# 5 Desenvolupament d'un software de comunicació amb el sensor

## 5.1 Introducció

Un cop s'ha estudiat el funcionament del sensor i s'ha experimentat amb el software original proporcionat pel fabricant, el següent pas és desenvolupar el nostre propi software que ens permeti comunicar-nos amb aquest.

L'objectiu d'aquest primer programa és realitzar una primera aproximació a la solució final en la qual ens trobarem amb els primers problemes de comunicació i ens servirà per perfilar l'estructura principal que haurà de seguir el programa definitiu.

Per visualitzar les dades capturades amb aquest primer programa s'ha decidit generar un arxiu .872 seguint el format utilitzat pel programa propietari Imagenex Yellowfin, de manera que sigui possible veure els resultats obtinguts de manera offline utilitzant aquest programa i evitar-nos així, de realitzar una interfície gràfica. Es pot trobar tota la informació sobre aquest format de fitxer a l'*annex A* d'aquest projecte.

El realitzar el disseny i la implementació d'aquest software s'ha tingut en conta, que més endavant s'utilitzarà aquest mateix programa com a base per dissenyar la solució final implementada en el robot.

## 5.2 Disseny i implementació d'un software basic per GNU/Linux en C++

Disseny

El disseny del programa esta basat amb 3 classes les quals s'encarreguen de realitzar les principals funcions del programa; realitza una connexió TCP, generar un arxiu .872 i controlar el sonar.

*CEthernetTest:*

Classe que realitza una conexió TCP, sobre la interfície Ethernet, entra el sonar i l'ordinador. Aquesta classe segueix la estructura d'una entitat client en una arquitectura client-servidor.

Els mètodes públics de la classe són els següents:

*CEthernetTest()*: Constructor de l'objecte.

***void*** *connect( const std::string& IP, const std::string& port )*: Mètode que inicia una connexió orientada a la connexió amb el protocol TCP sobre el model TCP/IP.

***void*** *connect( const std::string& IP, const int port )*: Sobrecarrega del mètode *connect*.

***void*** *send( const std::string& dataString )*: Mètode per enviar un cadena de caràcters a traves de la connexió oberta.

***void*** *send( const char* data, int nBytes ):* Sobrecarrega del mètode *send*.

***void*** *receive( std::string& dataString, int& nBytes ):* Mètode que per rebre dades de la connexió en forma de cadena de caràcters.

***void*** *receive( char* data, int& nBytes* ): Sobrecarrega del mètode *recieve*.

***void*** *close():* Mètode per tancar la connexió.

C872File:

Aquesta classe s'encarrega de generar un fitxer .872, seguint el format explicat en l'*annex A* d'aquest projecte, d'aquesta manera les dades capturades pel sensor es podran veure en el programa propietari Imagenex Yellowfin que acompanya el sensor.

Els seus mètodes públics són els següents:

*C872File():* Constructor de l'objecte.

***void*** *open(char* filename)*: Mètode que genera i obre un arxiu amb el nom passat com a paràmetre.

***void*** *inputData(char* portData, char* starBoardData, Byte freq, Byte range, Byte dataGain, char* headerBuffer)*: Introdueix les dades d'un ping a l'arxiu.

***void*** *close():* Mètode que tanca l'arxiu generat.

*CSideScanSonar:*

S'encarrega de controlar el funcionament del sensor de manera continuada. Aquest classe fa ús de les altres dos classes, *CEthernetTest* i *C872File*. Utilitza la classe *CEthernetTest* per controlar la connexió TCP, bàsicament; connectar, enviar i rebre paquets, i fa ús de la classe C872File per generar un fitxer compatible amb al programa propietari Imagenex Yellowfin.

Els seus mètodes públics són els següents:

*CSideScanSonar():* Constructor de l'objecte.

***void*** *runEndless():* Mètode que executa el sonar de manera infinita. Realitza pings de forma continuada i mentrestant genera un fitxer .872 amb les dades obtingudes, tal i com es mostra en el pseudocodi de la figura 5.1.

```
Funció runEndless()
        Connectar amb el sonar
        Mentre (Cert)
                Enviar un Switch Data Command amb el byte 18 = 0
                Rebre un Interface Return Data amb dades babord
                Enviar un Switch Data Command amb el byte 18 = 2
                Rebre un Interface Return Data amb dades estribord
                Guardar dades de babord i d'estribord a l'arxiu .872
        FMentre
FFunció
```

*Figura 5.1: Pseudocodi del mètode runEndless() utilitzat per controlar el sonar.*

Implementació:

S'ha escollit la implementació sobre una plataforma GNU/Linux i el llenguatge C++ per ser el sistema i el llenguatge utilitzat per l'arquitectura de control COLA2 on finalment serà implementat. També, s'ha utilitzat la llibreria de lliure distribució Boost per realitzar les comunicacions TCP i s'ha seguit l'estil de programació C++ utilitzat en l'arquitectura COLA2 em la intenció d'acostar aquesta primera implementació a la solució final.

En aquesta implementació es destacable la implementació del paquet Switch Data Command com una tupla per facilitar la modificació dels seu bytes de configuració, mentre que el paquet Interface Return Data, únicament de lectura, és tractat com una matriu unidimensional.

## 5.3 Comprovació del funcionament

Un cop s'ha realitzat la implementació, s'han realitzat algunes proves de funcionament amb el sensor. En aquestes proves simplement es comprova que les dades son enviades, rebudes i interpretades correctament pel sensor i que les dades tornades pel sensor siguin consistents.

Per realitzar aquestes proves, com ja s'ha realitzat abans, s'utilitza un assemblatge temporal del sensor i es treballa amb els transductors dins un contenidor (vegeu la figura 5.2).

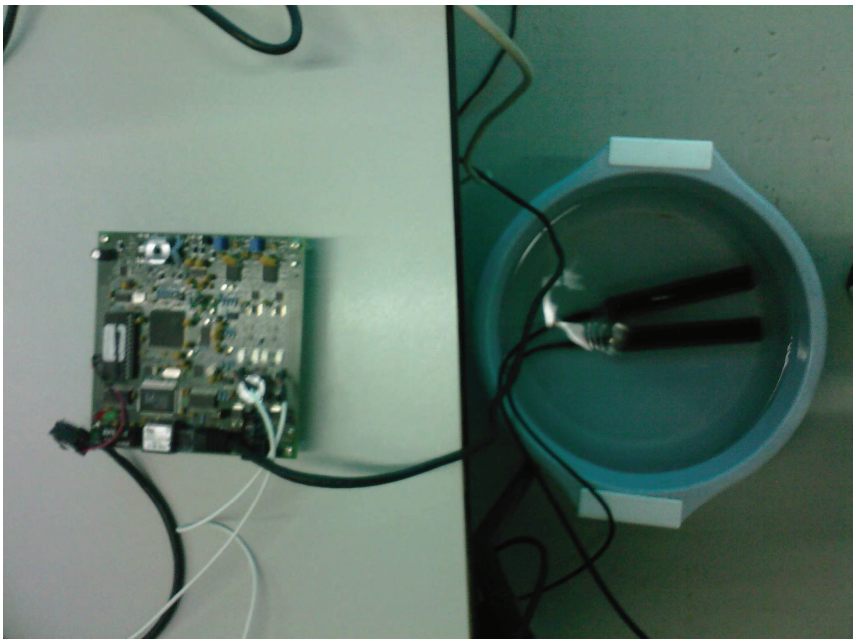


Figura 5.2: Assemblatge temporal del sonar d'escombrat lateral Imagenex model 872

Inicialment és realitzen proves de comunicació entre el sensor i el programa on únicament s'analitza el format i el contingut dels paquets TCP utilitzats en la comunicació, amb el programa per capturà paquets per GNU/Linux *Wireshark*.

Aquests paquets capturats es van comparà amb els paquets capturats anteriorment, amb el programa per Windows *IP tools*, quan s'utilitzava el programa propietari per Windows Yellowfin. En la comparació, es va observar que el format i els valors en els paquets TCP enviats utilitzant els dos programes eren els mateixos per les mateixes configuracions.

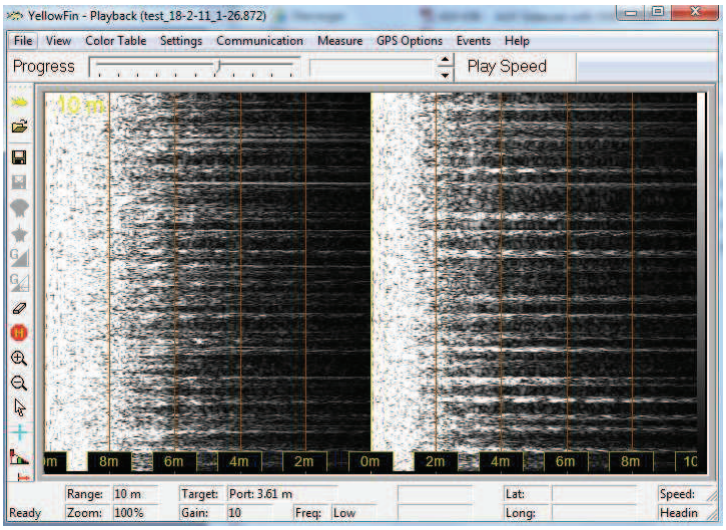



Figura 5.3: Captura del programa Yellowfin reproduint un arxiu .872 generat amb el programa per GNU/Linux. L'arxiu té erròniament les línees de babord invertides.

Un altre prova va ser la visualització d'un arxiu .872 capturat amb el nostre programa, en el programa yellowfin. En aquesta prova es va observar que el format utilitzat era en general correcte i funcional, però que era necessari invertir les línees de babord ja que apareixien al revés de l'habitual.

# 6 Integració al robot Girona 500

## 6.1 Introducció

Un cop s'ha realitzat un primer programa de prova i s'ha comprovat el seu correcte funcionament, el següent pas és la integració del sensor en el robot Girona 500.

En el procés integració del sensor en el Girona 500 distingim dos parts diferents que és van realitzar en paral·lel; la integració física i la integració software.

## 6.2 Integració del sensor a l'estructura del robot

### 6.2.1 Introducció

La integració física del sensor al robot va ser realitzar per membres del grup VICOROB paral·lelament a la integració del software. Podem distingim dos parts ben diferenciades.

### 6.2.2 Instal·lació elèctrica

En la instal·lació elèctrica és va substituir la instal·lació provisional realitzada únicament per provar el sensor, per una instal·lació definitiva totalment integrada amb el robot.

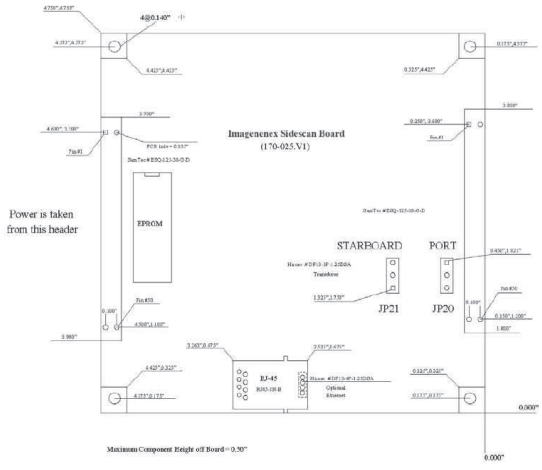



*Figura 6.1: Placa de control del sonar Imagenex model 782*

En aquest apartat distingim tres tasques principals:

1. Connexió dels dos transductors a la placa de control (vegeu la figura 6.1).
2. Connexió de la placa de control a una font d'alimentació.
3. Connexió de la placa de control a la xarxa del robot mitjançant la interfície Ethernet.

### 6.2.3 Instal·lació mecànica

En la instal·lació mecànica és van realitzar tres tasques principals:

1. Instal·lació dels transductors a l'estructura del robot (vegeu la figura 6.2).
2. Instal·lació de la placa de control.
3. Adaptació de tot el material utilitzat per suportar profunditats de 500 metres.

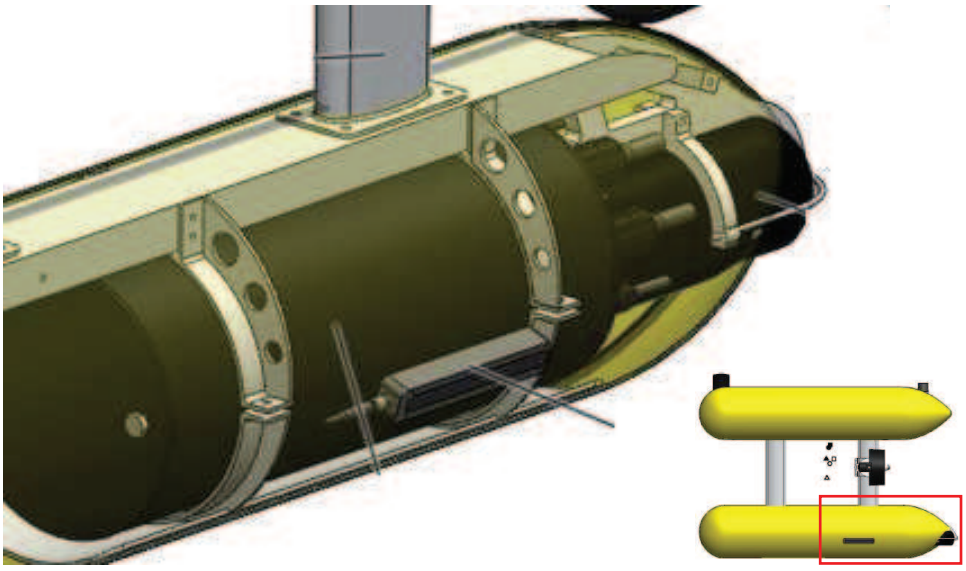


*Figura 6.2: Representació en 3D de la posició del sonar d'escombrat lateral en el robot Girona 500*

## 6.3 Integració del sensor a l'arquitectura de control COLA2

### 6.3.1 Introducció

Un cop s'ha estudiat el funcionament de l'arquitectura COLA2 i s'ha realitzat un primera aplicació de prova funcional, el següent pas és la integració del sensor a l'arquitectura.

Inicialment, quan plantegem la integració del sensor a l'arquitectura COLA2, podem diferenciar 3 parts bàsiques; un controlador TCP/IP per comunicar-se em la placa de control, un *driver* encarregat de controlar el sonar i addicionalment, una interfície gràfica per veure les dades.

De les dos primeres parts, ja hem implementat una primera aproximació en el programa de prova per GNU/Linux i ara el que caldrà es adaptar les classes de manera que compleixin les interfícies de l'arquitectura. Pel que fa a la interfície gràfica serà necessari implementar-la des de zero.

Referent al guardat de les dades obtingudes pel sensor, simplement, es generarà un log estàndard de l'arquitectura COLA2, de manera que és guardaran les dades amb aquest format i més endavant, en aquest projecte, es realitzarà un programa per poder mostrar aquestes dades.

### 6.3.2 Desenvolupament d'un device per realitzar connexions TCP

Quan afrontem la integració del sensor a l'arquitectura COLA2, el primer que observem és la necessitat d'implementar una interfície d'entrada/sortida que ens permet-hi comunicar-nos amb el sensor a traves del protocol TCP del model TCP/IP, ja que actualment l'arquitectura no compta amb aquesta interfície.

Per desenvolupar aquest *device*, utilitzarem com a base la classe *CEthernetTest,* creada i testejada anteriorment, adaptant-la inicialment a la classe interfície IIODevice. A més a més, serà necessari implementar altres operacions típiques de les comunicacions amb el model TCP/IP com per exemple *connectar*.

Igual que *CethernetTest*, aquesta classe ens ha de permetre mantenir una comunicació orientada a la connexió TCP amb un servidor especificat.

Aquesta nova classe anomenada *CTcpSocket,* implementarà els següents mètodes:

*CTcpSocket():* Constructor de l'objecte.

*void connect( const std::string& IP, const unsigned short port ):* Mètode que ens permet iniciar una connexió TCP en una direcció remota.

***unsigned char** readByte ( const unsigned int msTimeout = 0 ):* Mètode que ens permet llegir un byte de l'stream de la connexió TCP.

***void** read ( DataBuffer& dataBuffer,const unsigned int numOfBytes = 0, const unsigned int msTimeout = 0 ):* Mètode que ens permet llegir un numero de bytes concrets de l'stream de la connexió TCP.

***void*** *readUntil( DataBuffer& dataBuffer, const unsigned char delimiter = '\n', const unsigned int msTimeout = 0 )*:Mètode que ens permet llegir una cadena de caràcters de l'stream fins que trobem un byte especificat.

*std::string readLine ( const unsigned int msTimeout = 0, const unsigned char lineTerminator = '\n' )*: Mètode que ens permet llegir una cadena de caràcters de l'stream.

***void*** *writeByte ( const unsigned char dataByte ):* Mètode que ens permet escriure un byte a l'stream de la connexió.

***void*** write ( const DataBuffer& dataBuffer ): Mètode que ens permet escriure un nombre de bytes a l'stream

***void*** *write ( const std::string& dataString )*: Mètode que ens permet escriure una cadena de caràcters a l'stream.

### 6.3.3 Desenvolupament d'un driver pel sensor

Una vegada desenvolupat el device per realitzar connexions TCP, el següent pas és desenvolupar un nou driver el qual pugui controlar el sensor. Aquesta nova classe serà una adaptació de la classe *CSideScanSonar,* del primer programa realitzat, a un component de l'arquitectura COLA2.

Observem, que aquest component tindrà un device d'entrada/sortida, que serà una instancia de la classe *CTcpSocket* basada en la interfície *IIODevice,* la qual s'utilitzarà per comunicar-se amb el sensor. També em de tenir present, que a cada execució s'haurà de generà un log amb totes les dades capturades, de manera que després pugi ser visualitzat offline.

En primer lloc, serà necessari establir els paràmetres de configuració que tindrà el nou driver. Aquests paràmetres seran els següents:

- *device_timeout.* Temps d'espera màxim en el device d'E/S
- *device_ip.* La IP on es troba el sensor.
- *device_port.* El port on es troba el sensor.
- *range.* Valor del rang del sensor.
- *frequency.* Valor de freqüència del sensor.
- *data_gain.* Valor de guany del sensor.
- *balance_gain.* Valor del balanç del guany del sensor.

La figura 6.3 ens mostra una captura del fitxer de components on apareix la configuració del sonar.

```
<imagenex_sidescan_sonar_configuration>
    <device_timeout type="integer" value="1000" /> <!-- ms -->
    <device_ip type="string" value="192.168.0.7" />
    <device_port type="integer" value="4040" />
    <!-- Range: (meters) -->
    <!-- Low/Medium frequency: 10,20,30,40,50,60,80,100,125,150,200 -->
    <!-- High frequency: 10,20,30,40,50 -->
    <range type="integer" value="10" />
    <frequency type="integer" value="0" /> <!-- 0: low, 1: medium, 2: high -->
    <data_gain type="integer" value="10" /> <!-- 0dB to 40dB -->
    <balance_gain type="integer" value="30" /> <!-- 0 to 60 => +/-3dB on 0.1dB increments -->
</imagenex_sidescan_sonar_configuration>
```

*Figura 6.3: Captura de configuració del sonar d'escombrat lateral Imagenex*

En segon lloc, és necessari definir les classes data *manipulation*, les quals ens serviran per moure les dades des de i cap a XML, que com ja em vist, és el format utilitzat pel sistema per intercanviar les dades. En aquest cas necessitem dos classes:

- *ConfigDataManip*. La classe que tenen tots els components amb la configuració. Usa una tupla amb els següents elements:
  - int deviceTimeout
  - std::string deviceIP
  - int devicePort
  - int range
  - int frequency
  - int dataGain
  - int balanceGain ;Device_timeout

- *CImagenexSidescanSonarDataManip*. La classe amb les dades que es guardaran el log. Usa una tupla em els següents elements:
  - int serialStatus
  - int range
  - int frequency
  - int gain
  - Eigen::VectorXd portChannel
  - Eigen::VectorXd starboardChannel

Aquestes dos classes seran instanciades a traves dels objectes privats *configDataManip_* i *imagenexSidescanSonarDataManip_.*

Un cop definit l'arxiu de configuració i les classes manipuladores de dades, definim el mètodes públics del component. Com tots els components ha de implementar els següents mètodes públics:

*CImagenexSidescanSonar( const std::string& componentName )*: Constructor de la classe, inicialitzem tots els elements de la classe.

***void** initialize()*: Mètode que inicialitza el component. Dins d'aquest mètode serà necessari realitzar les següents tasques:

1. Carregar la configuració del component.
2. Establir la connexió amb el sensor a través del *device*.
3. Preparar un log per guardar les dades obtingudes pel sensor, de manera que després es puguin mostrar offline.
4. Executar un fil d'execució no periòdic amb la funció principal que controlarà el sensor, per conveni *run().*

*std::string getConfiguration() const*: Mètode que ens retorna la configuració del component, en format XML.

**void** setConfiguration( const std::string& configuration ): Mètode que ens permet assignar una configuració el component, passada en format XML.

Pel que fa els mètodes privats de la classe, aquest són els més importants:

***void*** *run()*: Aquest mètode privat és el fil d'execució iniciat pel mètode públic *inicialize().* Aquest mètode és l'encarregat de dur a terme l'algorisme que es comunica em el sensor, per tant, tindrà un algoritme semblant al que s'ha utilitzat a la classe *CSideScanSona,* en el mètode *runEndless().* En aquest cas, però, el mateix mètode no s'encarregarà de generar i enviar els paquets, sinó que utilitzarà dos mètodes privats; *sendSDC i recieveIRD.* També s'ha de destacar, que en cas que es produeixi un error menor es torna a enviar una altre vegada un petició de ping.

La figura 6.4 mostra l'algorisme basic amb pseudocodi.

```
Mètode run()
numeroPaquet := 0
        Mentre (Cert)
                sendSDC ( NumeroPaquet )
                recieveIRD( NumeroPaquet )

                Si numeroPaquet = 0 i no Error llavors
                        numeroPaquet := 2
                altrament
                        numeroPaquet := 0
                Fsi
        Fmentre
Fmètode
```

*Figura 6.4: Pseudocodi del algorisme basic del mètode privat run()*

***void*** *sendSDC( unsigned char packetNumber )*: Mètode que envia un paquet TCP amb el format Switch Data Command, el qual el byte 18 te al valor del *packetNumber* passat com a paràmetre, amb els valors vàlids 0 o 2. Cal destacar, que la classe usa una matriu unidimensional, on cada

posició correspon a un byte, per guardar el paquet i que comprova la correctesa de tots els valors abans d'enviar-lo.

**void** recieveIRD( unsigned char packetNumber ): Mètode que espera un paquet TCP amb el format Interface Return Data i en tracta les dades. Depenent del numero de paquet passat com a paràmetre, tractarà les dades com a dades de babord o com a dades d'estribord. Així, cada vegada que es rebin les dades d'estribord, es publicaran les dades dels dos canals mitjançant la operació *publish* i es guardaran en el log a traves de l'objecte *logger*. Per realitzar aquestes dos tasques també s'utilitzarà l'objecte *imagenexSidescanSonarDataManip_* el quan ens permetrà guardar temporalment les dades i passar-les a XML.
En aquest cas, igual que el mètode *sendSDC*, també s'utilitza una matriu unidimensional i es comprova que el paquet rebut sigui correcta.

### 6.3.4 Desenvolupament d'una interfície gràfica per representar les dades amb temps real

Una vegada implementat el driver, el següent pas és desenvolupar una interfície gràfica de manera que sigui possible veure les dades capturades pel sonar en temps real, de manera semblant al programa comercial Yellowfin.

La idea general, és que les imatges del fons marí es desplacin per la pantalla, de manera que sempre puguem veure les ultimes línees capturades i que desapareguin aquelles més velles. La figura 6.5 mostra un disseny conceptual de la finestra.

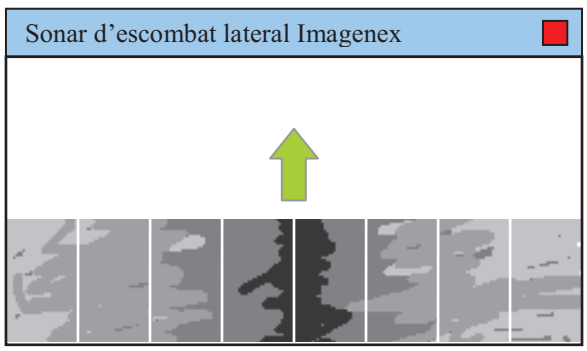



Figura 6.5: Disseny conceptual de la finestra de la interfície gràfica.

Aquesta interfície serà implementada dins el mateix component que el driver i serà implementada en part, usant la llibreria *OpenCV*. S'ha decidit utilitzar aquest llibreria perquè ja forma part de l'arquitectura i té totes les funciones necessàries per visualitzar i tractar imatges.

En el desenvolupament d'aquesta interfície és prioritzarà la eficiència. Em pogut observar que el sensor és molt ràpid capturant dades, i un tractament

lent de les imatges juntament amb l'execució de molts components a la vegada, podria disminuir la velocitat de captura del dispositiu. Així dons, hauria de ser possible poder activar i desactivar la interfície gràfica, com també escollir la seva freqüència de refresc.

En primer lloc, planifiquem l'algorisme utilitzat per mostrar les imatges.

L'algorisme, utilitzarà únicament dos *buffers* de memòria. Un objecte *cv::Mat,* anomenat *imageView*, que utilitzarem per guardar les imatges de *les OpenCV* i una matriu de *chars,* anomenada *circularBuffer,* utilitzada com molt bé indica el seu nom, com a *buffer* cíclic o circular.

El *buffer* cíclic funciona de manera que s'hi aniran introduint dades, per ordre, i en arribar al final del *buffer*, el que es fa, es tornar el principi, de manera que les noves dades començaran a sobreescriure les velles. Aquest sistema es realitza contínuament, de manera que el buffer sempre guarda les ultimes dades sobreescrivint les més velles.

Per mantenir aquest *buffer,* s'utilitzaran dos variables enteres; *bufferPos* i *bufferSize*. *bufferPos* guarda la ultima posició escrita, de manera que així es pot conèixer on han d'anar les noves dades. *bufferSize* guarda simplement la mida del *buffer,* que en el nostra cas, variarà segons la mida de la imatge que es vol mostrar.

La figura 6.6 mostra un esquema del seu funcionament.

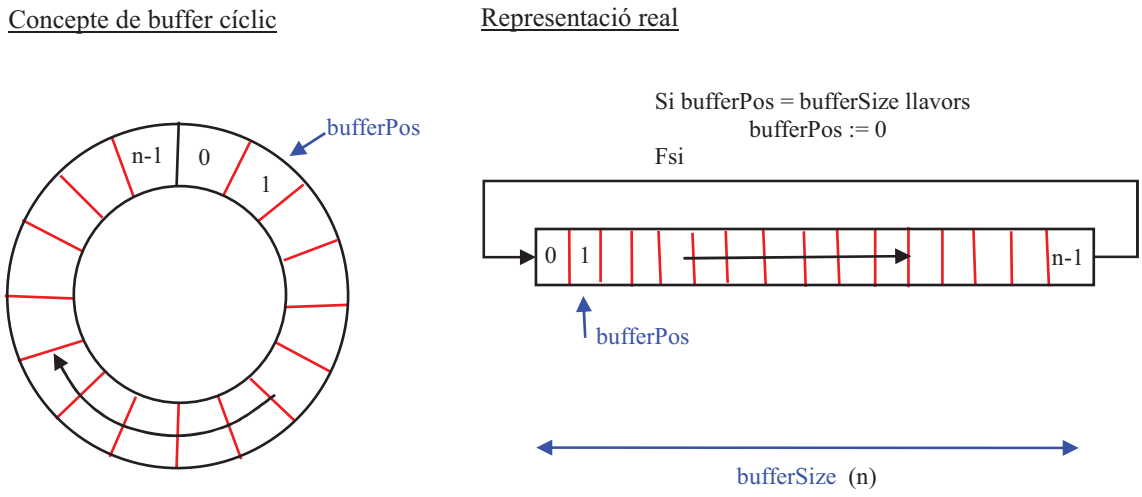



Figura 6.6: Esquema del funcionament d'un buffer cíclic

L'algorisme consisteix bàsicament en emplenar el *buffer* circular ordenadament, amb cada una de les noves imatges lineals sonores obtingudes pel sensor. De manera que, cada vegada que es vulgui mostrar la imatge sencera per pantalla, és realitzarà una copia del *buffer* circular a la *cv:.Mat*, de tal manera que el buffer *cv::Mat* quedi ordenat de forma correcta. Finalment, únicament

necessitarem mostrar per pantalla aquest *buffer* utilitzant les OpenCV. La figura 6.7 mostra un esquema d'aquest sistema.

D'aquesta manera, no és necessari actualitzar o refrescar la imatge, copiar el *buffer* cíclic a cv:Mat, cada vegada que es captura un nova imatge lineal i és possible establir un temps d'espera entre les actualitzacions.

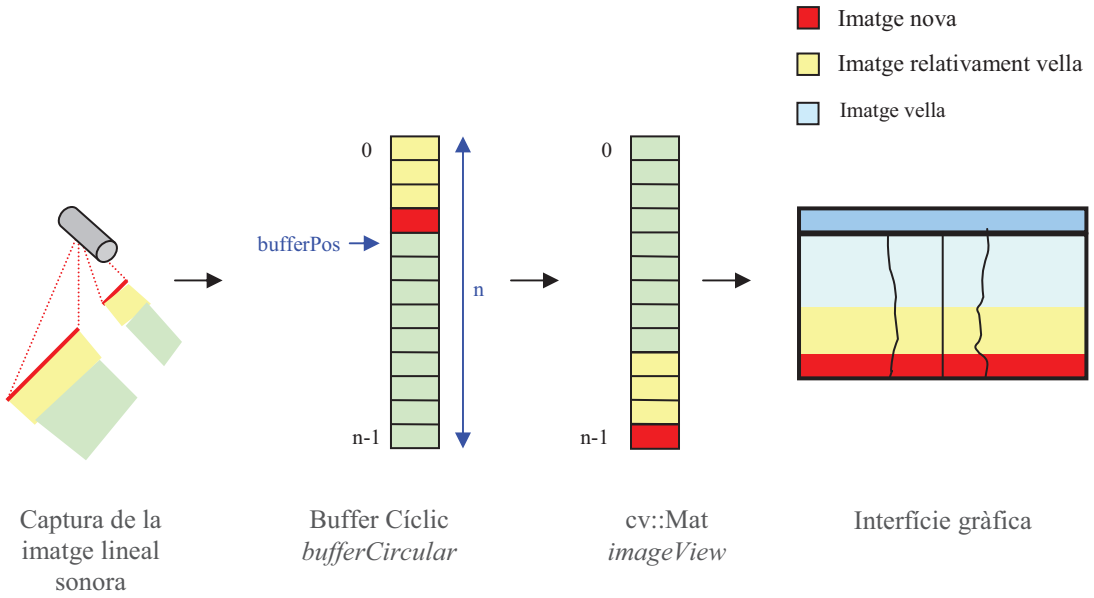



*Figura 6.7: Esquema basic del funcionament de l'interficial gràfica*

Un altre punt important de l'algorisme es la copia del b*uffe*r cíclic a el *buffer cv::Mat*. Aquesta copia es realitzarà de manera que en el buffer cv::Mat, el qual serà mostrat per pantalla, les imatges lineals quedin ordenades cronològicament. Per realitzar la copia de les dades s'ha decidit utilitzar la funció *memcpy* de *ANSI C* per la seva eficiència.

Juntament amb la creació de la interfície gràfica, també s'han establert nous paràmetres de configuració en el component.

Aquest nous paràmetres van tots precedits per la paraula *image* i són els següents:

- *image_create*. Mostrar les imatges en temps real.
- *image_width_px*. Amplada de la imatge en píxels.
- *image_height_px*. Altura de la imatge amb píxels.
- *image_ping*. Temps entre la captura de dos ping de la imatge.
- *image_refresh*. Temps d'espera per actualitzar la imatge.

Juntament amb la modificació del fitxer de components és necessari modificar la classe ConfigDataManip per acceptar aquest nous paràmetres.

Els mètodes privats afegits per crear la interfície gràfica també tenen la paraula *image* el davant i són els següents:

***void*** *imageInitializeView()*: Aquest mètode inicialitza i activa la interfície gràfica per veure la imatge sonora capturada pel sensor amb temps real.

***void*** *imagestopView()*: Mètode que desactiva la interfície gràfica del *driver.*

***void*** *imageAddChannel()*: Mètode que afegeix un canal (babord o estribord) al *buffer* circular. Si el canal és el de babord, a més a més ha de ser invertit, de manera que pugui ser visualitzat correctament.

***void*** *ImageDraw()*: Mostra la imatge actual que hi ha a el *buffer* circular. Primerament copia ordenadament el *buffer* circular a el *buffer* cv::Mat i tot seguit mostra el *buffer* cv::Mat.

***void*** *imageScaleChannel()*: Mètode que escala un canal (babord o estribord) a la amplada de la imatge.

La figura 6.8 ens mostra una captura de la interfície gràfica funcionant.

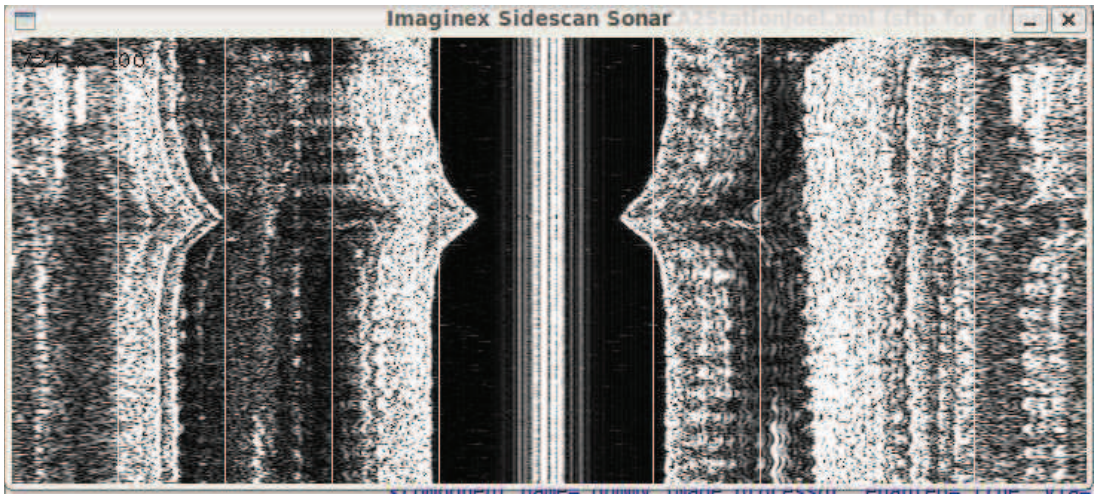



*Figura 6.8: Captura de la interfície gràfica implementada en el COLA2*

Finalment, per integrar la interfície gràfica en el driver serà necessari afegir la funció *imageInitializeView()* en el mètode *initialize().* A més a més, cada vegada que s'obté un nou paquet del sonar, si ha passat el temps especificat en el paràmetre *image*_ping de la configuració, s'haurà de redimensionar i afegir el canal obtingut en el *buffer* cíclic, per mitjà de les funcions *imageScaleChannel()* i *imageAddChannel()*. Finalment, s'haurà d'executar la funció image*Draw()* cada vegada que transcorri el temps especificat en el paràmetre *image_refresh* de la configuració.

# 7 Software de representació d'imatges acústiques obtingudes pel robot

## 7.1 Introducció

Un cop el sensor ja es troba integrat completament en el robot, ens sorgeix la necessitat de mirar de manera offline les imatges guardades en els logs, generats pel robot.

Davant la necessitat de fer una nova aplicació per reproduir els logs, es va decidir optà per utilitzar l'entorn i el llenguatge Matlab, degut a la seva facilitat d'us i a la amplia gama de funcions que incorpora.

En aquesta aplicació, únicament és vol copiar la interfície gràfica integrada en el robot, però aquesta vegada per reproduir els logs offline.

## 7.2 El logs de dades

Inicialment, es necessari que coneguem el format utilitzar pels logs generats pel driver del COLA2. Aquest logs són fitxers en text pla els quals tenen una entrada de dades per línia, a excepció de la primera, la qual és la capçalera del log on hi podem trobar el nom de cada camp.

A cada línia hi trobem, primer de tot, la data en que s'ha introduït la línia, tot seguit les dades separades per tabuladors i finalment un símbol de tant per cent *'%'* , un espai i el nom del component que ha generat el log.

```
1  2011 07 26 15 10 57 468 serial_status   range(m)    frequency   gain(dB)
2  2011 07 26 15 11 00 084 66  20  2   10  17 16 14 9 6 6 6 5 3 0 1 1 1 2 2 1
3  2011 07 26 15 11 00 129 66  20  2   10  7 4 0 2 3 2 2 2 1 0 1 1 1 2 4 5 5
4  2011 07 26 15 11 00 174 66  20  2   10  10 7 3 2 4 5 4 2 0 1 1 0 0 1 2 2 2
5  2011 07 26 15 11 00 219 66  20  2   10  8 3 3 6 7 6 3 1 1 1 1 1 1 1 2 2 2
6  2011 07 26 15 11 00 263 66  20  2   10  6 2 3 5 6 6 4 2 1 2 2 1 2 2 2 1 2
7  2011 07 26 15 11 00 308 66  20  2   10  5 1 3 5 6 4 2 2 2 2 1 2 3 3 3 2 3
8  2011 07 26 15 11 00 353 66  20  2   10  4 2 5 7 7 5 2 0 1 1 2 2 2 1 3 3 3
9  2011 07 26 15 11 00 397 66  20  2   10  1 3 5 5 5 4 3 2 1 2 2 2 1 1 2 2 1
10 2011 07 26 15 11 00 442 66  20  2   10  7 3 2 5 5 3 0 3 4 4 2 0 1 2 2 3 3
11 2011 07 26 15 11 00 487 66  20  2   10  7 5 4 4 4 3 1 1 1 1 1 1 2 2 2 2 2
```

*Figura 7.1: Fragment d'un log generat pel driver del sonar d'escombrat lateral.*

La data ocupa 7 columnes que són, en aquest ordre; any, mes, dia, hora, minut, segon i milisegon.

Les dades guardades en el log són, en aquest ordre, les següents,:

1. L'estat de la petició obtinguda del segon paquet de Interface Return Data, el que porta les dades d'estribord. Si tot ha estat correcta durant la captura, aquest valor hauria de ser 66 en decimal.
2. El rang usat en la captura, en metres.
3. La freqüència, amb els valors de 0,1 i 2, els quals signifiquen 260, 330 i 770 kHz.
4. El guany, en decibels.
5. Les 1000 dades ordenades de babord.
6. Les 1000 dades ordenades d'estribord.

La figura 7.1 mostra un fragment d'un log generat pel driver.

## 7.3 Disseny i Implementació del programa

Abans de centrar-nos en l'aplicació, necessitem una funció que ens retorni una matriu de valors a partir del fitxer del log, de manera que pugem treballar fàcilment amb les dades. Aquesta funció l'anomenarem *openLog* i tindrà la següent definició:

> *Function [data,nF,nC] = openLog (logFile)*:
> Aquesta funció se li passa la direcció d'un log generat per l'arquitectura COLA2 i retorna una matriu *(data)* amb totes les dades numèriques del log ordenades. A més a més, també retorna el numero de files *(nF)* i el numero de columnes *(nC)* del log. Per realitzar aquesta tasca, utilitza la funció *importdata* amb la qual separa les lines i acte seguit utilitza la funció *sscanf* per separar els nombres de cada línia.

Un cop solucionat el tema dels logs, referent a la interfície gràfica, em de dir que segueix el mateix disseny que la interfície implementada en el COLA2. En aquest cas també s'utilitzen dos *buffers,* el cíclic i el normal que s'utilitza per generar la imatge de la pantalla. El cíclic l'anomenem *circularBuffer* i el normal *image*.

Pel que fa el temps de reproducció, s'ha optat per reproduir les imatges amb un temps molt aproximat a l'original de la captura, de manera que el temps d'espera entra dos refresc d'imatge és el mateix temps que hi va haver entra la captura dels dos pings.

La figura 7.2 és l'algorisme utilitzat en aquest programa.

Carregar el log
Mostrar els paràmetres de configuració
Per cada línea del log
Obtenir les dades de babord i d'estribord de la línia actual
Pintar una nova línia en el buffer cíclic amb les dades obtingudes
Copiar correctament el buffer ciclic en el buffer normal
Dibuixar el buffer normal per la pantalla.
Controlar el final del buffer cíclic
Esperar un temps
FPer

Figura 7.2: Algoritme basic utilitzat pel programa de representació d'imatges acústiques obtingudes pel robot

El programa ha estat anomenat *SSSLogsView* i un cop implementat amb *Matlab,* ha pogut reproduir correctament tots els arxius especificats. La figura 7.3 mostra una captura del programa mentre es realitzava una reproducció.

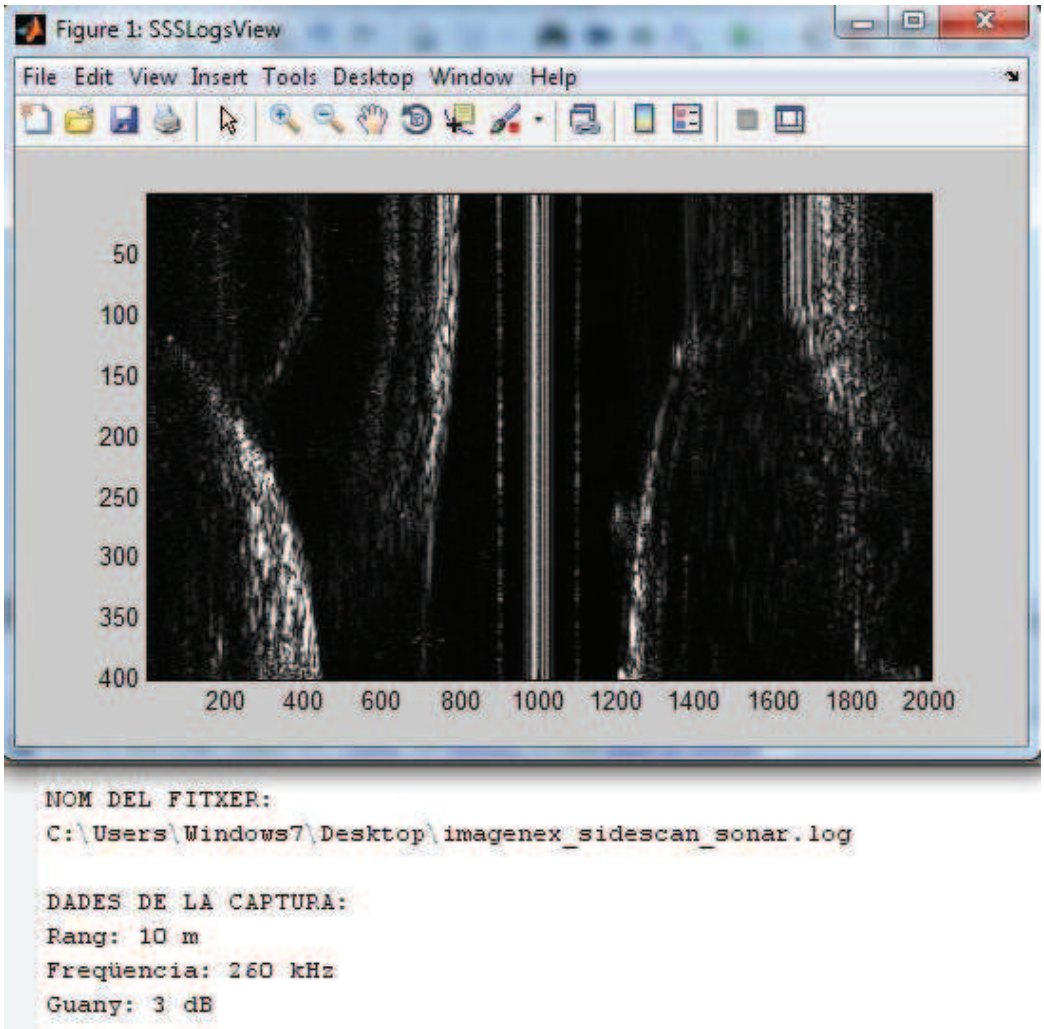



*Figura 7.3: Captura del programa SSSLogsView en execució.*

# 8 Software de reconstrucció d'imatges acústiques segons la navegació

## 8.1 Introducció

En aquest punt, ja és possible capturar i veure amb temps real dades mitjançant el robot Girona 500, i també poder reproduir de manera offline els logs capturats. Ara, es vol introduir un nou element, la navegació, que aportarà informació addicional al sensor la qual serà utilitzada per augmentar el realisme de les dades.

Com ja s'ha vist, el robot Girona 500 obté diferents dades de navegació de diversos sensors. Aquestes dades freqüentment són complementaries i ens indiquen paràmetres de navegació del robot des de diferents punts de vista. Aquest paràmetres són capturats en temps real i guardat en logs, igual que realitza el nostre sensor.

En aquest capítol, desenvoluparem una nova aplicació utilitzant el Matlab, que ens permeti mitjançant un log de navegació, complementar les dades obtingudes amb el nostre sensor.

Així doncs, utilitzant les dades de velocitat del robot, o en el seu defecte de posició, obtingudes durant el mateix moment en el qual es capturaven dades amb el nostra sensor, és possible substituir el temps de la coordenada vertical de les imatges actuals per distancia (vegeu l'esquema de la figura 8.1).

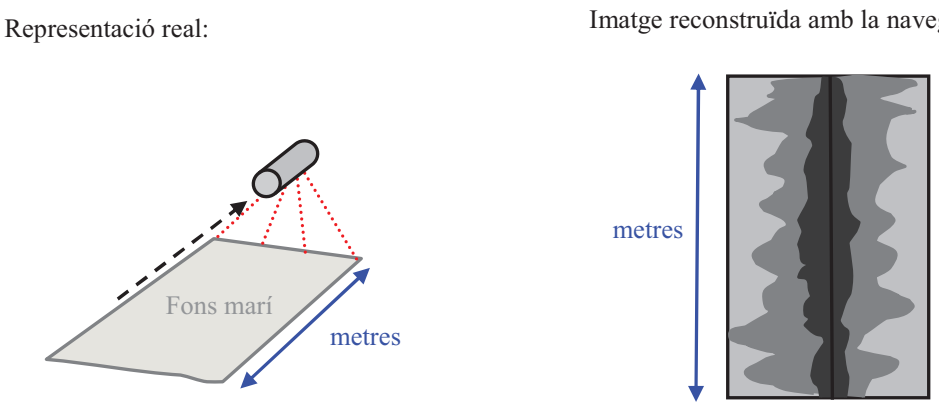



*Figura 8.1: Esquema basic d'una imatge reconstruïda utilitzant la navegació del robot*

Aquesta tasca es realitza de manera offline i es necessari un log capturat pel sonar d'escombrat lateral i un log del sensors de navegació.

## 8.2 El log de navegació del Girona 500

El robot Girona 500 compte amb un log de navegació, anomenat *girona500_navigator*, del qual és possible obtenir es dades de posició relativa i velocitat del robot en tot moment.

Pel nostre objectiu no és necessari conèixer la posició absoluta del robot de manera que per implementar el programa es possible escollir la posició la velocitat

El log de navegació segueix el mateix format explicat pel log del sonar d'escombrat lateral, de manera que únicament canviaran les dades guardades a cada línia del fitxer.

Observant la capçalera del fitxer, podem distingir aquest paràmetres entre les primeres dades:

1. La posició. Expressada en metres, són tres paràmetres: X,Y i Z.
2. El roll, en radiants.
3. El pitch, en radiants.
4. El heading, en radiants.
5. La velocitat, metres per segon, la qual són 3 valors; X, Y i Z.
6. La velocitat de gir, en radiants, amb els valors; roll, pitch i heading.

La figura 8.2 mostra una captura d'un fragment de l'arxiu de navegació del Girona 500.

```
2011 07 22 11 59 46 808 Position(m,X,Y,Z)(1-3)  Roll(rad)(4)    Pitch(rad)(5)   Heading(rad)(6)
2011 07 22 11 59 50 605 -0.0005031 0.0001929 2.354e-07 -0.006981 -0.006981 0    -0.009192 0.003
2011 07 22 11 59 50 614 -0.0005939 0.0002278 4.219e-06 -0.006981 -0.006981 0    -0.009164 0.003
2011 07 22 11 59 50 634 -0.0007792 0.0002991 4.238e-06 -0.006981 -0.006981 0    -0.009106 0.003
2011 07 22 11 59 50 708 -0.001452 0.0005583 4.308e-06 -0.006981 -0.006981 0 -0.008896 0.003444 9
2011 07 22 11 59 50 734 -0.001688 0.0006497 4.333e-06 -0.006981 -0.006981 0 -0.008822 0.003421 9
2011 07 22 11 59 50 834 -0.002567 0.0009906 4.428e-06 -0.006981 -0.006981 0 -0.008547 0.003336 9
2011 07 22 11 59 50 858 -0.002771 0.00107 4.45e-06 -0.006981 -0.006981 0    -0.008484 0.003316 9
2011 07 22 11 59 50 872 -0.003787 0.001052 -0.000249 -0.006981 -0.006981 0  -0.01556 0.002828 -0
2011 07 22 11 59 50 934 -0.00477 0.00123 -0.0003754 -0.006981 -0.006981 0   -0.01525 0.002784 -0
2011 07 22 11 59 51 008 -0.005899 0.001436 -0.0005234 -0.006981 -0.006981 0 -0.0149 0.002732 -0.
2011 07 22 11 59 51 031 -0.006234 0.001498 -0.0005684 -0.006981 -0.006981 0 -0.01479 0.002717 -0
2011 07 22 11 59 51 125 -0.007636 0.001755 6.176e-05 -0.006981 -0.006981 0  -0.01435 0.002652 0.
2011 07 22 11 59 51 134 -0.007762 0.001779 8.229e-05 -0.006981 -0.005236 -8.792e-06 -0.01431 0.0
2011 07 22 11 59 51 152 -0.007896 0.00135 -0.0007631 -0.006981 -0.005236 -9.751e-06 -0.01344 -0.
```

*Figura 8.2: Captura d'un fragment d'un fitxer de navegació del robot Girona 500*

## 8.3 Disseny i implementació del programa

Un cop es coneix el log de navegació del Girona 500, ja es pot dissenyar i implementar un programa em Matlab que ens reconstrueixi la imatge a partir dels dos logs.

Finalment, referent el tema del log de navegació, s'ha decidit utilitzar la posició del robot, ja que pareix més facilitats que utilitzar la velocitat, encara que els dos mètodes són correctes.

En aquest punt, és fa necessària una nova funció que ens permeti calcular d'interval de temps entra dos dates del log. Aquesta funció que d'entrada podria semblar trivial, necessita d'una certa atenció, ja que s'ha de tenir en compte que la data es dona de manera completa i s'haurà de evitar tenir problemes sobretot en els canvis de dia i mes. Aquesta funció, s'anomenarà *msInterval* complirà la següent definició:

*Function ms = msInterval ( DataArray1,dataArray2 )*: Aquesta funció, ens retornarà els milisegons de diferencia entra dos dates passades en forma de matriu de 7 elements, de manera que siguin: any, mes, dia, hora, minut, segon i milisegon, en aquest ordre. Cal remarcar que s'utilitza la funció datenum, la qual ens torna el nombre de dies passats fins a una data data, per evitar problemes amb el nombre de dies dels mesos o hem els anys de traspàs. La funció retorna els milisegons de diferencia en cas que sigui menys de 1 segons, altrament retorna el valor 9999.

En el desenvolupament d'aquesta aplicació, utilitzarem com a base l'aplicació per representar les imatges acústiques obtingudes pel robot realitzada en el darrer capítol, també amb Matlab. El més remarcable és l'ús de la mateixa funció *openLog*, la qual ens retorna una matriu de nombres a partir d'un log.

La tasca d'aquesta aplicació serà generar una imatge sonora a partir de les dades obtingudes del log del sonar d'escombrat lateral, però respectant les distancies en l'eix vertical de la imatge, utilitzant les dades de posició relatives del robot. Per tenir en compte aquest eix, serà necessari mirar únicament els eixos X i Y del robot ja que un desplaçament en l'eix Z no afecta la distancia en la coordenada vertical de la imatge. La figura 8.3 intenta esquematitzar aquest fet.

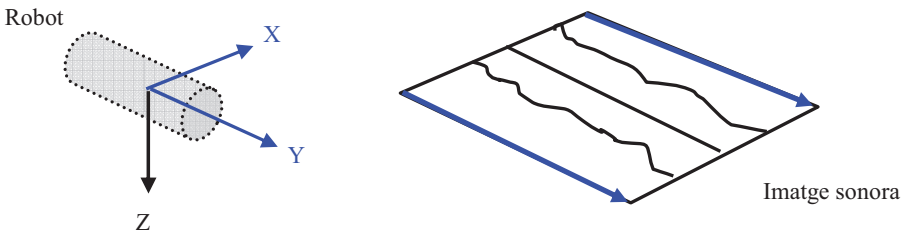



*Figura 8.3: Esquema basic de les coordenades d'un robot i la seva relació amb l'eix vertical d'una imatge sonora.*

Pel que fa a la resta del algorisme, cal destaca, que es basa en trobar per cada imatge lineal sonora, la posició aproximada en la qual va estar obtinguda, i partir d'aquestes posicions calcular la distancia entre cada imatge lineal. La figura 8.4 mostra un esquema conceptual del procés.

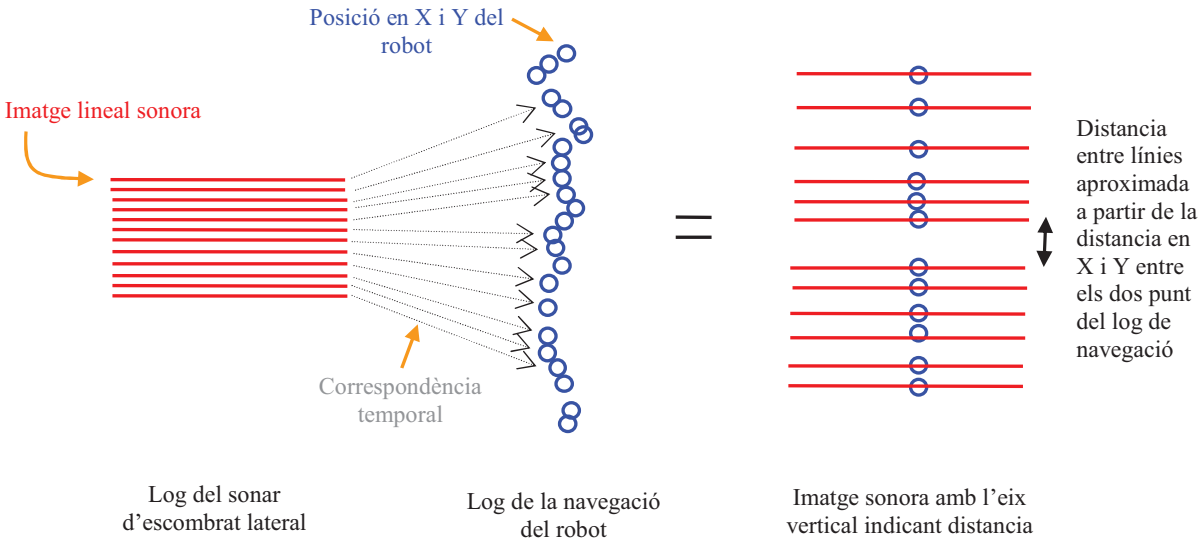



*Figura8.4: Esquema conceptual de la correspondència temporal entra les imatges lineals del sonar i els posicions de la navegació i com la seva composició pot donar lloc a una imatge on l'eix vertical indica distancia.*

.
Per donar major claredat  i certa coherència a les imatge s'utilitzarà la mateixa resolució per l'eix vertical que per l'eix horitzontal, ja que encara que l'eix horitzontal no representa el mateix eix en la imatge que en el fons marí, a diferència del vertical, la utilització de la mateixa resolució donarà més consistència a la imatge. Recordem que a l'eix horitzontal la resolució per píxel és rang/1000.

Per canviar la resolució de l'eix vertical, s'utilitzarà una variable anomenada factResol, factor de resolució, la qual dins el codi, al ser multiplicada per les distancies en metres, les convertirà en la resolució adequada. Aquest factor de resolució s'obtindrà del rang i partir de la formula: factResol = 1000/rang.
Un cop aplicat aquest factor, la imatge resultant tindrà una resolució vertical de rang/1000 igual que la resolució horitzontal.

L'aplicació seguirà l'algoritme de la figura 8.4.

```
Carregar el log del sonar d'escombrat lateral
Carregar el log de navegació del Girona 500
Mostrar els paràmetres de configuració
rang := Obtenir rang de la primera linea del log del sonar
pos2 := posició del robot obtinguda de la primera línia del log de navegació
factResol := 1000/rang
dist := 0
imgPos := 0
Per cada línea del log del sonar
imageDate := Obtenir la data de la línea del sonar
 j := Busquem el numero de línia del log de navegació que te una
            diferencia de temps més petita amb imageDate
pos1 := Obtenim la posició del robot a partir de j
dist := dist + (distancia entre pos1 i pos2) * factResol

Si dist >= 1 llavors
        Obtenim la imatge lineal amb les dades de babord i estribord
        Pintem la imatge final amb la imatge lineal des de imgPos
                fins a (imgPos +dist -1)
        imgPos := imgPos + dist
        dist := 0
FSi

pos2 := pos1
Fper

Crear fitxer amb la imatge
```

Figura 8.4: Algorisme basic utilitzat pel programa de reconstrucció d'imatges acústiques segons la navegació

El programa, una vegada implementat s'ha anomenat SSSwithNavegation.

## 8.4 Proves de funcionament

Un cop s'ha acabat d'implementar l'aplicació i s'ha realitzat una petita prova satisfactòriament, el següent pas, es realitzar un prova en la qual puguem observar el correcta funcionament de la reconstrucció utilitzant la navegació i el seu nivell de precisió.

La prova consistiria en tira un objecta del qual se'n coneguessin les mides a la piscina del CIRS i llavors, utilitzant el Girona 500 de manera teleoperada, es realitzaria una passada per sobre l'objecte, de manera que es pogués capturar una imatge amb el sonar d'escombrat lateral i deduir les mides de l'objecte a partir de la reconstrucció de la imatge utilitzant la navegació.

Finalment, per realitzar aquesta prova es va utilitzar una escala d'alumini de aproximadament de 2,5 metres d'altura. Es va escollir un objecte gros, metàl·lic i em una forma ven definida per facilitar la detecció de l'objecte. La figura 8.5 mostra un esquema de la prova.

Un cop realitzada la prova es van reconstruir les imatges amb el programa SSSwithNavigation i es van examinar.

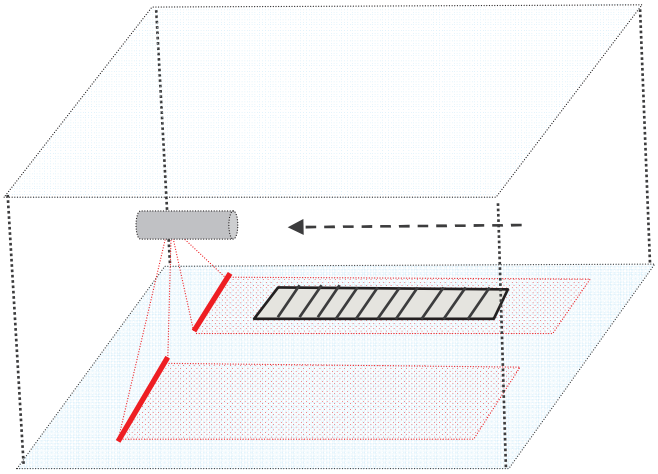


*Figura 8.5: Esquema de la prova de reconèixer una escala.*

Les imatges obtingudes eren una mica fosques però es va poder observar l'escala correctament. Gratament es va comprovar, mesurant els píxels amb l'aplicació Microsoft Paint, que la mida de la escala era de aproximadament 250 píxels amb una resolució de 0.01 metres, la qual cosa ens donava efectivament una mida de 2,5 metres. Tot i l'ús d'un rang molt petit i d'haver realitzat una pesada molt lenta amb el robot, la precisió dels resultats es remarcable i ens indica que el sensor esta dotat d'unes grans qualitats. La figura 8.6 mostra una captura de la mesura de la escala.

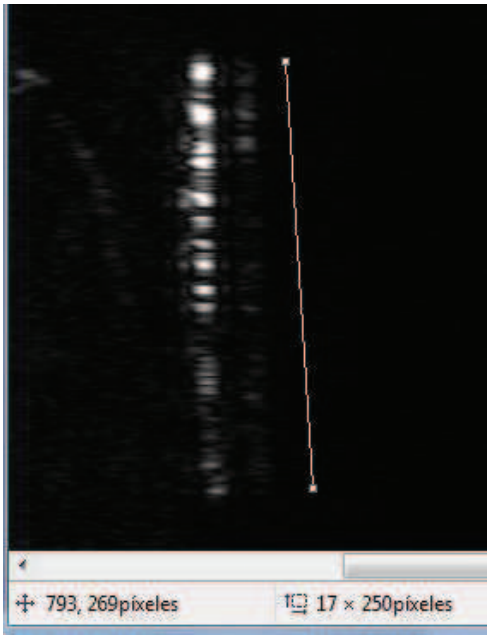



*Figura 8.6: Captura del programa Paint utilitzat per mesurar el píxels de l'escala.*

# 9 Conclusions i treballs futurs

## 9.1 Conclusions

Un cop acabat el projecte, se'n pot fer una valoració molt correcte i satisfactòria. En línies generals, s'han complert tots els objectius proposats en un inici i s'han obtingut tots els resultats esperats.

Cal destacar la gran quantitat de coneixements adquirits durant el transcurs de les tasques, la majoria de l'àmbit informàtic. Aquest aprenentatge continu no ha estat exclusivament de nous coneixements, sinó també de metodologies, experiències, i persones.

Des de un punt de vista general, el projecte avarca una gran quantitat de temes de diferents àmbits els quals s'han anat aprofundien de manera gradual al llarg del seu desenvolupament. Cada tema, s'ha anat descomponent i entenent a mesura que s'aprofundia en el seu contingut.

Es considera que s'han adquirit amplis coneixements sobre robòtica, programació, arquitectures de control i sensors sonar. Més concretament, cal destacar l'aprenentatge de conceptes basic sobre robòtica submarina, com podria ser el principals components que forment part d'un robot submarí, les seves funcionalitat o alguns del propòsits més remarcables del seu disseny. També, d'una manera més amplia i profunda, s'ha estudiat la seva arquitectura de control, amb conceptes basics sobre el seu disseny i amb l'adquisició de capacitats suficients per integrar nous components els sistema. Pel que fa el sonars i en concret el sonar d'escombrat lateral, s'han adquirit i assimilat coneixements basics sobre els seus principis teòrics, les seves funcions i paràmetres de configuració, anàlisis de les imatges generades o conceptes basics sobre els seu funcionament i protocols.

Cal destacar la gran quantitat de coneixement que s'adquireix en tingué que programar sobre una arquitectura de control tant amplia com la COLA2, la qual ens ha obligat a conèixer un gran quantitat de conceptes sobre el seu disseny i el seu model de funcionament. A l'hora que era necessari aprendre les llibreries i codi que l'implementaven i que en alguns casos, un mateix hauria d'utilitzar.

Finalment, també és de gran rellevància dins d'aquest projecte, les tasques d'anàlisi i disseny de les diverses solucions software que s'han tingut que realitzar per cobrir una amplia gama de necessitats, de manera que ha estat necessari l'anàlisi individual de cada problema amb l'objectiu de cercar la solució més adequada dins cada entorn de desenvolupament.

## 9.2 Treballs futurs

A partir de la feina realitzada en aquest projecte, es possible realitzar diversos avanços sobre el mateix tema.

En primer lloc es podria modificar el driver del COLA2 de manera que fos possible reconfigurar el sensor durant l'execució de l'aplicació. Seguint aquesta línia, és podrien adaptar els valors de configuració dinàmicament segons al alguns paràmetres obtinguts pel mateix robot. Una possibilitat seria modificar els valors de rang o del guany segons l'altura del robot respecte el fons.

En en segon lloc, es pot continuar la cerca en el tema de la reconstrucció d'imatges, de manera que es podrien automatitzar algunes de les feines que actualment necessitem realitzar visualment, com per exemple, la detecció d'objectes rellevants a partir de les ombres i els pics d'intensitat que provoquen.

Finalment, també seria possible adaptar el driver actual de l'arquitectura COLA2 a ROS, una nova arquitectura de control instal·lada en els robots en els darrers dies propers a la finalització d'aquest projecte.

# ANNEX A: El format .872

Aquest format és utilitzat pel programa per windows de la companyia Imagenex Yellowfin, el qual acompanyava el sensor.

Els arxius són un conjunt de blocs de 4096 cada un els quals es troben col·locats un darrera l'altre. Cada bloc és una ping capturat, amb les dades d'estribord i de babord.

Cada bloc te els 4096 bytes i distribuïts amb el següent format:

| | |
|---|---|
| 0 | **ASCII '8'** |
| 1 | **ASCII '7'** |
| 2 | **ASCII '2'** |
| 3 | **872 File Version**<br>**0** |
| 4-7 | **Ping Number** – s'incrementa en cada ping<br>El byte 4 és el de més pes, el 7 el de menys. |
| 8-9 | **Total Bytes 'N'** - Numero de bytes que s'escriuen amb un ping<br>El byte 8 és el de més pes, el 9 el de menys.<br>**4096**. |
| 10-11 | **Data Points Per Channel** Numero de bytes per canal<br>**1000** |
| 12 | **Bytes Per Data Point** Numero de bytes per cada punt d'un canal<br>**Sempre 1** |
| 13 | **Data Point Bit Depth** Numero de bits d'intensitat en cada punt del canal.<br>**Sempre 8** |
| 14 | **GPS Type/ Number of GPS Strings** No s'usa<br>**0** |
| 15-16 | **GPS String File Offset** No s'usa<br>**3000** |
| 17-18 | **Event/Annotation Counter** No s'usa |

19-30 **Date** – Data del sistema, acaba amb un caràcter nul. (12 bytes)
"**DD-MMM-YYYY**"

31-39 **Time** – Hora del sistema, acaba amb un caràcter nul. (9 bytes)
"**HH:MM:SS**"

40-44 **Thousandths of Seconds** – Temps del sistema, acaba amb un caràcter nul.
(5 bytes) "**.ttt**"

45 **Operating Frequency**. Freqüència de treball.
0 = 260 kHz
1 = 330kHz
2 = 770 kHz

46 **Range Index** – Números que indiquen els rangs vàlids.
5 - 10m
6 - 20m
7 - 30m
8 - 40m
9 - 50m
10 - 60m
11 - 80m
12 - 100m
13 - 125m
14 - 150m
15 - 200m

47 **Data Gain**. Guany del sistema.
De 0 a100 per cent

48 **Channel Balance**
De 0 a 60. +/-3dB en increments de 0.1 dB

49-50.1 **Repetition Rate.** Temps entre els pings.
El byte 49 és el de menys pes, el 50 el de menys.

51-52 **Sound Velocity.** Velocitat del so..
Velocitat del so(m/s) * 10

53-64 12 bytes de la capçalera del paquet Interface Return Data.

65-999 **Reservat.** Tot zeros.

1000-1999 **Port Channel Echo Data** Dades de babord
1000 bytes ordenats del més llunyà al torpede, al més proper.

2000-2999 **Starboard Channel Echo Data**
1000 bytes ordenats del més proper al torpede, al més llunyà.

3000-3nnn **GPS Strings** – (100 bytes/string)

3nnn-4093 **Zero Fill**

4094-4095 **Pointer to Previous Ping**
El byte 4094 és el de més pes i 4095 el de menys.
Els últims 2 bytes d'aquest ping contenen un nombre de 16 bits que es la suma del numero de bytes d'aquest ping i el ping anterior.
Numero de bytes cap al ping anterior = ((Byte 4094)<<8) | (Byte 4095) = 8192